THE SINGLE BIGGEST PROBLEM IN COMMUNICATION IS THE ILLUSION THAT IT HAS TAKEN PLACE.

GEORGE BERNARD SHAW

WORDS EMPTY AS THE WIND ARE BEST LEFT UNSAID.

HOMER

EVERYTHING BECOMES A LITTLE DIFFERENT AS SOON AS IT IS SPOKEN OUT LOUD.

HERMANN HESSE

LANGUAGE IS A VIRUS FROM OUTER SPACE.

WILLIAM S. BURROUGHS
Contents

I Communication 15

1 Deterministic Protocols 17
   Some Examples of Communication Problems and Protocols 17
   Rigorously Defining Communication Protocols 19
   Rectangles 20
   Balancing Protocols 22
   From Rectangles to Protocols 23
   Lower Bounds Based on Rectangles 25
   Rectangle Covers 33

2 Rank 41
   Lower Bounds on Communication using Rank 43
   Upper bounds on Communication From Rank 46
   Non-negative Rank 51

3 Randomized Protocols 55
   Some Examples of Randomized Protocols 55
   Variants of Randomized Communication Complexity 59
   Public Coins vs Private Coins 61
   Nearly Monochromatic Rectangles 62
4 Numbers On Foreheads
   Some Examples of Protocols in the Number-On-Forehead Model
   Cylinder Intersections
   Lower Bounds from Ramsey Theory

5 Discrepancy
   Jensen’s Inequality in Combinatorics
   Lower Bounds for Inner-Product
   Lower Bounds for Disjointness in the Number-on-Forehead Model
   Using Concentration Bounds to Control Discrepancy

6 Information
   Entropy
   Chain Rule and Conditional Entropy
   Divergence and Mutual Information
   Lower Bound for Indexing
   Randomized Communication of Disjointness
   Lower Bound for Number of Rounds

7 Compressing Communication
   Correlated Sampling
   Compressing a Single Round of Communication
   Compressing Entire Protocols with Low Internal Information
   Direct Sums in Randomized Communication Complexity
   Round Lower bounds for Tree Pointer-Chasing

II Complexity

8 Circuits and Proofs
   Boolean Circuits
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Karchmer-Wigderson Games</td>
<td>146</td>
</tr>
<tr>
<td></td>
<td>Lower bounds on the Depth of Monotone Circuits</td>
<td>148</td>
</tr>
<tr>
<td>8</td>
<td>Monotone Circuit Depth Hierarchy</td>
<td>150</td>
</tr>
<tr>
<td>8</td>
<td>Boolean Formulas</td>
<td>151</td>
</tr>
<tr>
<td>8</td>
<td>Boolean Depth Conjecture</td>
<td>154</td>
</tr>
<tr>
<td>8</td>
<td>Proof Systems</td>
<td>155</td>
</tr>
<tr>
<td>8</td>
<td>Resolution Refutations</td>
<td>155</td>
</tr>
<tr>
<td>8</td>
<td>Cutting Planes</td>
<td>159</td>
</tr>
<tr>
<td>9</td>
<td>Streaming Algorithms and Branching Programs</td>
<td>165</td>
</tr>
<tr>
<td></td>
<td>Lower bounds on streaming algorithms</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td>Lower bounds on branching programs</td>
<td>172</td>
</tr>
<tr>
<td>10</td>
<td>Data Structures</td>
<td>175</td>
</tr>
<tr>
<td></td>
<td>Dictionaries</td>
<td>175</td>
</tr>
<tr>
<td></td>
<td>Maintaining a Set of Numbers</td>
<td>176</td>
</tr>
<tr>
<td></td>
<td>Lower Bounds on Static Data Structures</td>
<td>182</td>
</tr>
<tr>
<td></td>
<td>Lower bounds on Dynamic Data Structures</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>Graph Connectivity</td>
<td>192</td>
</tr>
<tr>
<td>11</td>
<td>Extension Complexity of Polytopes</td>
<td>199</td>
</tr>
<tr>
<td></td>
<td>Basic properties and features of polytopes</td>
<td>199</td>
</tr>
<tr>
<td></td>
<td>Transformations of polytopes</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>Algorithms from Polytopes</td>
<td>205</td>
</tr>
<tr>
<td></td>
<td>Extension Complexity of a Polytope</td>
<td>207</td>
</tr>
<tr>
<td></td>
<td>Slack Matrices</td>
<td>214</td>
</tr>
<tr>
<td></td>
<td>Lower bounds on Extension Complexity</td>
<td>216</td>
</tr>
<tr>
<td>12</td>
<td>Distributed Computing</td>
<td>227</td>
</tr>
<tr>
<td></td>
<td>Coloring the Network</td>
<td>227</td>
</tr>
<tr>
<td></td>
<td>Computing the Diameter of the Network</td>
<td>229</td>
</tr>
<tr>
<td></td>
<td>Computing the Girth of the Network</td>
<td>230</td>
</tr>
</tbody>
</table>
Introduction

Communication complexity is a mathematical theory that addresses a fundamental question:

If two or more people want to determine something about the information they collectively possess, how long does their conversation need to be?

Besides being mathematically interesting, the theory is motivated by the endeavor to understand the limits of computation in its many guises. Perhaps it is not very surprising that communication is so relevant to computation—it is hard to imagine any notion of computation that does not involve communication in one way or another.

There is no better illustration of the power and versatility of communication complexity than one of its earliest applications—to proving lower bounds on the area required to lay out digital chips¹. A chip-design specifies how to compute a function \( f(x_1,\ldots,x_n) \) by laying out the components of the chip on a grid, as in Figure 1. Each component either stores one of the inputs to the function, or does some computation on the values coming from adjacent components. It is vital to minimize the area used in the design, because the area directly relates to the cost, power consumption, reliability and speed of the chip.

Since there are \( n \) inputs, we certainly need area \( n \) to compute the function \( f \). When can we hope to match this simple lower bound? Communication complexity can be used to show that many functions require much larger area, even as large as \( n^2 \), no matter what chip-design is used! The crucial insight is that the chip can be thought of as two people having a conversation to determine the value of \( f \). If a chip-design has area \( A \), one can argue that there must be a way to cut the chip into two parts containing a similar number of inputs, so that only \( \approx \sqrt{A} \) wires are cut, as in the figure. Imagine that one side of the chip represents one person, and the other side represents another person. Then the chip-design describes how \( f \) can be computed by two people using a conversation whose length

¹ Thompson, 1979
is proportional to the number of wires that were cut. If we can show that $f$ requires a conversation of length $t$, this proves that the area $A$ must be at least $t^2$, no matter how the components of the chip are laid out. One can argue that many functions require conversations of length proportional to $n$, and so area that is proportional to $n^2$.

In the years following the basic definitions by Yao\(^2\), communication complexity has established itself as an essential tool for proving lower bounds on all kinds of computational devices. It is general enough that it captures something important about all computational processes. Yet, it is simple and natural enough that beautiful ideas from a wide range of mathematical disciplines—linear algebra, combinatorics, geometry, analysis, probability and information theory—can be used to understand it. So, the theory of communication complexity is simultaneously simple, beautiful, and widely applicable. It is here to stay.

In this book, we give an introduction to the concepts of communication complexity, and show how they can be used to understand many other models of computation.

Each page of the book has a large margin, where you can find references to the relevant literature, diagrams, and additional explanations of arguments in the main text. We encourage the reader to switch back and forth between the margin and the main text, as they find appropriate.

This book is a living document: comments about the content are always appreciated, and the authors intend to keep the book up to date for the foreseeable future.

Acknowledgements

Conventions and Preliminaries

In this chapter, we set up notation and explain some basic facts that are used throughout the book.

Sets, Numbers and Functions

For a positive integer $k$, we use $[k]$ to denote the set $\{1, 2, \ldots, k\}$. $2^k$ denotes the power set of $[k]$, i.e. the family of all subsets of $[k]$. $[k]^{< n}$ denotes the set of all strings of length less than $n$ over the alphabet $[k]$, including the empty string. $|z|$ denotes the length of the string $z$.

All logarithms are computed base 2 unless otherwise specified. A boolean function is a function whose values are in the set $\{0, 1\}$.

Random variables are denoted by capital letters (like $A$) and values they attain are denoted by lower-case letters (like $a$). Events in a probability space will be denoted by calligraphic letters (like $\mathcal{E}$). Given $a = a_1, a_2, \ldots, a_n$, we write $a_{\leq i}$ to denote $a_1, \ldots, a_i$. We define $a_{< i}$ similarly. We write $a_S$ to denote the projection of $a$ to the coordinates specified in the set $S \subseteq [n]$.

Given two real valued functions $f(n), g(n) > 0$, we write $f(n) \leq O(g(n))$ if there are numbers $n_0, c > 0$, such that if $n > n_0$ then $f(n) \leq cg(n)$. We write $g(n) \geq \Omega(f(n))$ when $f(n) \leq O(g(n))$. We write $f(n) \leq o(g(n))$, if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$.

Graphs

A graph on the set $[n]$ (called the vertices) is a collection of sets of size 2 (called edges). A clique $C \subseteq [n]$ in the graph is a subset of the vertices such that every subset of $C$ of size 2 is an edge of the graph. An independent set $I \subseteq [n]$ in the graph is a set that does not contain any edges. A path in the graph is a sequence of vertices $v_1, \ldots, v_n$ such that $\{v_i, v_{i+1}\}$ is an edge for each $i$. A cycle is a path whose first and last vertices are the same. A cycle is called simple if all of its edges are distinct. A graph is said to be connected if there is a path between every two distinct vertices in the graph. A graph is
called a tree if it is connected and has no simple cycles. The degree of a vertex in a graph is the number of edges it is contained in. A leaf in a tree is a vertex of degree one. Every tree has at least one leaf. It follows by induction on \( n \) that every tree of size \( n \) has exactly \( n - 1 \) edges.

### Probability

Throughout this book, we consider only finite probability spaces, or uniform distributions on intervals in \( \mathbb{R} \).

Let \( p \) be a probability distribution on a set \( \Omega \). That is, \( p : \Omega \to [0,1] \) and \( \sum_{a \in \Omega} p(a) = 1 \). Let \( A \) be a random variable chosen according to \( p \). That is, for each \( a \in \Omega \) we have \( \Pr[A = a] = \Pr_p[A = a] = p(a) \). We use the notation \( p(a) \) to denote both the distribution of the variable \( A \), and the number \( \Pr[A = a] \). The meaning will be clear from context. For example, if \( \Omega = \{0,1\}^2 \) and \( A \) is uniformly distributed in \( \Omega \), then \( p(a) \) sometimes denotes the uniform distribution on \( \Omega \), and if \( a = (0,0) \), \( p(a) \) denotes the number \( 1/4 \).

We write \( p(a|b) \) to denote either the distribution of \( A \) conditioned on the event \( B = b \), or the number \( \Pr[A = a|B = b] \). Given a distribution \( p(a,b,c,d) \), we write \( p(a,b,c) \) to denote the marginal distribution on the variables \( a,b,c \) (or the corresponding probability).

We often write \( p(ab) \) instead of \( p(a,b) \) for conciseness of notation. In the example above, if \( B = A_1 + A + 2 \), and \( b = 1 \), then \( p(a|b) \) either denotes the uniform distribution on \( \{(0,1),(1,0)\} \) when \( a \) is a free variable, and if \( a = (0,1) \), \( p(a|b) = 1/2 \).

If \( E \) is an event, we write \( p(E) \) to denote its probability according to \( p \). We denote by \( \mathbb{E}_{p(a)}[g(a)] \) the expected value of \( g(a) \) with respect to \( p \). We write \( A \rightarrow M \rightarrow B \) to assert that \( p(amb) = p(m) \cdot p(a|m) \cdot p(b|m) \).

The statistical distance (also known as total variational distance) between \( p(x) \) and \( q(x) \) is defined to be:

\[
|p - q| = (1/2) \sum_x |p(x) - q(x)| = \max_T p(T) - q(T),
\]

where the maximum is taken over all subsets \( T \) of the universe.

For example, if \( p \) is uniform on \( \Omega = \{0,1\}^2 \) and \( q \) is uniform on \( \{(0,1),(1,0),(1,1)\} \subset \Omega \) then when \( a \) is a free variable, \( |p(a) - q(a)| \) denotes the statistical distance between the distributions, which is \( 1/4 \), and when \( a = (0,0) \), \( |p(a) - q(a)| = 1/4 \).

We sometimes write \( p(x) \approx q(x) \), to indicate that \( |p(x) - q(x)| \leq \epsilon \). Suppose \( A, B \) are two variables in a probability space \( p \). For ease of
notation, we write \( p(a|b) \approx p(a) \) for average \( b \) to mean that
\[
\mathbb{E}_{p(b)} [ |p(a|b) - p(a)| ] \leq \epsilon.
\]

**Some Useful Inequalities**

**Markov’s Inequality**

Suppose \( X \) is a non-negative random variable, and \( \gamma \) is a number. Markov’s inequality bounds the probability that \( X \) exceeds \( \gamma \) in terms of the expected value of \( X \):
\[
\mathbb{E}[X] \geq p(X > \gamma) \cdot \gamma \Rightarrow p(X > \gamma) \leq \mathbb{E}[X] / \gamma.
\]

**Chernoff-Hoeffding Bound and Estimates on Binomial Coefficients**

The Chernoff-Hoeffding bound controls the concentration of the sum of independently distributed bits around its expectation. Suppose \( X_1, \ldots, X_n \) are independent identically distributed bits. Let \( \mu = \mathbb{E} [\sum_{i=1}^n X_i] \). The Chernoff-Hoeffding bound says that for any \( 0 < \delta < 1 \),
\[
\Pr \left[ \sum_{i=1}^n X_i - \mu > \delta \mu \right] \leq e^{-\delta^2 \mu / 3}.
\]
The following form of the bound applies when \( \delta > 1 \),
\[
\Pr \left[ \sum_{i=1}^n X_i > (1 + \delta) \mu \right] \leq e^{-\delta \mu / 3}.
\]
When \( X_1, \ldots, X_n \) are uniformly random bits, the bounds above give estimates on binomial coefficients. For a number \( 0 \leq a \leq n/2 \), we have:
\[
\sum_{k \in [n]:|k-n/2|>a} \binom{n}{k} \leq 2^n \cdot e^{-4a^2 / 3n}.
\]
We also have the following upper bounds on binomial coefficients: for all \( k \in [n] \),
\[
\binom{n}{k} \leq 2^{n+1} / \sqrt{\pi n}.
\]

**Approximating Linear Functions by Exponentials**

We will often need to approximate linear functions with exponentials:
\( e^{-x} \geq 1 - x \) when \( x \geq 0 \), and \( 1 - x \geq 2^{-2x} \) when \( 0 \leq x \leq 1/2 \).
**Cauchy-Schwartz Inequality**

The Cauchy-Schwartz inequality says that for two vectors \( x, y \in \mathbb{R}^n \), their inner product is at most the products of their lengths:

\[
\left| \sum_{i=1}^n x_i y_i \right| = |\langle x, y \rangle| \leq \|x\| \cdot \|y\| = \sqrt{\sum_{i=1}^n x_i^2} \cdot \sqrt{\sum_{i=1}^n y_i^2}.
\]

**Jensen’s Inequality and Convexity**

A function \( f : \mathbb{R} \to \mathbb{R} \) is said to be convex if

\[
\frac{f(x) + f(y)}{2} \geq f \left( \frac{x + y}{2} \right),
\]

for all \( x, y \) in the domain. It is said to be concave if

\[
\frac{f(x) + f(y)}{2} \leq f \left( \frac{x + y}{2} \right).
\]

Some convex functions: \( x^2, e^x, x \log x \). Some concave functions: \( \log x, \sqrt{x} \).

Jensen’s inequality says if a function \( f \) is convex, then \( \mathbb{E}[f(X)] \geq f(\mathbb{E}[X]) \), for any real-valued random variable \( X \). Similarly, if \( f \) is concave, then \( \mathbb{E}[f(X)] \leq f(\mathbb{E}[X]) \). In this book, we often say that an inequality follows by convexity when we mean that it can be derived by applying Jensen’s inequality to some function.

A consequence of Jensen’s inequality is the Arithmetic-Mean Geometric-Mean inequality:

\[
\frac{\sum_{i=1}^n a_i}{n} \geq \left( \prod_{i=1}^n a_i \right)^{1/n},
\]

which can be proved using the concavity of the log function:

\[
\log \left( \frac{\sum_{i=1}^n a_i}{n} \right) \geq \frac{\sum_{i=1}^n \log a_i}{n} = \log \left( \prod_{i=1}^n a_i^{1/n} \right).
\]

Try to prove the Cauchy-Schwartz inequality using convexity.
Part I

Communication
1

Deterministic Protocols

To understand the mathematics of communication, we need to give a rigorous definition that specifies what a conversation between people really is, and say how we can measure its complexity. However, the concept of communication is so fundamental to humans that we can appreciate the concept even before a rigorous definition is in place.

With that in mind, we begin by discussing some interesting examples of communication problems. This will set the stage for the rigorous discussion later. In the sections that follow, we provide mathematical definitions of communication protocols and related concepts, and prove some basic facts about them.

Some Examples of Communication Problems and Protocols

A communication protocol specifies a way for a set of people to have a conversation. Each person has access to a different source of information. Their goal is to learn some feature of all the information that they collectively know.

Equality Suppose two people named Alice and Bob are given two $n$-bit strings. Alice is given $x$ and Bob is given $y$, and they want to know if $x = y$. There is a trivial solution: Alice can send her input $x$ to Bob, and Bob can let her know if $x = y$. This is a deterministic protocol that takes $n + 1$ bits of communication. Interestingly, we shall prove that no deterministic protocol is more efficient. On the other hand, for every number $k$, there is a randomized protocol that uses only $k + 1$ bits of communication and errs with probability at most $2^{-k}$: the parties can use randomness to hash their inputs and compare the hashes. More on this in Chapter 3.

Cliques and Independent Sets Here Alice and Bob are given a graph $G$ on $n$ vertices. In addition, Alice knows a clique $C$ in the graph,
and Bob knows an independent set \( I \) in the graph. They want to know whether \( C \) and \( I \) share a common vertex or not, and they want to know this using a short conversation. Describing \( C \) or \( I \) takes about \( n \) bits, because in general the graph may have \( 2^n \) cliques or \( 2^n \) independent sets. So, if Alice and Bob try to tell each other what \( C \) or \( I \) is, that will lead to a very long conversation.

Here we discuss a clever interactive protocol allowing Alice and Bob to have an extremely short conversation for this task. They will send at most \( O(\log^2 n) \) bits. If \( C \) contains a vertex \( v \) of degree less than \( n/2 \), Alice sends Bob the name of \( v \). This takes just \( O(\log n) \) bits of communication. Now, either \( v \in I \), or Alice and Bob can safely discard all the non-neighbors of \( v \), since these cannot be a part of \( A \). This eliminates at least \( n/2 \) vertices from the graph. Similarly, if \( I \) contains a vertex \( v \) of degree at least \( n/2 \), Bob sends Alice the name of \( v \). Again, either \( v \in C \), or Alice and Bob can safely delete all the neighbors of \( v \) from the graph, which eliminates about \( n/2 \) vertices. If all the vertices in \( C \) have degree more than \( n/2 \), and all the vertices in \( I \) have degree less than \( n/2 \), then \( C \) and \( I \) do not share a vertex. The conversation can safely terminate. So, in each round of communication, either the parties know that \( C \cap I = \emptyset \), or the number of vertices is reduced by a factor of 2. After \( k \) rounds, the number of vertices is at most \( n/2^k \). If \( k \) exceeds \( \log n \), the number of vertices left will be less than 1, and Alice and Bob will known if \( C \) and \( I \) share a vertex or not. This means that at most \( \log n \) vertices can be announced before the protocol ends, proving that at most \( O(\log^2 n) \) bits will be exchanged before Alice and Bob learn what they wanted to know.

One can show if the conversation involves only one message from each party, then at least \( \Omega(n) \) bits must be revealed for the parties to discover what they want to know. So, interaction is vital to bringing down the length of the conversation.

**Disjointness with sets of size \( k \)** Alice and Bob are given two sets \( A, B \subseteq [n] \), each of size \( k \ll n \), and want to know if the sets share a common element. Alice can send her set to Bob, which takes \( \log \binom{n}{k} \approx k \log n \) bits of communication. There is a randomized protocol that uses only \( O(k) \) bits of communication. Alice and Bob sample a random sequence of sets in the universe, Alice announces the name of the first set that contains \( A \). If \( A \) and \( B \) are disjoint, this eliminates half of \( B \). In Chapter 3, we prove that repeating this procedure gives a protocol with \( O(k) \) bits of communication.

**Disjointness with \( k \) parties** The input is \( k \) sets \( A_1, \ldots, A_k \subseteq [n] \), and there are \( k \) parties. The \( i \)’th party knows all the sets except for
the $i$'th one. The parties want to know if there is a common element in all sets. We know of a clever deterministic protocol with $O(n/2^k)$ bits of communication, and we know that $\Omega(n/4^k)$ bits of communication are required. No randomized protocol can have communication less than $\Omega(\sqrt{n}/2^k)$, but we do not know how to use randomness to reduce the communication.

**Summing 3 numbers** The input is three numbers $x, y, z \in [n]$. Alice knows $(x, y)$, Bob knows $(y, z)$ and Charlie knows $(x, z)$. The parties want to know whether or not $x + y + z = n$. Alice can tell Bob $x$, which would allow Bob to announce the answer. This takes $O(\log n)$ bits of communication. There is a clever deterministic protocol that communicates $\sqrt{\log n}$ bits, and one can show that the length of any deterministic conversation must increase with $n$. To contrast, there is a randomized protocol that solves the problem with a conversation whose length is a constant.

**Pointer Chasing** The input consists of two functions $f, g : [n] \to [n]$, where Alice knows $f$ and Bob knows $g$. Let $a_0, a_1, \ldots, a_k \in [n]$ be defined by setting $a_0 = 1$, and $a_i = f(g(a_{i-1}))$. The goal is to compute $a_k$. There is a simple $k$ round protocol with communication $O(k \log n)$ that solves this problem, but any protocol with fewer than $k$ rounds requires $\Omega(n)$ bits of communication.

**Rigorously Defining Communication Protocols**

Here we define exactly what we mean by a 2 party deterministic communication protocol. The definition is meant to capture a conversation between the two players. Suppose Alice’s input comes from a set $X$ and Bob’s input comes from $Y$. A protocol $\pi$ is specified by

---

*Figure 1.2: An execution of a protocol.*
a rooted binary tree. Every internal vertex $v$ has 2 children. Every internal vertex $v$ is associated with either the first or second party, and a function $f_v : X \rightarrow \{0, 1\}$, or $f_v : Y \rightarrow \{0, 1\}$ mapping the input known to that party to a bit that can be interpreted as a child of $v$.

Given inputs $(x,y) \in X \times Y$, the outcome of the protocol $\pi(x,y)$ is a leaf in the protocol tree, computed as follows. The parties begin by setting the current vertex to be the root of the tree. If the first party is associated with the current vertex $v$, she announces the value $f_v(x)$. Similarly, if the second party is associated with $v$, he announces the value $f_v(y)$. Both parties set the new current vertex to be the child of $v$ indicated by the announced value of $f_v$. This process is repeated until the current vertex is a leaf, and this leaf is the outcome of the protocol. So, the inputs $(x,y)$ induce a path from the root of the protocol tree to the leaf $\pi(x,y)$. This path corresponds to the conversation between the parties.

The length of the protocol $\pi$, denoted $\|\pi\|$, is the depth of the protocol tree\(^a\). The number of rounds of the protocol is $k$ if $k$ is the smallest number such that the execution of the protocol involves the parties exchanging at most $k$ binary strings. In other words, it is the maximum, over all root-leaf paths, of the number of alternations between the players along the path.

Given a function $g : X \times Y \rightarrow Z$ we say that $\pi$ computes $g$ if $\pi(x,y)$ determines $g(x,y)$ for every input $(x,y) \in X \times Y$. In other words, we say $\pi$ computes $g$ if there is a map $D$ from the leaves of the protocol tree to $Z$ so that $D(\pi(x,y)) = g(x,y)$ for all $x,y$. The communication complexity of a function $g$ is $c$ if there is protocol of length $c$ that computes $g$, but no protocol can compute the function with less than $c$ bits of communication.

Let us make some basic observations that follow from these definitions:

**Fact 1.1.** The number of rounds in $\pi$ is always at most $\|\pi\|$.

**Fact 1.2.** The number of leaves in the protocol tree of $\pi$ is at most $2^{\|\pi\|}$.

**Rectangles**

The concept of combinatorial rectangles plays a crucial role in our understanding of communication complexity. A rectangle is a subset of the form $R = A \times B \subseteq X \times Y$. See Figure 1.3.

**Lemma 1.3.** A set $R \subseteq X \times Y$ is a rectangle if and only if whenever $(x,y),(x',y') \in R$, we have $(x',y), (x,y') \in R$.

For brevity, we often say rectangles, instead of combinatorial rectangles.

For $k$ party protocols, a rectangle is a cartesian product of $k$ sets.

\(^a\)The length of the longest path from root to leaf.
Proof. If $R = A \times B$ is a rectangle, then $(x, y), (x', y') \in R$ means that $x, x' \in A$ and $y, y' \in B$. Thus $(x, y'), (x', y) \in A \times B$. On the other hand, if $R$ is an arbitrary set with the given property, if $R$ is empty, it is a rectangle. If $R$ is not empty, let $(x, y) \in R$ be an element. Define $A = \{x' : (x', y) \in R\}$ and $B = \{y' : (x, y') \in R\}$. Then by the property of $R$, we have $A \times B \subseteq R$, and for every element $(x', y') \in R$, $x' \in A, y' \in B$, so $R \subseteq A \times B$. Thus $R = A \times B$. □

If a function is defined by a rectangle $A \times B$, then it certainly has a very simple communication protocol. If Alice and Bob want to know if their inputs belong to the rectangle or not, Alice can send a bit indicating if $x \in A$, and Bob can send a bit indicating if $y \in B$. These two bits determine whether or not $(x, y) \in A \times B$.

The importance of rectangles stems from the fact that every protocol can be described using rectangles. For every vertex $v$ in a protocol $\pi$, let $R_v \subseteq X \times Y$ denote the set of inputs $(x, y)$ that would lead the protocol to pass through the vertex $v$ during the execution, and let

$X_v = \{x \in X : \exists y \in Y \ (x, y) \in R_v\}$,

$Y_v = \{y \in Y : \exists x \in X \ (x, y) \in R_v\}$.

**Lemma 1.4.** For every vertex $v$ in the protocol tree, $R_v$ is a rectangle with $R_v = X_v \times Y_v$. Moreover, the rectangles given by all the leaves of the protocol tree form a partition of the set of inputs $X \times Y$.

Proof. The lemma follows by induction. For the root vertex $r$, we see that $R_r = X \times Y$, so indeed the lemma holds. Now consider an arbitrary vertex $v$ such that $R_v = X_v \times Y_v$. Let $u, w$ be the children of $v$
Figure 1.4: A partition of the space into 6 rectangles.

in the protocol tree. Suppose the first party is associated with $v$, and $u$ is the vertex that the players move to when $f_v(x) = 0$. Define:

$$X_u = \{x \in X_v : f_v(x) = 0\},$$

$$X_w = \{x \in X_v : f_v(x) = 1\}.$$

We see that $X_u$ and $X_w$ form a partition of $X_v$, and $R_u = X_u \times Y_u$ and $R_w = X_w \times Y_w$ form a partition of $R_v$. In this way, we see that the leaves in the protocol tree induce a partition of the entire space of inputs into rectangles.

Often, the purpose of the protocol is to compute a function $g : X \times Y \rightarrow \{0, 1\}$. In this case, it is useful to understand the concept of a monochromatic rectangle. We say that a rectangle $R \subset X \times Y$ is monochromatic with respect to $g$ if $g$ is constant on $R$. See Figure 1.5. We say that the rectangle is 1-monochromatic if $g$ only takes the value 1 on the rectangle, and 0-monochromatic if $g$ only takes the value 0 on $R$.

**Fact 1.5.** If a protocol $\pi$ computes a function $g : X \times Y \rightarrow \{0, 1\}$, and $v$ is a leaf in $\pi$, then $R_v$ is a monochromatic rectangle.

Combining this fact with Lemmas 1.2 and 1.4 gives:

**Theorem 1.6.** If the communication complexity of $g : X \times Y \rightarrow \{0, 1\}$ is $c$, then $X \times Y$ can be partitioned into at most $2^c$ monochromatic rectangles.

**Balancing Protocols**

**Lemma 1.2** is sharp when the protocol tree is a full binary tree; then the number of leaves in the protocol tree is exactly $2^c$. Does it ever make sense to have a protocol tree that is not balanced? It turns out that one can always balance an unbalanced tree while approximately preserving the number of nodes in the tree.
Theorem 1.7. If \( \pi \) is a protocol with \( \ell \) leaves, then there is a protocol that computes the outcome \( \pi(x, y) \) with length at most \( |2\log_{3/2} \ell| \).

To prove the theorem, we need a well-known lemma about trees.

Lemma 1.8. In every protocol tree with \( \ell > 1 \) leaves, there is a vertex \( v \) such that the subtree rooted at \( v \) contains \( r \) leaves, and \( \ell/3 \leq r < 2\ell/3 \).

Proof. Consider the sequence of vertices \( v_1, v_2, \ldots \) defined as follows. The vertex \( v_1 \) is the root of the tree, which is not a leaf by the assumption on \( \ell \). For each \( i > 0 \), the vertex \( v_{i+1} \) is the child of \( v_i \) that has the most leaves under it, breaking ties arbitrarily. Let \( \ell_i \) denote the number of leaves in the subtree rooted at \( v_i \). Then, \( \ell_{i+1} \geq \ell_i/2 \), and \( \ell_{i+1} < \ell_i \). Since \( \ell_1 = \ell \), and the sequence is decreasing until it hits 1, there must be some \( i \) for which \( \ell/3 \leq \ell_i < 2\ell/3 \). \( \square \)

Proof of Theorem 1.7. In each step of the balanced protocol (see Figure 1.7), the parties pick a vertex \( v \) as promised by Lemma 1.8, and decide whether \( (x, y) \in R_v \) using two bits of communication. That is, Alice sends a bit indicating if \( x \in X_v \) and Bob sends a bit indicating if \( y \in Y_v \). If \( x \in X_v \) and \( y \in Y_v \), then the parties repeat the procedure at the subtree rooted at \( v \). Otherwise, the parties delete the vertex \( v \) and its subtree from the protocol tree and continue the simulation. In each step, the number of leaves of the protocol tree is reduced by a factor of at least \( 2/3 \), so there can be at most \( \log_{3/2} \ell \) such steps. \( \square \)

From Rectangles to Protocols

Given Theorem 1.6, one might wonder whether every partition of the inputs to rectangles can be realized by a protocol. While this is not true in general (see Figure 1.8), we can show that a small partition yields an efficient protocol.

Theorem 1.9. If \( X \times Y \) can be covered by \( 2^c \) monochromatic rectangles with respect to \( g \), then there is a protocol that computes \( g \) with \( O(c^2) \) bits of communication.

One can prove Theorem 1.9 by reduction to the clique versus independent set problem—we leave the details as an exercise. Here we prove a more general statement:

Theorem 1.10. Let \( R \) be a collection of \( 2^c \) rectangles that form a partition of \( X \times Y \). For \( (x, y) \in X \times Y \), let \( R_{x,y} \) be the unique rectangle in \( R \) that contains \( (x, y) \). Then there is a protocol of length \( O(c^2) \) that on input \( (x, y) \) computes \( R_{x,y} \).
We now prove Theorem 1.10. The parties are given inputs $(x, y)$ and know a collection of rectangles $\mathcal{R}$ that partition the set of inputs. The aim of the protocol is to find the unique rectangle containing $(x, y)$. In each round of the protocol, one of the parties announces the name of a rectangle in $\mathcal{R}$. We shall ensure that each such announcement allows the parties to discard at least half of the remaining rectangles.

A key concept we need is that of rectangles intersecting horizontally and vertically. We say that two rectangles $R = A \times B$ and $R' = A' \times B'$ intersect horizontally if $A$ intersects $A'$, and intersect vertically if $B$ intersects $B'$. The basic observation is that if $x \in A \cap A'$ and $y \in B \cap B'$, then $(x, y) \in A \times B$ and $(x, y) \in A' \times B'$. This proves:

**Fact 1.11.** If $R, R'$ are disjoint rectangles, they cannot intersect both horizontally and vertically.

This leads to the following definition:

**Definition 1.12.** Say that a rectangle $R = (A \times B) \in \mathcal{R}$ is

- horizontally good if $x \in A$, and $R$ horizontally intersects at most half of the rectangles in $\mathcal{R}$, and

- vertically good if $y \in B$, and $R$ vertically intersects at most half of the rectangles in $\mathcal{R}$.

Say that $R$ is good if it is either horizontally good or vertically good.

We claim that there is always at least one good rectangle:

**Claim 1.13.** $R_{x,y}$ is good.

**Proof.** Fact 1.11 implies that every rectangle in $\mathcal{R}$ does not intersect $R_{x,y}$ both horizontally and vertically. Thus either at most half of the rectangles in $\mathcal{R}$ intersect $R_{x,y}$ horizontally, or at most half of them intersect $R_{x,y}$ vertically. See Figure 1.10. 

Figure 1.9: Rectangles 1 and 2 intersect vertically, while rectangles 1 and 3 intersect horizontally.

An efficient partition of the 1’s of the input space into rectangles also leads to an efficient protocol (Exercise 1.1).
In each step of the protocol, one of the parties announces the name of a good rectangle $R$, which must exist, since $R_{x,y}$ is good. This leads to half of the rectangles in $\mathcal{R}$ being discarded. If $R$ is horizontally good, then the players can discard all the rectangles that do not intersect $R$ horizontally. Otherwise they discard all the rectangles that do not intersect $R$ vertically.

When only one rectangle remains, the protocol achieves its goal. Since $\mathcal{R}$ can survive at most $c$ such discards, and a rectangle in the family can be described with $c$ bits of communication, the communication complexity of the protocol is at most $O(c^2)$.

Recent work\(^4\) has shown that there is function $g$ under which the inputs can be partitioned into $2^c$ monochromatic rectangles, yet no protocol can compute $g$ using $o(c^2)$ bits of communication, showing that Theorem 1.9 is tight.

**Lower Bounds Based on Rectangles**

We turn to proving that some problems do not have efficient protocols. The easiest way to prove a lower bound is to use the characterization provided by Theorem 1.6. If we can show that the inputs cannot be partitioned into $2^c$ monochromatic rectangles, or do not have large monochromatic rectangles, then that proves that there is no protocol computing the function with $c$ bits of communication.

\(^4\) Göös et al., 2015; and Kothari, 2015
Size of Monochromatic Rectangles

Equality Consider the equality function $EQ : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$ defined as:

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases} \quad (1.1)$$

Alice can send Bob her input, and Bob can respond with the value of a function, giving a protocol of length $n + 1$.

Is there a protocol with complexity $n$? Since any such protocol induces a partition into $2^n$ monochromatic rectangles, a first attempt at proving a lower bound might try and show that there is no large monochromatic rectangle. If we could prove that, then we could argue that many monochromatic rectangles are needed to cover the whole input. However, the equality function does have large monochromatic rectangles. For example, the rectangle $R = \{(x,y) : x_1 = 0, y_1 = 1\}$. This is a rectangle with density $\frac{1}{4}$, and it is monochromatic, since $EQ(x,y) = 0$ for every $(x,y) \in R$.

The solutions is to show that equality does not have a large 1-monochromatic rectangle, and argue that this is good enough to prove a lower bound.

**Claim 1.14.** If $R$ is a 1-monochromatic rectangle, then $|R| = 1$.

**Proof.** Observe that if $x \neq x'$, then the points $(x, x)$ and $(x', x')$ cannot be in the same monochromatic rectangle. Otherwise, by Lemma 1.3, $(x, x')$ would also have to be included in this rectangle. Since the rectangle is monochromatic, we would have $EQ(x, x') = EQ(x, x)$, which is a contradiction. \hfill \Box

Since there are $2^n$ inputs $x$ with $EQ(x, x) = 1$, this means $2^n$ rectangles are needed to cover such inputs. There is also at least one more 0-monochromatic rectangle. So, we need more than $2^n$ monochromatic rectangles to cover all the inputs. We conclude:

**Theorem 1.15.** The deterministic communication complexity of $EQ$ is $n + 1$.

Inner-Product The Hadamard Matrix is a well known example of a matrix that has many nice combinatorial properties. It corresponds to the inner-product function $IP : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$ over $\mathbb{F}_2$ defined by

$$IP(x,y) = \sum_{i=1}^{n} x_i y_i \mod 2. \quad (1.2)$$
Here we can leverage some linear algebra to place a bound on the size of the largest monochromatic rectangle. Suppose $R = A \times B$ is a 0-monochromatic rectangle. This means that for every $x \in A$ and $y \in B$, we have $\text{IP}(x, y) = 0$. Thus, $B$ must be contained in the dual space of $A$. If $d_1$ denotes the dimension of the $\mathbb{F}_2$-span of $A$ and $d_2$ denotes the dimension of the $\mathbb{F}_2$-span of $B$, this means that $d_1 + d_2 \leq n$. Thus $|R| = |A||B| \leq 2^{d_1} \cdot 2^{d_2} \leq 2^n$.

Let us compute the number of inputs to $\text{IP}$ for which $\text{IP}(x, y) = 0$. When $x = 0$, the inner product is always 0. This gives $2^n$ inputs. When $x \neq 0$, exactly half the settings of $y$ must give 0. This is because if say $x_i = 1$ for some $i$, then for any input $y$, let $y'$ be the same as $y$ except in the $i$'th coordinate. Then we see that $\text{IP}(x, y) - \text{IP}(x, y') = 1 \mod 2$. So exactly half of the $y$'s must give $\text{IP}(x, y) = 0$ when $x \neq 0$. Thus the number of inputs for which $\text{IP}(x, y) = 0$ is $2^n + (2^n - 1)2^n / 2$.

To conclude, at least

$$\frac{2^n + (2^n - 1)2^n / 2}{2^n} = 1 + (2^n - 1)/2 > 2^{n-1}$$

0-rectangles are needed to cover the 0 inputs.

So, we get:

**Theorem 1.16.** The deterministic communication complexity of $\text{IP}$ is at least $n$.

**Disjointness** Next, consider the disjointness function $\text{Disj} : 2^{[n]} \times 2^{[n]} \to$
{0, 1} defined by:

$$\text{Disj}(X, Y) = \begin{cases} 
1 & \text{if } X \cap Y = \emptyset, \\
0 & \text{otherwise}.
\end{cases}$$ (1.3)

Alice can send her whole set $X$ to Bob, which gives a protocol with communication $n + 1$. Can we prove that this is optimal? Once again, this function does have large monochromatic rectangles, for example the rectangle $R = \{(X, Y) : 1 \in X, 1 \in Y\}$, but we shall show that there are no large monochromatic 1-rectangles. Indeed, suppose $R = A \times B$ is a 1-monochromatic rectangle. Let $X' = \bigcup_{X \in A} X$ and $Y' = \bigcup_{Y \in B} Y$. Then $X'$ and $Y'$ must be disjoint, so $|X'| + |Y'| \leq n$. On the other hand, $|A| \leq 2^{|X'|}, |B| \leq 2^{|Y'|}$, so $|R| = |A||B| \leq 2^n$. We have shown:

**Claim 1.17.** Every 1-monochromatic rectangle of $\text{Disj}$ has size at most $2^n$.

On the other hand, the number of disjoint pairs $(X, Y)$ is exactly $3^n$. That’s because for every element of the universe, there are 3 possibilities: to be in $X$, be in $Y$ or be in neither. Thus, at least $3^n/2^n = 2^{\log 3 - 1}n$ monochromatic rectangles are needed to cover the 1’s of $\text{Disj}$, an so :

**Theorem 1.18.** The deterministic communication complexity of $\text{Disj}$ is at least $(\log 3 - 1)n$. 

We shall soon prove an optimal lower bound for disjointness.
Richness

Sometimes we need to understand asymmetric communication protocols, where we need separate bounds on the communication complexity of Alice and Bob. The concept of richness is useful here:

**Definition 1.19.** A function $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ is said to be $(u, v)$-rich if there is a set $V \subseteq \mathcal{Y}$ of size $|V| = v$ such that for all $y \in V$, we have $|\{x \in \mathcal{X} : g(x, y) = 1\}| \geq u$.

Richness allows us to carefully control the shape of large 1-monochromatic rectangles induced by asymmetric protocols:

**Lemma 1.20.** If $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ is $(u, v)$-rich with $u, v > 0$, and if there is a protocol for computing $g$ where Alice sends at most $a$ bits and Bob sends at most $b$ bits, then $g$ admits a $\frac{u^2}{2^a - 1} \times \frac{v^2}{2^b - 1}$-monochromatic rectangle.

**Proof.** The statement is proved inductively. For the base case, if the protocol does not communicate at all, then $g(x, y) = 1$ for all $x \in \mathcal{X}, y \in \mathcal{Y}$, and the statement holds.

If Bob sends the first bit of the protocol, then Bob partitions $\mathcal{Y} = \mathcal{Y}_0 \cup \mathcal{Y}_1$. One of these two sets must have at least $v/2$ of the inputs $y$ that show that $g$ is $(u, v)$-rich. By induction, this set contains a $\frac{u^2}{2^a - 1} \times \frac{v^2}{2^b - 1}$-monochromatic rectangle, as required. On the other hand, if Alice sends the first bit, then this bit partitions $\mathcal{X}$ into two sets $\mathcal{X}_0, \mathcal{X}_1$. Every input $y \in \mathcal{Y}$ that has $u$ ones must have $u/2$ ones in either $\mathcal{X}_0$ or $\mathcal{X}_1$. Thus there must be at least $v/2$ choices of inputs $y \in \mathcal{Y}$ that have $u/2$ ones for $g$ restricted to $\mathcal{X}_0 \times \mathcal{Y}$ or for $g$ restricted to $\mathcal{X}_1 \times \mathcal{Y}$. By induction, we get that there is a 1-monochromatic rectangle with dimensions $\frac{u^2}{2^a - 1} \times \frac{v^2}{2^b - 1}$, as required.

Now let us see some examples where richness can be used to prove lower bounds.

**Lopsided Disjointness** Suppose Alice is given a set $X \subseteq [n]$ of size $k < n$, and Bob is given a set $Y \subseteq [n]$, and they want to compute whether the sets are disjoint or not. Now the obvious protocol is for Alice to send her input to Bob, which takes $\log(\binom{n}{k})$ bits. However, what can we say about the communication of this problem if Alice is forced to send much less than $\log(\binom{n}{k})$ bits?

To prove a lower bound, we need to analyze rectangles of a certain shape. We restrict our attention to special family of sets for Alice and Bob, as in Figure 1.16. Let $n = 2kt$, and suppose $Y$ contains exactly one element of $2i - 1, 2i$, for each $i$, and that $X$ contains exactly one element from $2t(i - 1) + 1, \ldots, 2ti$ for each $i \in [k]$.

**Claim 1.21.** If $A \times B$ is a 1-monochromatic rectangle, then $|B| \leq 2^{kt-k|A|^{1/k}}$. 

---

5 Miltersen et al., 1998

6 In Chapter 2, we show that the communication complexity of this problem is at least $\log(\binom{n}{k})$. 

---
Proof. We claim that $\left| \bigcup_{X \in A} X \right| \geq k|A|^{1/k}$. Indeed, if the union $\bigcup_{X \in A} X$ has $a_i$ elements in $\{2t(i-1)+1, \ldots, 2ti\}$, then
\[
\left| \bigcup_{X \in A} X \right| = \sum_{i=1}^{k} a_i \geq k \left( \prod_{i=1}^{k} a_i \right)^{1/k} \geq k|A|^{1/k}.
\]
by the arithmetic-mean, geometric mean inequality.

$\bigcup_{X \in A} X$ cannot contain both $2i, 2i+1$ for any $i$, since one of these two elements belongs to a set in $B$. Thus, the number of possible choices for sets in $B$ is at most $2^{kt-k|A|^{1/k}}$.

The disjointness matrix here is at least $(t^k, 2^{kt})$-rich, since every choice $Y$ allows for $t^k$ possible choices for $X$ that are disjoint. By Lemma 1.20, any protocol where Alice sends $a$ bits and Bob sends $b$ bits induces a 1-monochromatic rectangle with dimensions $t^k/2^a \times 2^{kt-a-b}$, so Claim 1.21 gives:
\[
2^{kt-a-b} \leq 2^{kt-kt/2^{a/k}}
\]
\[
\Rightarrow a + b \geq \frac{n}{2^{a/k+1}}.
\]
We conclude:

**Theorem 1.22.** If $X, Y \subseteq [n], |X| = k$ and Alice sends at most $a$ bits and Bob sends at most $b$ bits in a protocol computing $\text{Disj}(X,Y)$, then $a + b \geq \frac{n}{2^{a/k+1}}$.

For example, for $k = 2$, if Alice sends at most $\log n$ bits to Bob, then Bob must send at least $\Omega(\sqrt{n})$ bits to Alice in order to solve lopsided disjointness.

**Span** Suppose Alice is given a vector $x \in \{0,1\}^n$, and Bob is given a $n/2$ dimensional subspace $V \subseteq \{0,1\}^n$. Their goal is figure out whether or not $x \in V$. As in the case of disjointness, we start by claiming that the inputs do not have 1-monochromatic rectangles of a certain shape:

**Claim 1.23.** If $A \times B$ is a 1-monochromatic rectangle, then $|B| \leq 2^{n^2/2 - n \log |A|}$. 
Proof. The set of x’s in the rectangle spans a subspace of dimension at least \(\log |A|\). The number of \(n/2\) dimensional subspaces that contain this span is thus at most \(\binom{2^n}{n/2 - \log |A|} \leq 2^{n^2/2 - n \log |A|}\). \(\Box\)

The problem we are working with is at least \((2^{n/2}, 2^{n^2/4}/n!)-\) rich, since there are at least \(2^{n^2/4}/n!\) subspaces, and each contains \(2^{n/2}\) vectors. Applying Lemma 1.20 and Claim 1.23, we get that if there is a protocol where Alice sends \(a\) bits and Bob sends \(b\) bits, then

\[
2^{n^2/4 - a - b} \leq 2^{n^2/2 - n \log 2^{n^2/4 - a}}
\]

\[
\Rightarrow n^2/4 - a(n + 1) - n \log n \leq b.
\]

**Theorem 1.24.** If Alice sends \(a\) bits and Bob sends \(b\) bits to solve the span problem, then \(b \geq n^2/4 - a(n + 1) - n \log n\).

For example, if Alice sends at most \(n/8\) bits, then Bob must send at least \(\Omega(n^2)\) bits in order to solve the span problem. One of the players must send a linear number of the bits in their input.

**Fooling Sets**

A set \(S \subset X \times Y\) is called a fooling set if every monochromatic rectangle can share at most 1 element with \(S\). Fooling sets can be used to prove several basic lower bounds on communication.

**Greater-than** Our first example using fooling sets is the greater-than function, \(GT : [n] \times [n] \rightarrow \{0, 1\}\), defined as:

\[
GT(x, y) = \begin{cases} 
1 & \text{if } x > y, \\
0 & \text{otherwise.}
\end{cases}
\] (1.4)

The trivial protocol computing greater-than has complexity \(1 + \lceil \log n \rceil\) bits, and we shall show that this is essentially tight. The methods we used for the last two examples will surely not work here, because \(GT\) has large 0-monochromatic rectangles (like \(R = \{(x, y) : x < n/2, y > n/2\}\) and large 1-monochromatic rectangles (like \(R = \{(x, y) : x > n/2, y < n/2\}\)). Instead we shall use a fooling set to prove the bound. Consider the set of \(n\) points \(S = \{(x, x)\}\). We claim:

**Claim 1.25.** Two points of \(S\) cannot lie in the same monochromatic rectangle.

Indeed, if \(R\) is monochromatic, and \(x < x',\) but \((x, x), (x', x') \in R\), then since \(R\) is a rectangle, \((x', x) \in R\). This contradicts the

A subspace of dimension \(n/2\) can be specified by picking the \(n/2\) basis vectors. For each such vector, there are at least \(2^{n/2}\) available choices. However, we have over counted by a factor of \(n!\), since every permutation of the basis vectors gives the same subspace. This gives that there are at least \(2^{n^2/4}/n!\) subspaces.

![Figure 1.17: The greater than function has large monochromatic rectangles of both colors.](image)
We need at least \( n \) monochromatic rectangles to cover pairs of the type \((0, e_i)\), where \( e_i \) is the \( i \)’th unit vector. So once again, we have shown that the number of monochromatic rectangles must be at least \( n \), proving:

**Theorem 1.26.** The deterministic communication complexity of \( GT \) is at least \( \log n \).

**Disjointness** Fooling sets also allow us to prove tighter lower bounds on the communication complexity of disjointness. Consider the set \( S = \{(X, \overline{X}) : X \subseteq [n]\} \), namely the set of pairs of sets and their complements. No monochromatic rectangle can contain two such pairs, because if such a rectangle contained \((X, \overline{X}), (Y, \overline{Y})\) for \( X \neq Y \), then it would also contain both \((X, \overline{Y}), (Y, \overline{X})\), but at least one of the last pair of sets must intersect, while the first two pairs are disjoint. Since \( |S| = 2^n \), and at least one more 0-monochromatic rectangle is required, this proves:

**Theorem 1.27.** The deterministic communication complexity of disjointness is \( n + 1 \).

**Krapchenko’s Method**

We end the lower bounds part of this chapter with a method for non-boolean relations. Let \( \mathcal{X} = \{x \in \{0, 1\}^n : \sum_{i=1}^n x_i = 0 \mod 2\} \) and \( \mathcal{Y} = \{y \in \{0, 1\}^n : \sum_{i=1}^n y_i = 1 \mod 2\} \). Since \( \mathcal{X} \) and \( \mathcal{Y} \) are disjoint, for every \( x \in \mathcal{X}, y \in \mathcal{Y} \), there is an index \( i \) such that \( x_i \neq y_i \). Suppose Alice is given \( x \) and Bob is given \( y \), and they want to find such an index \( i \). How much communication is required?

Perhaps the most trivial protocol is for Alice to send Bob her entire string, but we can use binary search to do better. Notice that

\[
\sum_{i \leq n/2} x_i + \sum_{i > n/2} x_i \neq \sum_{i \leq n/2} y_i + \sum_{i > n/2} y_i \mod 2.
\]

Alice and Bob can thus exchange \( \sum_{i \leq n/2} x_i \mod 2 \) and \( \sum_{i \leq n/2} y_i \mod 2 \). If these values are not the same, they can safely restrict their attention to the strings \( x_{\leq n/2}, y_{\leq n/2} \) and continue. On the other hand, if the values are the same, they can continue the protocol on the strings \( x_{> n/2}, y_{> n/2} \). In this way, in every step they communicate 2 bits and eliminate half of their input string, giving a protocol of communication complexity \( 2 \log n \).

It is easy to see that \( \log n \) bits of communication are necessary, because that’s how many bits it takes to write down the answer. Now we shall prove that \( 2 \log n \) bits are necessary, using a variant of fooling sets. Consider the set of inputs

\[
S = \{(x, y) \in \mathcal{X} \times \mathcal{Y} : x, y \text{ differ in only 1 coordinate}\}.
\]
S contains \( n \cdot 2^{n-1} \) inputs, since one can pick an input of \( S \) by picking \( x \in X \) and flipping any of the \( n \) coordinates. We will not be able to argue that every monochromatic rectangle must contain only one element of \( S \) or bound the number of elements in any way. Instead, we will prove that if such a rectangle does contain many elements of \( S \), then it is big:

**Claim 1.28.** Suppose \( R \) is a monochromatic rectangle that contains \( r \) elements of \( S \). Then \( |R| \geq r^2 \).

The key observation here is that two distinct elements \((x, y), (x, y') \in S\) cannot be in the same monochromatic rectangle. For if the rectangle was labeled \( i \), then \((x, y), (x, y')\) must disagree in the \( i\)’th coordinate, but since they both belong to \( S \) we must have \( y = y' \). Similarly we cannot have two distinct elements \((x, y), (x', y) \in S\) that belong to the same monochromatic rectangle. Thus, if \( R = A \times B \) has \( r \) elements of \( S \), we must have \( |A| \geq r, |B| \geq r \), proving that \( |R| \geq r^2 \).

Now suppose there are \( t \) monochromatic rectangles that partition the set \( S \), and the \( i\)’th rectangle covers \( r_i \) elements of \( S \). Then \( |S| = \sum_{i=1}^{t} r_i \), but since the rectangles are disjoint, \( 2^{2n-2} \geq \sum_{i=1}^{t} r_i^2 \). Using these facts and the Cauchy-Schwartz inequality:

\[
2^{2n-2} \geq \frac{1}{t} \sum_{i=1}^{t} r_i^2 \geq \left( \frac{1}{t} \sum_{i=1}^{t} r_i \right)^2 = n^2 2^{2n-2} / t,
\]

proving that \( t \geq n^2 \). This shows that the binary search protocol is the best one can do.

**Rectangle Covers**

Given that rectangles play such a crucial role in the communication complexity of protocols, it is worth studying alternative ways to use them to measure the complexity of functions. Here we investigate what one can say if we count the number of monochromatic rectangles needed to cover all of the inputs.

**Definition 1.29.** We say that a boolean function has a 1-cover of size \( C \) if there are \( C \) monochromatic rectangles whose union is all of the inputs that evaluate to 1. We say that the function has a 0-cover of size \( C \) if there are \( C \) monochromatic rectangles whose union is all of the inputs that evaluate to 0.

By Theorem 1.6, every function that admits a protocol with communication \( c \) also admits a 1-cover of size at most \( 2^c \) and a 0-cover of size at most \( 2^c \). Conversely, Theorem 1.9 shows that small covers can be used to give small communication.
Can the logarithm of the cover number be significantly different from the communication complexity? Consider the disjointness function, defined in (1.3). For $i = 1, 2, \ldots, n$, define the rectangle $R_i = \{(X, Y) : i \in X, i \in Y\}$. Then we see that $R_1, R_2, \ldots, R_n$ form a 0-cover for disjointness. So there is a 0-cover of size $n$, yet (Theorem 1.27) the communication complexity of disjointness is $n + 1$. In fact, by the proof of Theorem 1.9, this must mean that any 1-cover for disjointness must have at least $2^{\Omega(\sqrt{n})}$ rectangles.

Another interesting example is the $k$-disjointness function. Here Alice and Bob are given sets $X, Y \subseteq [n]$, each of size $k$. We shall see in Chapter 2 that the communication complexity of $k$-disjointness is at least $\log \binom{n}{k} \approx k \log(n/k)$. As above, there is a 0-cover of $k$-disjointness using $n$ rectangles.

**Claim 1.30.** $k$-disjointness has a 1-cover of size $2^{2k} \ln \binom{n}{k}^2$.

We prove Claim 1.30 using the probabilistic method. Sample a random 0-rectangle by picking a set $S \subseteq [n]$ uniformly at random, and using the rectangle $R = \{(X, Y) : X \subseteq S, Y \subseteq [n] \setminus S\}$. Namely, the set of all inputs $X, Y$ where $X$ is contained in $S$, and $Y$ is contained in the complement of $S$. Now sample $t = 2^{2k} \ln \binom{n}{k}^2$ such rectangles independently. The probability that a particular disjoint pair $(X, Y)$ is included in any single rectangle is $2^{-2k}$. So the probability that the pair is excluded from all the rectangles is

$$(1 - 2^{-2k})^t < e^{-2 \cdot 2^k t} \leq \left(\frac{n}{k}\right)^{-2},$$

by the choice of $t$. Since the number of disjoint pairs $(X, Y)$ is at most $\binom{n}{k}^2$, this means that the probability that any disjoint pair is excluded
by the $t$ rectangles is less than 1. So there must be $t$ rectangles that cover all the 1 inputs.

Setting $k = \log n$, we have found 1-cover with $t = O(n^2 \log^2 n)$ rectangles. This means that all the entries of the matrix can be covered with $2^{O(\log n)}$ monochromatic rectangles. However, we shall see in Chapter 2 that the communication complexity of $k$-disjointness is exactly $\log (\binom{n}{k}) = \Omega(\log^2 n)$. This example shows that Theorem 1.9 is tight, at least when it comes to covers.

Direct-sums in Communication Complexity

The direct-sum question is about the complexity of solving several copies of a given problem—if a function requires $c$ bits of communication, how much communication is required to compute $k$ copies of the function?

Given a function $g : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$, we define $g^k : (\{0,1\}^n)^k \times (\{0,1\}^n)^k \rightarrow \{0,1\}^k$ by

$$g^k((x_1,\ldots,x_k),(y_1,\ldots,y_k)) = (g(x_1,y_1),g(x_2,y_2),\ldots,g(x_k,y_k)).$$

If the communication complexity of $g$ is $c$, then the communication complexity of $g^k$ is at most $kc$. Can it be lower? Nontrivial examples are known that show that one can find cheaper protocols for solving many copies of certain kinds of tasks, though no such examples are known for computing functions.

Suppose Alice is given a set $S \subseteq [n]$ of size $n/2$, with $n$ even. Bob has no input. The goal of the players is to output an element of $S$. Alice can send Bob the minimum element of her set. This can be done with communication $\lceil \log(n/2 + 1) \rceil$, since the elements $n/2 + 2,\ldots,n$ can never be the minimum of $S$. Moreover, $\lceil \log(n/2 + 1) \rceil$ bits are necessary. Indeed, if fewer bits are sent, then the set of elements $P$ that Bob could potentially output is of size at most $n/2$, and so the protocol would fail if Alice is given the complement of $P$ as input.

On the other hand, we show that the parties can solve $k$ copies of this problem with $k + \log(nk)$ bits of communication, while the naive protocol would take $k \log(n/2 + 1)$ bits of communication. The key claim is:

**Claim 1.31.** There is a set $Q \subseteq [n]^k$ of size $nk^2$ with the property that for any $S_1,S_2,\ldots,S_k$, each of size $n/2$, there is an element $q \in Q$ such that $q_i \in S_i$ for every $i = 1,2,\ldots,k$.

This claim gives the protocol—Alice simply sends Bob the name of the element of $Q$ with the required property.
Proof of claim. To find such a set $Q$, we pick $|Q|$ elements at random from $[n]^k$ uniformly at random. For any fixed $S_1, \ldots, S_k$, the property that $Q$ misses this tuple is

$$(1 - (1/2)^k)|Q| \leq e^{-(1/2)^k}|Q|.$$ 

Setting $|Q| = nk2^k$, the probability that $Q$ does not have an element that works for some tuple is at most

$$2^{nk}e^{-(1/2)^k} \leq 2^{nk}e^{nk} < 1,$$

so such a $Q$ does exist. \hfill \Box

Nevertheless, we shall use the concept of rectangle covers to prove\footnote{Feder et al., 1995}:

**Theorem 1.32.** If $g$ requires $c$ bits of communication, then $g^k$ requires at least $k(\sqrt{c} - \log n - 1)$ bits of communication.

In fact, one can show that even computing the two bits $\bigwedge_{i=1}^k g(x_i, y_i)$, and $\bigvee_{i=1}^k g(x_i, y_i)$ requires $k(\sqrt{c} - \log n - 1)$ bits of communication\footnote{See Exercise 1.12.}.

The heart of the proof is the following lemma:

**Lemma 1.33.** If $g^k$ can be computed with $\ell$ bits of communication, then the inputs to $g$ can be covered by $\lceil 2\ell \cdot 2^{\ell/k} \rceil$ monochromatic rectangles.

**Theorem 1.9** and **Lemma 1.33** imply that $g$ has a protocol with communication $\left(\ell/k + \log n + 1\right)^2$. Thus,

$$c \leq \left(\ell/k + \log n + 1\right)^2 \\
\Rightarrow \ell \geq k(\sqrt{c} - \log n - 1),$$

as required.

Now we turn to proving **Lemma 1.33**. We find rectangles that cover the inputs to $g$ iteratively. Let $S \subseteq \{0, 1\}^n \times \{0, 1\}^n$ denote the set of inputs to $g$ that have not yet been covered by one of the monochromatic rectangles we have already found. Initially, $S$ is the set of all inputs. We claim:

**Claim 1.34.** There is a rectangle that is monochromatic under $g$ and covers at least $2^{-\ell/k}|S|$ of the inputs from $S$.

**Proof.** Since $g^k$ can be computed with $\ell$ bits of communication, by **Theorem 1.6**, the set $S^k$ can be covered by $2^{\ell}$ monochromatic rectangles. So there must be some monochromatic rectangle $R$ that covers at least $2^{-\ell}|S|^k$ of these inputs. For each $i$, define

$$R_i = \{(x, y) \in \{0, 1\}^n \times \{0, 1\}^n : \exists (a, b) \in R, a_i = x, b_i = y\},$$
which is a rectangle, since $R$ is a rectangle. $R_i$ is simply the projection of the rectangle $R$ to the $i$'th coordinate. Moreover, since this rectangle is monochromatic under $g^k$, it must be monochromatic under $g$. Since

$$\prod_{i=1}^k |R_i \cap S| \geq |R \cap S^k| \geq 2^{-\ell} |S|^k.$$ 

there must be some $i$ for which $|R_i \cap S| \geq 2^{-\ell/k} |S|$.

We repeatedly pick rectangles using Claim 1.34 until all of the inputs to $g$ are covered. After $\lceil 2n2^{\ell/k} \rceil$ steps, the number of uncovered inputs is at most

$$2^{2n} \cdot (1 - 2^{-\ell/k})^{2n2^{\ell/k}} = 2^{2n} \cdot e^{-2^{\ell/k}} < 1.$$ 

Using $1 - x \leq e^{-x}$ for all $x$.

**Exercise 1.1**

Show that if $g : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ is such that $g^{-1}(1)$ can be partitioned into $2^c$ rectangles, then $g$ has communication complexity at most $O(c^2)$.

**Exercise 1.2**

Suppose Alice and Bob each get a subset of size $k$ of $[n]$, and want to know whether these sets intersect or not. Show that at least $\log(n/k)$ bits are required.

**Exercise 1.3**

Suppose Alice gets a string $x \in \{0, 1\}^n$ which has more 0's than 1's, and Bob gets a string $y \in \{0, 1\}^n$ that has more 1's than 0's. They wish to communicate to find a coordinate $i$ where $x_i \neq y_i$. Show that at least $2 \log n$ bits of communication are required.

**Exercise 1.4**

Show that almost all functions $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ require communication $\Omega(n)$.

**Exercise 1.5**

Let $X$ and $Y$ be families of subsets of $[n]$. Assume for all $x \in X$ and $y \in Y$ the intersection of $x$ and $y$ contains at most 1 element, that is, $|x \cap y| \leq 1$. Define the communication problem as follows. Alice receives $x \in X$, Bob receives $y \in Y$, and they wish to evaluate the function $f : X \times Y \rightarrow \{0, 1\}$ defined as $f(x, y) = |x \cap y|$. Show the deterministic complexity of $f$ is $O(\log^2(n))$.

**Exercise 1.6**
Recall that for a simple undirected graph $G$, the chromatic number $\chi(G)$ is the minimum number of colors needed to color the vertices of $G$ so that no two adjacent vertices have the same color. Show that $\log \chi(G)$ is at most the deterministic communication complexity of $G$’s adjacency matrix.

**Exercise 1.7**

Alice and Bob receive inputs $x, y \in \mathbb{Z}^n$, where $x_i, y_i \geq 0$ for all $i$. They want to compute an index $m \in [n]$ minimizing

$$\left| \sum_{j < i} x_j + y_j - \sum_{j > i} x_j + y_j \right|.$$ 

Exhibit a deterministic protocol with $O(\log n)$ bits of communication for finding $m$ as above. Show no protocol can do asymptotically better. What happens when the union and median are as sets?

**Exercise 1.8**

Show that for every $0 \leq \alpha < 1$, any deterministic protocol for estimating the Hamming distance between two strings $x, y \in \{0, 1\}^n$ up to an additive error $\alpha n$ must have length at least $\Omega(n)$.

**Exercise 1.9**

Consider the partial function $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$, where the input to each party is interpreted as two $n/2$ bit strings, defined by

$$f(x, x', y, y') = \begin{cases} 1 & \text{if } x = y \text{ and } x' \neq y', \\ 0 & \text{if } x \neq y \text{ and } x' = y'. \end{cases}$$

Show that there are $2n$ monochromatic rectangles under $f$. Use fooling sets to show that the communication complexity of $f$ is at least $\Omega(n)$. This proves that an analogue of Theorem 1.9 does not hold for partial functions.

**Exercise 1.10**

For a boolean function $g$, define $g^{\wedge k}$ by $g(x_1, x_2, \ldots, x_k, y_1, \ldots, y_k) = \bigwedge_{i=1}^k g(x_i, y_i)$. Show that if $g^{\wedge k}$ has a 1-cover of size $2^\ell$, then $g$ has a 1-cover of size $2^{\ell/k}$.

**Exercise 1.11**

In this exercise, we will show\(^\dagger\) that an optimal direct sum theorem does not hold for the deterministic communication complexity of relations. Consider the problem where Alice is given a subset $X \subseteq [n]$
of size $t$, and Bob is given no input. The players want to output an element of $X$.

1. Show that $\log(n - t + 1)$ bits of communication are required for any deterministic protocol (and also sufficient).

2. Show that if Alice is given $k$ sets $X_1, \ldots, X_k$, each of size $t$, and the parties want to compute an element from each of the sets, then there is a deterministic protocol that communicates only

$$O(k \log(n/t) + \log(kn))$$

bits, which is significantly less than $k \log(n - t + 1)$, when $t = n/2$.

**Exercise 1.12**

Show that if $g : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ requires $c$ bits of communication, then any protocol computing both $\bigwedge_{i=1}^k g(x_i, y_i)$ and $\bigvee_{i=1}^k g(x_i, y_i)$ requires $k(\sqrt{c/2} - \log n - 1)$ bits of communication. Hint: Find a small 1-cover using the protocol for computing $\bigvee_{i=1}^k g(x_i, y_i)$, and 0-cover using the protocol for computing $\bigwedge_{i=1}^k g(x_i, y_i)$. 

Hint: Pick a random subset of $[n]^k$ of size $(n/t)^k$ in $\binom{[n]}{t}$, and argue that it intersects $X_1 \times \cdots \times X_k$ with positive probability.
Matrices give a powerful way to represent mathematical objects. Once an object is encoded as a matrix, the many tools of linear algebra can be used to understand properties of the object. This broad approach has been very fruitful in many areas of mathematics, and we shall use it to understand communication complexity as well.

We can represent a function \( g : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\} \) by an \( m \times n \) matrix \( M \), where \( m = |\mathcal{X}| \) is the number of rows and \( n = |\mathcal{Y}| \) is the number of columns, and the \((i, j)\)'th entry is \( M_{ij} = g(i, j) \). If we interpret the inputs to the parties as unit column vectors \( e_i, e_j \), we have

\[
g(i, j) = e_i^T M e_j.
\]

The rank of a matrix is the maximum size of a set of linearly independent rows in the matrix. We write \( \text{rank}(M) \) to denote the rank of a matrix \( M \). The driving question of this chapter is:

How does the relate to communication complexity?

Rank has many other useful interpretations, that may seem different at first:

**Fact 2.1.** For an \( m \times n \) matrix \( M \), the following are equivalent:

- \( \text{rank}(M) = r. \)
- \( r \) is the smallest number such that \( M \) can be expressed as \( M = AB \), where \( A \) is an \( m \times r \) matrix, and \( B \) is an \( r \times n \) matrix.
- \( r \) is the smallest number such that \( M \) can be expressed as the sum of \( r \) matrices of rank 1.
- \( r \) is the largest number such that \( M \) has \( r \) linearly independent columns.

The second characterizations of rank discussed above suggests that the rank of a matrix is closely related to communication complexity:
Alice and Bob can use a factorization $M = AB$, where $A$ is an $m \times r$ matrix and $B$ is an $r \times n$, to get a protocol for computing $g$. To compute $g(i,j) = e_i^T M e_j = e_i^T A B e_j$, Alice can send Bob $e_i^T A$, and then Bob can multiply this vector with $B e_j$. This involves transmitting a vector of at most $r$ numbers. This is not quite an efficient protocol, because each of the numbers in the vector may require lots of bits to encode. To give a rigorous bound, we need to make a few more observations about the rank.

**Some Properties**

The rank of a matrix has some very nice properties that make it easy to work with.

**Fact 2.2.** If a matrix $A$ is obtained from $B$ by permuting rows or columns then $\text{rank}(A) = \text{rank}(B)$.

$A$ is called a submatrix of $B$ if $A$ can be obtained by deleting some rows and columns of $B$.

**Fact 2.3.** If $A$ is a submatrix of $B$ then $\text{rank}(A) \leq \text{rank}(B)$.

**Fact 2.4.** $\text{rank}\left(\begin{bmatrix} A & C \\ 0 & B \end{bmatrix}\right) \geq \text{rank}(A) + \text{rank}(B)$.

**Fact 2.5.** $|\text{rank}(A) - \text{rank}(B)| \leq \text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$.

A consequence of Fact 2.5 is that the constants used to represent a function $g(i,j)$ do not change the rank in a big way. For example, if $M$ is a matrix with 0/1 entries, one can define a matrix $M'$ of the same dimensions by

$$M'_{i,j} = (-1)^{M_{i,j}}.$$  

This operation replaces 1’s with −1’s and 0’s with 1’s. Now observe that $M' = J - 2M$, where $J$ is the all 1’s matrix, and so

**Fact 2.6.** $|\text{rank}(M') - \text{rank}(M)| \leq \text{rank}(J) = 1$.

The tensor product of an $m \times n$ matrix $M$ and an $m' \times n'$ matrix $M'$ is the $mm' \times nn'$ matrix $T = M \otimes M'$ whose entries are indexed by tuples $(i,i')$, $(j,j')$ defined by

$$T_{(i,i'),(j,j')} = M_{i,j} \cdot M'_{i',j'}.$$  

The tensor product multiplies the rank, a fact that is very useful for proving lower bounds.

**Fact 2.7.** $\text{rank}(M \otimes M') = \text{rank}(M) \cdot \text{rank}(M')$.
The matrices we are working with have 0/1 entries, so one can view their entries as real numbers, rationals, elements of the field $\mathbb{F}_2$ of size 2. Each of those interpretations gives a different definition of rank. However, we have:

**Lemma 2.8.** If $M$ has 0/1 entries, then the rank of $M$ over the reals is equal to its rank over the rationals, which is at least as large as its rank over $\mathbb{F}_2$.

**Proof.** The proof of the first equality follows from Gaussian elimination: If the rank over the rationals is $r$, we can always apply an invertible linear transformation to the rows using rational coefficients to bring the matrix into this form:

$$M = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & M_{1,r+1} & \ldots & M_{1,n} \\ 0 & 1 & 0 & \ldots & 0 & M_{2,r+1} & \ldots & M_{2,n} \\ 0 & 0 & 1 & \ldots & 0 & M_{3,r+1} & \ldots & M_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & M_{r,r+1} & \ldots & M_{r,n} \\ 0 & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

This transformation does not affect the rank over the reals, and now it is clear that the rank over the reals is exactly $r$.

To prove that the rank over the rationals is at least as large as the rank over $\mathbb{F}_2$, observe that if any set of rows is linearly dependent over the rationals, then we can find an integer linear dependence between them, and so get a linear dependence over $\mathbb{F}_2$. So if $r$ rows are linearly independent over $\mathbb{F}_2$, they must also be linearly independent over the rationals and the reals.

One consequence of Lemma 2.8 is:

**Lemma 2.9.** A matrix with 0/1 entries of rank $r$ has at most $2^r$ distinct rows, and at most $2^r$ distinct columns.

**Proof.** Since the rank over $\mathbb{F}_2$ is also at most $r$, every row must be expressible as the linear combination of some $r$ rows over $\mathbb{F}_2$. There are only $2^r$ such linear combinations possible, so there can be at most $2^r$ distinct rows.

**Lower Bounds on Communication using Rank**

The communication complexity of $M$ cannot exceed its rank:

**Theorem 2.10.** If a matrix has rank $r$, then its communication complexity is at most $r + 1$. 

Proof. If the matrix has rank \( r \), then by Lemma 2.9, it has at most \( 2^r \) distinct rows. So Alice need only announce which of these distinct rows corresponds to her input. This takes \( r \) bits of communication. Bob can then respond with the value of the function. 

The rank of a matrix also gives a lower bound on its communication complexity, via the following lemma:

Lemma 2.11. If the 1's of a 0/1 matrix \( M \) can be partitioned into \( 2^c \) monochromatic rectangles, then its rank is at most \( 2^c \).

Proof. For every rectangle \( R = A \times B \), let \( M_R \) denote the 0/1 matrix whose \((i,j)\) entry is 1 iff \((i,j) \in R\). We have \( \text{rank}(M_R) = 1 \). Moreover, \( M \) can be expressed as the sum of \( 2^c \) such matrices. By Fact 2.1, this means \( \text{rank}(M) \leq 2^c \).

Since every function with low communication gives rise to a partition into monochromatic rectangles (Theorem 1.6), we immediately get:

Theorem 2.12. If a matrix has rank \( r > 0 \), then its communication complexity is at least \( \log r \).

Theorem 2.12 allows us to prove lower bounds on the communication complexity of many of the examples we have already considered. Let us revisit some of them.

Equality We start with the equality function, defined in (1.1). The matrix of the equality function is just the identity matrix. The rank of the matrix is \( 2^n \), proving that the communication complexity of equality is \( n + 1 \).

Greater-than Consider the greater than function, defined in (1.4). The matrix of this function is the upper-triangular matrix which is 1 above the diagonal and 0 on all other points. Once again the matrix has full rank. This proves that the communication complexity is more than \( \log n \).

Disjointness Consider the disjointness function, defined in (1.3). Let \( D_n \) be 0/1 matrix that represents disjointness. Let us order the rows and columns of the matrix in lexicographic order so that the last rows/columns correspond to sets that contain \( n \). We see that \( D_n \) can be expressed as:

\[
D_n = \begin{bmatrix}
D_{n-1} & D_{n-1} \\
D_{n-1} & 0
\end{bmatrix}
\]

In other words \( D_n = D_1 \otimes D_{n-1} \), and so \( \text{rank}(D_n) = 2 \cdot \text{rank}(D_{n-1}) \) by Fact 2.7. We conclude that \( \text{rank}(D_n) = 2^n \), proving that the communication complexity of disjointness is \( n + 1 \).
**k-disjointness**  Consider the disjointness function restricted to sets of size at most $k$. In this case, the matrix is an $\sum_{i=0}^{k} \binom{n}{i} \times \sum_{i=0}^{k} \binom{n}{i}$ matrix. Let us write $D_{n,k}$ to represent the matrix for this problem. We shall use polynomials to prove that this matrix too has full rank.

For two sets $X, Y \subseteq [n]$ of size at most $k$, define the monomial $m_X(z_1, \ldots, z_n) = \prod_{i \in X} z_i$, and the string $z_Y \in \{0, 1\}^n$ such that $(z_Y)_i = 0$ if and only if $i \in Y$. This ensures that $\text{Disj}(X, Y) = m_X(z_Y)$. Any non-zero linear combination of the rows corresponds to a linear combination of the monomials we have defined, and so gives a non-zero polynomial $f$. We show that for any such polynomial $f$, there is a set $Y$ of size at most $k$ so that $f(z_Y) \neq 0$, so $f$ is non-zero. This proves that the rank of the matrix is full.

To show this, let $X$ be a set that corresponds to a monomial of maximum degree in $f$. Let us restrict the values of all variables outside $X$ to be equal to 1. This turns $f$ into a non-zero polynomial that only depends on the variables corresponding to $X$. In this polynomial, let $X'$ denote the set of variables in a minimal monomial that has a non-zero coefficient. Consider the assignment $z_Y$ for $Y = X \setminus X'$. Now, $f(z_Y)$ is equal to the coefficient of this minimal monomial, which is non-zero.

**Inner-product**  Our final example is the inner product function $\text{IP} : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ defined by

$$\text{IP}(x, y) = \sum_{i=1}^{n} x_i y_i \mod 2. \quad (2.1)$$

Again, it will be helpful to use Fact 2.6. If $P_n$ represents the matrix...
of IP after sorting the rows and columns lexicographically, and replacing 1 with −1 and 0 with −1, we see that

\[ P_n = \begin{bmatrix} P_{n-1} & P_{n-1} \\ P_{n-1} & -P_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes P_{n-1}, \]

and so by Fact 2.7, \( \text{rank}(P_n) = 2 \cdot \text{rank}(P_{n-1}) \). This proves that \( \text{rank}(P_n) = 2^n \), and so by Fact 2.6 the communication complexity of IP is at least \( n \).

**Upper bounds on Communication From Rank**

Lovasz and Saks conjectured\(^1\) that Theorem 2.12 is closer to the truth than Theorem 2.10:

**Conjecture 2.13.** There is a constant \( \alpha \) such that the communication complexity of a non-constant matrix \( M \) is at most \( \log^\alpha \text{rank}(M) \).

We do know of examples that requires \( \alpha \geq 2 \) in such an inequality\(^2\), so we cannot expect the communication complexity of a matrix to be proportional to the logarithm of its rank.

Our main goal in this section is to prove the following theorem\(^3\):

**Theorem 2.14.** If the rank of a matrix is \( r > 1 \), then its communication complexity is at most \( O(\sqrt{r} \log^2 r) \).

The proof of Theorem 2.14 that we present relies on a beautiful theorem from convex geometry called John’s theorem\(^4\), that we discuss below. We use John’s theorem to prove the following lemma\(^5\), which applies to any low rank matrix with 0/1 entries.

**Lemma 2.15.** Any \( m \times n \) matrix that has 0/1 entries and rank \( r \geq 0 \) must contain a monochromatic submatrix of size at least \( mn \cdot 2^{-20\sqrt{r} \log r} \).

Before proving the lemma, let us see\(^6\) how to use it to get a protocol (see Figure 2.2 for an outline).

**Proof of Theorem 2.14.** Assume \( r > 9 \), since otherwise the proof is complete by Theorem 2.10. Let \( R \) be the rectangle promised by Lemma 2.15. Rearranging the rows and columns, we can write the matrix as:

\[ M = \begin{bmatrix} R & A \\ B & C \end{bmatrix}. \]

Since the rank of \( R \) is at most 1, Fact 2.5 and Fact 2.4 can be used to show:

\[
\text{rank} \left( \begin{bmatrix} R \\ B \end{bmatrix} \right) + \text{rank} \left( \begin{bmatrix} R & A \end{bmatrix} \right) \leq \text{rank} \left( \begin{bmatrix} R & A \\ B & C \end{bmatrix} \right) + 3 \quad (2.2)
\]
This is because:
\[
\begin{align*}
\text{rank } & \left( \begin{bmatrix} R \\ B \end{bmatrix} \right) + \text{rank } \left( \begin{bmatrix} R & A \\ B & C \end{bmatrix} \right) \\
\leq & \text{rank } (A) + \text{rank } (B) + 2 \\
\leq & \text{rank } \left( \begin{bmatrix} 0 & A \\ B & C \end{bmatrix} \right) + 2 \\
\leq & \text{rank } \left( \begin{bmatrix} R & A \\ B & C \end{bmatrix} \right) + \text{rank } \left( \begin{bmatrix} -R & 0 \\ 0 & 0 \end{bmatrix} \right) + 2 \\
\leq & \text{rank } \left( \begin{bmatrix} R & A \\ B & C \end{bmatrix} \right) + 3.
\end{align*}
\]

Now if
\[
\text{rank } \left( \begin{bmatrix} R \\ B \end{bmatrix} \right) \leq \text{rank } \left( \begin{bmatrix} R & A \end{bmatrix} \right),
\]
then Bob can tell Alice if his input is consistent with \( R \). If it is consistent, then the players can make a lot of progress by setting
\[
M = \begin{bmatrix} R \\ B \end{bmatrix},
\]
and continuing the protocol on the matrix \( M \). They have reduced the rank of the matrix by at least a factor of \( \frac{3}{2} \). If Bob’s input is not consistent with \( R \), then the players set
\[
M = \begin{bmatrix} A \\ C \end{bmatrix},
\]
and recursively continue the protocol on \( M \). In this case, they reduced the size of the matrix by at least a factor of \( 1 - 2^{-20 \sqrt{\log r}} \).

The length of the protocol described above may be large. However, we shall prove that it does have a small number of leaves. By Lemma 2.9, we can assume that the matrix \( M \) has at most \( 2^r \) rows and columns. The number of transmissions in this protocol where Alice or Bob says that their input is inconsistent with the monochromatic rectangle is at most \( 2r \ln 2 \cdot 2^{20 \sqrt{\log r}} \), since after that many transmissions, the number of entries in the matrix have been reduced to
\[
2^{2r} (1 - 2^{-20 \sqrt{\log r}}) 2r 2^{20 \sqrt{\log r}} < 2^{2r} e^{-2^{-20 \sqrt{\log r}} 2r \ln 2} 2^{20 \sqrt{\log r}} = 1.
\]

The number of transmissions where Alice or Bob announces that their input is consistent with the rectangle is at most \( O(\log r) \), since after that many transmissions, the rank of the matrix is reduced to
\[
\frac{7}{2} (t + 3)/2 \leq 21/3, \text{ when } t \geq 9
\]
less than 9. Thus, the number of leaves in this protocol is at most
\[
2 \ln 2 \cdot 2^{\sqrt{r} \log^2 r + O(\log r)} \leq 20(\sqrt{r} \log^2 r).
\]

Finally, by Theorem 1.7, we can balance the protocol tree to obtain
a protocol computing \( M \) with length \( O(\sqrt{r} \log^2 r) \).

It only remains to prove Lemma 2.15. It is no loss of generality to assume that the matrix has more 0’s than 1’s—if this is not the case, we can always work with the matrix \( J - M \). The lemma is proved in two steps. First we show that \( M \) must contain a large rectangle that is almost monochromatic:

**Claim 2.16.** If at least half of the entries in \( M \) are 0’s, then there is a submatrix \( T \) of \( M \) of size at least \( mn \cdot 2^{-16\sqrt{r} \log r} \) such that the fraction of 1’s in \( T \) is at most \( 1/r^3 \).

The second claim shows how to find a large zero rectangle in a matrix with low rank and few ones.

**Claim 2.17.** If \( T \) is 0/1 matrix of rank \( r \) so that at most \( 1/r^3 \) of the entries are 1’s, then there is a 0-submatrix consisting of at least half of the rows and half of the columns of \( T \).

**Proof.** Call a row of \( T \) good if the fraction of 1’s in it is at most \( 2/r^3 \). At least half the rows of \( T \) must be good, or else \( T \) would have more than \( 1/r^3 \) fraction of 1’s overall. Let \( T' \) be the submatrix obtained by restricting \( T' \) to the good rows. Since \( \text{rank}(T') \leq r \), it has \( r \) rows \( A_1, \ldots, A_r \) that span all the other rows of \( T' \). Each row \( A_i \) has at most \( 2/r^3 \) fraction of 1’s, so at most \( r \cdot 2/r^3 \leq 1/2 \) fraction of the columns can contain a 1 in one of these \( r \) rows. Let \( T'' \) be the submatrix obtained by restricting \( T' \) to the columns that do not have a 1 in the rows \( A_1, \ldots, A_r \). Then \( T'' \) must be 0, since every row of \( T' \) is a linear combination of \( A_1, \ldots, A_r \).

It only remains to prove Claim 2.16, which relies on some interesting results from convex geometry. A set \( K \subseteq \mathbb{R}^r \) is called convex if whenever \( x, y \in K \), then all the points on the line segment between \( x \) and \( y \) are also in \( K \). The set is called symmetric if whenever \( x \) is in \( K \), \( -x \) is also in \( K \). An ellipsoid centered at 0 is a set of the form:

\[
E = \left\{ x \in \mathbb{R}^r : \sum_{i=1}^{r} \langle x, u_i \rangle^2 / \alpha_i^2 \leq 1 \right\},
\]

where \( u_1, \ldots, u_r \) is an orthonormal basis of \( \mathbb{R}^r \), and \( \alpha_1, \ldots, \alpha_r \) are non-zero numbers.

**Theorem 2.18 (John’s Theorem).** Let \( K \subseteq \mathbb{R}^r \) be a symmetric convex body such that the unit ball is the most voluminous of all ellipsoids contained in \( K \). Then every element of \( K \) has Euclidean length at most \( \sqrt{r} \).

\(^8\) If we replace \( M \) with \( J - M \), where \( J \) is the all 1’s matrix, this can increase the rank by at most 1, but now the role of 0’s and 1’s has been reversed.

\(^9\) Gavinsky and Lovett, 2014
John’s theorem can be used to show that any matrix with 0/1 entries has a useful factorization:

**Lemma 2.19.** Any boolean matrix $M$ of rank $r$ can be expressed as $M = AB$, where $A$ is an $m \times r$ matrix whose rows are vectors of length at most $\sqrt{r}$, and $B$ is an $r \times n$ matrix whose columns are vectors of length at most 1.

**Proof.** Since the matrix has rank $r$, we know that $M$ can be expressed as $M = A'B'$, where $A'$ is an $m \times r$ matrix, and $B'$ is an $r \times n$ matrix. Moreover, the rows of $A'$ must all be linearly independent, and columns of $B'$ must all be linearly independent, or the rank of $M$ would be less than $r$. The matrices $A', B'$ do not necessarily satisfy the length constraints. Let $v'_1, \ldots, v'_m$ be the rows of $A'$, and $w'_1, \ldots, w'_n$ be the columns of $B'$.

Let $K'$ be the convex hull of $\{\pm v'_1, \ldots, \pm v'_m\}$. Our first goal is to modify these vectors so that the ellipsoid of maximum volume in $K'$ is the unit ball. Although this may not be the case initially, we show how to transform $A', B'$ to $A, B$ and consequently $K'$ to $K$ so that the ellipsoid of maximum volume contained in $K$ is the unit ball.

Suppose

$$E' = \left\{ x \in \mathbb{R}^r : \sum_{i=1}^{r} \langle x, u_i \rangle^2 / a_i^2 \leq 1 \right\}$$

is the ellipsoid of maximum volume contained in $K'$. We apply a linear transformation to the space of rows of $A'$ to get the rows of $A$. Each row $v'_i = \sum_{j=1}^{r} \beta_i u_j$ in $A'$ is replaced by the row $v_i = \sum_{j=1}^{r} \alpha_i \beta_i u_j$. This has the desired effect on the ellipsoid of the maximal volume, the ellipsoid of maximum volume in the convex hull $K$ of $\{\pm v_1, \ldots, \pm v_m\}$ is now

$$E = \left\{ x \in \mathbb{R}^r : \sum_{i=1}^{r} \langle x, u_i \rangle^2 \leq 1 \right\},$$

The "$\sqrt{r}$" and "1" in the statement of Lemma 2.19 can be replaced by any numbers whose product is $\sqrt{r}$. The lemma also holds for all matrices $M$ so that $|M_{i,j}| \leq 1$ for all $i, j$. 
which is the unit ball. To compensate for scaling the rows of $A'$, we also need to scale the columns of $B'$. Each column $w'_i = \sum_{j=1}^r \gamma_j u_i$ in $B'$ is replaced by the column $w_i = \sum_{j=1}^r (1/\alpha_i)\gamma_j u_i$. This ensures that:

$$\langle v_i, w_j \rangle = \sum_{k=1}^r \alpha_k \beta_k (1/\alpha_i)\gamma_k u_i = \sum_{k=1}^r \beta_k \gamma_k u_i = \langle v'_i, w'_j \rangle,$$

and so $M = A'B' = AB$. By John’s theorem, every vector $v_i$ must have length at most $\sqrt{r}$.

It only remains to argue that vectors $w_1, \ldots, w_n$ are of length at most 1. This is where we use the fact that the matrix has entries of small magnitude. Consider any $w_i$, and the unit vector in the same direction $e_i = w_i/\|w_i\|$. The length of $w_i$ can be expressed as $\langle w_i, e_i \rangle$. Since $e_i$ is in the unit ball, it is also contained in $K$, so $e_i = \sum j \mu_j v_j + \sum_j \kappa_j (-v_j)$ is a convex combination of the $v_j$'s.

Thus

$$\langle w_i, e_i \rangle = \left\langle w_i, \sum_j \mu_j v_j \right\rangle + \left\langle w_i, -\sum_j \kappa_j v_j \right\rangle = \sum_j \mu_j \langle w_i, v_j \rangle + \sum_j \kappa_j \langle w_i, -v_j \rangle \leq \sum_j \mu_j + \sum_j \kappa_j = 1,$$

where the inequality follows from the fact that $M$ has 0/1 entries. \qed

Now let us use Lemma 2.19 to complete the proof.

**Proof of Claim 2.16.** Let $A, B$ be the two matrices guaranteed by Lemma 2.19. Let $v_1, \ldots, v_m$ be the rows of $A$ and $w_1, \ldots, w_n$ be the columns of $B$. Let $\theta_{ij} = \arccos \left( \frac{\langle v_i, w_j \rangle}{\|v_i\| \|w_j\|} \right)$ be the angle between the unit vectors in the directions of $v_i$ and $w_j$. We claim that

$$\theta_{ij} \begin{cases} \frac{\pi}{2} & \text{if } M_{ij} = 0, \\ \frac{\pi}{2} - \frac{2\pi}{r\sqrt{r}} & \text{if } M_{ij} = 1. \end{cases}$$

When $v_i, w_j$ are orthogonal, the angle is $\pi/2$. When the inner product is closer to 1, we use the fact that $\arccos(\alpha) \leq \pi/2 - 2\pi\alpha/7$, which implies that the angle is at most $\arccos \left( \frac{1}{\sqrt{r}} \right) \leq \frac{\pi}{2} - \frac{2\pi}{r\sqrt{r}}$.

The existence of the rectangle we seek is proved via the probabilistic method. Consider the following experiment. Sample $t$ vectors $z_1, \ldots, z_t \in \mathbb{R}^r$ of length 1 uniformly and independently at random, and use them to define the rectangle

$$R = \{ (i, j) : \forall k \in [t], \langle v_i, z_k \rangle > 0, \langle w_j, z_k \rangle < 0 \}.$$

For fixed $(i, j)$ and $k$, the probability that $\langle v_i, z_k \rangle > 0$ and $\langle w_j, z_k \rangle < 0$
0 is exactly $1 - \frac{\pi/2 - \theta_{ij}}{2\pi}$. So for a fixed $(i, j)$ we get

$$
\Pr_{z_1, \ldots, z_n}[(i, j) \in R] = \begin{cases} 
\left(\frac{1}{4}\right)^t & \text{if } M_{i,j} = 0, \\
\left(\frac{1}{4} - \frac{1}{7\sqrt{r}}\right)^t & \text{if } M_{i,j} = 1.
\end{cases}
$$

Let $R_1$ denote the number of 1’s in $R$ and $R_0$ denote the number of 0’s. Set $t = \lceil7\sqrt{r}\log r \rceil$. By what we have just argued,

$$
E[R_0] \geq \frac{mn}{2 \cdot 4^t},
$$

and using the Fact that $1 - x \leq e^{-x}$ for $0 < x < 1$,

$$
E[R_1] \leq \frac{mn}{2 \cdot 4^t} \cdot \left(1 - \frac{4}{7\sqrt{r}}\right)^t \leq \frac{mn}{2 \cdot 4^t} \cdot e^{-\frac{4}{7\sqrt{r} \cdot t}} \leq \frac{mn}{2 \cdot 4^t} \cdot r^{-4 \log r}.
$$

Now let $Q = R_0 - r^4R_1$. By linearity of expectation, we have

$$
E[Q] \geq \frac{mn}{2 \cdot 4^t} \cdot (1 - 1/r) \geq mn \cdot 2^{-16\sqrt{r} \log r}.
$$

There must be some rectangle $R$ realizing this value of $Q$. Only a $1/r^3$ fraction of such a rectangle can correspond to 1 entries of the matrix, or else $Q$ would be negative.

**Open Problem 2.20.** Find a more direct/geometric argument for proving Lemma 2.15 or Theorem 2.14.

**Non-negative Rank**

Another way to measure the complexity of a matrix is by measuring its non-negative rank. The non-negative rank of an $m \times n$ boolean matrix $M$ is the smallest number $r$ such that $M = AB$, where $A, B$ are matrices with non-negative entries, such that $A$ is an $m \times r$ matrix and $B$ is an $r \times n$ matrix. Equivalently, it is the smallest number of non-negative rank 1 matrices that sum to $M$. Clearly, we have:

**Fact 2.21.** $\text{rank}(M) \leq \text{rank}_+(M)$.

However, $\text{rank}(M)$ and $\text{rank}_+(M)$ may be quite different. For example, given a set of numbers $X = \{x_1, \ldots, x_n\}$ of size $n$, consider the $n \times n$ matrix defined by $M_{i,j} = (x_i - x_j)^2 = x_i^2 + x_j^2 - 2x_ix_j$. Since $M$ is the sum of three rank 1 matrices, $\text{rank}(M) \leq 3$. On the other hand, we can show by induction on $n$ that $\text{rank}_+(M) \geq \log n$. Indeed, if $\text{rank}_+(M) = r$, then there must be non-negative rank 1 matrices $R_1, \ldots, R_r$ such that $M = R_1 + \ldots + R_r$. Let the support of $R_1$ be the rectangle $A \times B \subset X \times X$. Then we must have that either $|A| \leq n/2$ or $|B| \leq n/2$, or else there will be an element $x \in A \cap B$.

Another example is the inner-product matrix indexed by binary strings $x, y \in \{0, 1\}^n$ where $M_{xy} = \langle x, y \rangle$. By definition, $\text{rank}(M) \leq n$, but we shall see in Chapter 6 that $\text{rank}_+(M) \geq 2^{\Omega(n)}$. 
but $M_{x,x} = 0$. Suppose without loss of generality that $|A| \leq n/2$. Let $M'$ be the submatrix that corresponds to the numbers of $X \setminus A$. Then, by induction,

$$r - 1 \geq \text{rank}_+(M') \geq \log(n/2) = \log(n) - 1.$$ 

The non-negative rank could give stronger lower bounds on communication complexity than the rank:

**Theorem 2.22.** If a matrix has non-negative rank $r > 1$, then its communication complexity is greater than $\log r$.

When the non-negative rank is small, the matrix has polylogarithmic communication complexity$^{10}$:

**Theorem 2.23.** The communication complexity of $M$ is at most

$$O(\log(\text{rank}_+(M)) \cdot \log(\text{rank}(M))) \leq O(\log^2(\text{rank}_+(M))).$$

**Proof.** If a matrix $M$ with 0/1 entries has small non-negative rank $r$, we must have $M = R_1 + \ldots + R_r$, where $R_1, \ldots, R_r$ are non-negative rank 1 matrices. The set of non-zero entries of each matrix $R_i$ must form a monochromatic rectangle in $M$ with value 1. Thus, $M$ must admit a 1-cover of size $r$. We shall use this 1-cover to define an efficient communication protocol.

The protocol is similar to the one used to prove Theorem 1.9. If the rank of the matrix is $r < 9$, the parties use a constant number of bits to compute the output. If $r > 9$, then for every rectangle $R$ in the 1-cover, we can write

$$M = \begin{bmatrix} R & A \\ B & C \end{bmatrix}.$$ 

As in 2.2, either

$$\text{rank} \left( \begin{bmatrix} R & A \end{bmatrix} \right) \leq (\text{rank}(M) + 3)/2,$$

(2.3)

or

$$\text{rank} \left( \begin{bmatrix} R \\ B \end{bmatrix} \right) \leq (\text{rank}(M) + 3)/2.$$ 

(2.4)

So in each step of the protocol, if Alice sees an $R$ that is consistent with her input and satisfying (2.3), she announces its name, or if Bob sees a rectangle $R$ in the cover that is consistent with his input and satisfying (2.4), he announces its name.

Both parties then restrict their attention to the appropriate submatrix, which reduces the rank of $M$ by a factor of $2/3$. This can continue for at most $O(\log \text{rank}(M))$ steps before the rank of the matrix becomes less than 9.
On the other hand, if neither party finds such an \( R \), then there is no such \( R \) that covers their input, so they can safely conclude that the entry they seek is 0.

**Exercise 2.1**

Show that there is a boolean matrix of rank \( r \) with \( 2^r \) distinct rows, and \( 2^r \) distinct columns. Conclude that Lemma 2.9 is sharp.

**Exercise 2.2**

Let \( M \) be a 0/1 matrix with exactly \( t \) ones in each row and column. Show that you can cover the zeros of \( M \) using \( O(t \log |X| + \log |Y|) \) monochromatic rectangles.

**Exercise 2.3**

Show that the protocol in Figure 2.2 goes through even if we weaken Lemma 2.15 to only guarantee a rectangle with rank at most \( r/8 \) (instead of rank at most one, or monochromatic).

**Exercise 2.4**

For any symmetric matrix \( M \in \{0, 1\}^{n \times n} \) with ones in all diagonal entries, show that

\[
2^c \geq \frac{n^2}{|M|'}
\]

where \( c \) is the deterministic communication complexity of \( M \), and \( |M| \) is the number of ones in \( M \).

**Exercise 2.5**

For any boolean matrix \( M \), define \( \text{rank}_2(M) \) to be the rank of \( M \) over the field with two elements \( \mathbb{F}_2 \). Exhibit an explicit family of matrices \( M \in \{0, 1\}^{n \times n} \) with the property that \( c \geq \text{rank}_2(M)/10 \), where \( c \) is the deterministic communication complexity of \( M \). This falsifies the analogue of log-rank conjecture for \( \text{rank}_2 \).

**Exercise 2.6**

Show that if \( f \) has fooling set of size \( s \) then \( \text{rk}(M_f) \geq \sqrt{s} \). *Hint: tensor product.*

**Exercise 2.7**

For a real matrix \( M \) and \( \epsilon > 0 \) define the \( \epsilon \)-approximate rank of \( M \) to be \( \text{rank}_\epsilon(M) = \min\{ \text{rank}(A) : |A_{i,j} - M_{i,j}| \leq \epsilon \text{ for all } i, j \} \).

1. Find a boolean matrix with rank \( r \) and 1/3-approximate rank at most \( O(\log r) \).
2. Prove the following strengthening of Theorem 2.14: The communication complexity of a matrix $M$ is at most $O(\sqrt{\text{rank}_{1/3}(M)} \log^{2} \text{rank}(M))$. 
Access to randomness is an enabling feature in many computational processes. Randomized algorithms are often easier to understand and more elegant than their deterministic counterparts. In this chapter, we show how randomization can be used to give protocols that are far more efficient than the best deterministic protocols for many problems.

We start by giving some examples of protocols where the use of randomness gives an advantage that cannot be matched by deterministic protocols. Later, we give rigorous definitions for randomized protocols, and prove some basic facts about them. We do not discuss any lower bounds on randomized communication complexity in this chapter. Lower bounds for randomized protocols can be found in Chapter 5 and Chapter 6.

Some Examples of Randomized Protocols

Equality Suppose Alice and Bob are given access to $n$ bit strings $x, y$, and want to know if these strings are the same or not (1.1). In Chapter 1 we showed that the deterministic communication complexity of this problem is $n + 1$.

There is a simple randomized protocol, where Alice and Bob use a shared random string. Alice and Bob use the shared randomness to sample a random function $h : \{0, 1\}^n \rightarrow \{0, 1\}^k$. Then, Alice sends $h(x)$ to Bob, and Bob responds with a bit indicating whether or not $h(y) = h(x)$. If $h(x) = h(y)$, they conclude that $x = y$.

If $h(x) \neq h(y)$, they conclude that $x \neq y$. The number of bits communicated is $k + 1$. The probability of making an error is at most $2^{-k}$—if $x = y$ then $h(x) = h(y)$, and if $x \neq y$ then the probability that $h(x) = h(y)$ is at most $2^{-k}$.

This protocol may seem less than satisfactory, because the number...
of shared random bits required is quite large. We can reduce the number of random bits used if we use an error correcting code. This is a function $C : \{0, 1\}^n \to [2^k]^n$, such that if $x \neq y$, then $C(x)$ and $C(y)$ differ in all but $m/2^{-\Omega(k)}$ coordinates. It can be shown that if $m$ is set to $10n$, then for any $k$, most functions $C$ will be error correcting codes.

Given the code, Alice can pick a random coordinate of $C(x)$ and send it to Bob, who can then check whether this coordinate is consistent with $C(y)$. This takes $\log m + \log 2^k = O(\log n + k)$ bits of communication, and again the probability of making an error is at most $2^{-\Omega(k)}$. In this protocol, the players do not require a shared random string, since the choice of $C$ is made once and for all when the protocol is constructed, and before the inputs are seen.

**Greater-than** Suppose Alice and Bob are given numbers $x, y \in [n]$ and want to know which one is greater $(1, 4)$. We have seen that any deterministic protocol for this problem requires $\log n$ bits of communication. However, there is a randomized protocol that requires only $O(\log \log n)$ bits of communication.

Here we describe a protocol that requires only $O(\log \log n \cdot \log \log n)$ communication. The inputs $x, y$ can be encoded by $\ell$-bit binary strings, where $\ell = O(\log n)$. To determine whether $x \geq y$, it is enough to find the most significant bit where $x$ and $y$ differ. To find this bit, we use the randomized protocol for equality described above, along with binary search. In the first step, Alice and Bob will use the protocol for equality to exchange $k$ bits that determine whether the $n^2$ first (most significant) bits of $x$ and $y$ are the same. If they are the same, the parties continue with the last $n/2$ bits. If not, the parties discard the second half of their strings. In this way, after $O(\log n)$ steps, they find the first bit of difference in their inputs. In order to ensure that the probability of making an error in all of these $O(\log n)$ steps is small, we set $k = O(\log \log n)$. By the union bound, this guarantees that the protocol succeeds with high probability.

**k-Disjointness** Suppose Alice and Bob are given 2 sets $X, Y \subseteq [n]$ of size at most $k$, and want to know if these sets intersect or not. In Chapter 2, we used the rank method to argue that at least $\log \binom{n}{k} \approx k \log(n/k)$ bits of communication are required. Here we give a randomized protocol\(^\dagger\) that requires only $O(k)$ bits of communication, which is more efficient when $k \ll n$.

Alice and Bob start by exchange 2 bits to announce whether or not either of them has the empty set as an input. If one of them has the empty set as input, then their sets are disjoint, and the protocol terminates. If neither of their sets is empty, they use the protocol uses $k2^n$ shared random bits.

Many beautiful explicit constructions of error correcting codes are also known.

**Input:** Alice knows $x \in \{0, 1\}^n$.
**Output:** Whether or not $x = y$.

Alice picks a coordinate $i \in [m]$ uniformly at random.
Alice sends Bob $i, C(x)_i$;
Bob announces whether $C(x)_i = C(y)_i$;

Figure 3.2: A protocol for equality using private randomness and an error correcting code $C$.

The protocol requiring $O(\log \log n)$ bits of communication is described in Exercise ??.

**Input:** Alice knows $x \in \{0, 1\}^f$.
**Output:** Largest $i$ such that $x_i \neq y_i$, if such an $i$ exists.

Let $f = \lfloor n/2 \rfloor$.
while $|J| > 1$ do
let $J'$ be the first $|J|/2$ elements of $J$;
both parties use shared randomness to sample a random function $h : \{0, 1\}^{|J'|} \to \{0, 1\}^{2\log \log n}$;
Alice sends $h(x_J)$, which is $h$ evaluated on the bits in $J'$; Bob announces whether or not $h(x_J) = h(y_J)$;
if $h(x_J) = h(y_J)$ then
	Alice and Bob replace $J = J \setminus J'$;
else
	Alice and Bob replace $J = J'$;
end
both parties announce $x_J, y_J$ to decide if which of $x, y$ is greater;\(^\dagger\)

Figure 3.3: Public-coin protocol for greater than.

\(^\dagger\)Håstad and Wigderson, 2007

In Chapter 6, we show that $\Omega(k)$ bits are required.
shared randomness to sample a sequence of sets $R_1, R_2, \ldots \subseteq [n]$, uniformly at random. Alice announces the index $i$ of the first set $R_i$ that contains her set, and Bob announces the index $j$ of the first set $R_j$ that contains his set. This can be done with at most $2(\log(i) + \log(j) + 2)$ bits of communication. Now Alice can safely replace her set with $X \cap R_i$, and Bob can replace his set with $Y \cap R_j$—if the sets were disjoint, they remain disjoint, and if they were not disjoint, they must still intersect. They repeat the above process.

We argue that if the sets are disjoint, this process must end, at least in expectation, after $O(k)$ bits of communication. Suppose $X, Y$ are disjoint. Let us start by analyzing the expected number of bits that are communicated in the first step. We claim:

**Claim 3.1.** $\mathbb{E}[i] = 2^{|X|}, \mathbb{E}[j] = 2^{|Y|}$.

**Proof.** The probability that the first set of the sequence contains $X$ is exactly $2^{-|X|}$. In the event that it does not contain $X$, we are picking the first set that contains $X$ from the rest of the sequence. Thus:

$$\mathbb{E}[i] = 2^{-|X|} \cdot 1 + (1 - 2^{-|X|}) \cdot (\mathbb{E}[i] + 1)$$

$$\Rightarrow \mathbb{E}[i] = 2^{|X|}.$$  

The bound on the expected value of $j$ is the same.

So, by Jensen’s inequality applied to the log function, the expected length of the first step is at most

$$2 \mathbb{E} \left[ \log(i) + \log(j) \right] \leq 2 (\log(\mathbb{E}[i]) + \log(\mathbb{E}[j])) = 2(|X| + |Y|).$$ (3.1)

**Claim 3.2.** If $X \cap Y = \emptyset$, the expected number of bits communicated by the protocol is at most $8|X| + 8|Y| + 4$.

**Proof.** We show that for every non-negative integer $L$, if $(X, Y)$ are sets in $[n]$ so that $X \cap Y = \emptyset$ and $|X| + |Y| \leq L$ then the expected

\begin{verbatim}
Input: Alice knows $X \subseteq [n]$, Bob knows $Y \subseteq [n]$.
Output: Whether or not $X \cap Y = \emptyset$.
while $|X| > 1$ and $|Y| > 1$ and at most $120k + 20$ bits have been communicated so far do
    Alice and Bob use shared randomness to sample random subsets $R_1, R_2, \ldots \subseteq [n]$;
    Alice sends Bob the smallest $i$ such that $X \subseteq R_i$;
    Bob sends Alice the smallest $j$ such that $Y \subseteq R_j$;
    Alice replaces $X = X \cap R_i$;
    Bob replaces $Y = Y \cap R_j$;
end
if $X = \emptyset$ or $Y = \emptyset$ then
    Alice and Bob conclude that the sets were disjoint;
else
    Alice and Bob conclude that the sets were intersecting;
end
\end{verbatim}

Figure 3.5: Public-coin protocol for $k$-disjointness.
number of bits communicated is at most $C(L) \leq 8L + 4$. Similarly, the bound clearly holds when $L = 0$.

The proof proceeds by induction on $L$. For the base case, if $|X| + |Y| \leq L = 1$ then one of the sets is empty and indeed at most $2 \leq 8L + 4$ bits are communicated.

For the inductive step, assume $L \geq 2$. Equation (3.1) shows that the expected number of bits communicated in the first step of the protocol is at most $4 + 2|X| + 2|Y| \leq 4 + 2L$. The two new sets are $X \cap R_i$ and $Y \cap R_i$. By induction, the expected communication can be bounded:

$$C(L) = \sum_{L' = 0}^{|X| + |Y| - 1} \Pr[|X \cap R_i| + |Y \cap R_i| = L'] \cdot C(L') 
\leq \frac{C(L)}{2^{-L}} + \sum_{L' = 0}^{|X| + |Y| - 1} \Pr[|X \cap R_i| + |Y \cap R_i| = L'] \cdot (8L' + 4) $$

since $\Pr[|X \cap R_i| + |Y \cap R_i| = L] = 2^{-L}$.

$$\leq \frac{C(L)}{2^{-L}} + \mathbb{E}[8(|X \cap R_i| + |Y \cap R_i|) + 4].$$

We have $\mathbb{E}[|X \cap R_i|] = |X|/2$ and $\mathbb{E}[|Y \cap R_i|] = |Y|/2$, so $\mathbb{E}[|X \cap R_i| + |Y \cap R_i|] = L/2$. Thus, we have

$$C(L) \leq \frac{1}{1 - 2^{-L}} \cdot (4L + 4) \leq \frac{4}{3} \cdot (4L + 4) \leq 8L + 4,$$

for $L \geq 2$. \hfill $\Box$

Claim 3.2 means that if $X, Y$ are disjoint, the expected communication is at most $8|X| + 8|Y| + 4$. By Markov’s inequality, if the sets are disjoint then the probability that the protocol communicates more than $10 \cdot (8|X| + 8|Y| + 4)$ bits is at most $1/10$. Thus, if we run this process until $160k + 40$ bits have been communicated (or it terminates because one of the sets is empty), the probability of making an error is at most $1/10$.

Gap-Hamming Suppose Alice and Bob are given two strings $x, y \in \{+1, -1\}^n$, and want to estimate the Hamming-distance:

$$\Delta(x, y) = |\{i \in [n] : x_i \neq y_i\}| = \frac{n - (x, y)}{2}.$$

We say that a protocol $\pi$ approximates the Hamming distance up to a parameter $m$, if $|\pi(x, y) - \Delta(x, y)| \leq m$. In Exercise 1.8, we studied the relationship between $m$ and the deterministic communication complexity of $\pi$, and showed that for any $\alpha < 1$, approximating the Hamming distance up to $\alpha n$ requires communication $\Omega(n)$. Here we show that there is a significantly better randomized protocol.
Alice and Bob use shared randomness to sample \(i_1, \ldots, i_k \in [n]\) uniformly at random, and then communicate \(2k\) bits to compute 
\[\gamma = \frac{1}{k} \cdot \left| \{j \in [k] : x_{i_j} \neq y_{i_j}\} \right|\]. They output \(\gamma n\).

We now analyze the probability that this protocol makes an error, when \(\Delta(x, y) \leq n/2\). Define \(Z_1, \ldots, Z_k\) by
\[Z_j = \begin{cases} 
1 & \text{if } x_{i_j} \neq y_{i_j}, \\
0 & \text{otherwise}.
\end{cases}\]

The expected value of each \(Z_j\) is \(\Delta(x, y)/n\). So if \(m \leq \Delta(x, y)\), we can apply the Chernoff-Hoeffding bound to conclude:
\[
\Pr[|\pi(x, y) - \Delta(x, y)| > m] = \Pr[|\gamma k - \Delta(x, y)k/n| > mk/n] 
\leq e^{-\left(\frac{m}{n\Delta(x, y)}\right)^2 \cdot \frac{\Delta(x, y)k}{n}} 
\leq e^{-\frac{mk^2}{n\Delta(x, y)n}}.
\]

If \(m > \Delta(x, y)\), we have
\[
\Pr[|\pi(x, y) - \Delta(x, y)| > m] = \Pr[\gamma k > \Delta(x, y)k/n + mk/n] 
\leq e^{-\frac{m}{\Delta(x, y)n} \cdot \frac{\Delta(x, y)k}{n}} 
\leq e^{-\frac{mk}{n\Delta(x, y)n}}.
\]

Since \(m \leq n\) and \(\Delta(x, y) \leq n\), in either case this probability is at most \(e^{-\frac{mk^2}{3n^2}}\). Thus, if, for example, we set \(k = 3n^2/m^2\), we obtain a protocol whose probability of making an error is at most \(1/e\).

**Variants of Randomized Communication Complexity**

A **randomized protocol** is a deterministic protocol where each party has access to a random string, in addition to the inputs to the protocol. The random string is sampled independently from the inputs, but may have an arbitrary distribution known to the players.

We say that a protocol uses **public coins** if all parties have access to a common shared random string. We say that the protocol uses **private coins** if each party privately samples an independent random string. Every private coin protocol can be simulated by a public coin protocol, and we shall soon see a partial converse: every public coin protocol can be simulated with private coins, with a small increase in the communication.

There are two established ways to measure the probability that a randomized protocol makes an error:

For \(m = n^{0.6}\) we get a protocol of length \(O(n^{0.4})\), and for \(m = \sqrt{n}/\epsilon\) the length is \(O(\epsilon^2 n)\).

One can always simulate any randomized protocol by a protocol that uses uniformly random bits as the shared randomness.

If a randomized protocol never makes an error, we can fix the randomness to obtain a deterministic protocol that is always correct.
Worst-case We say that a randomized protocol has error $\epsilon$ in the worst-case if the probability that the protocol makes an error is at most $\epsilon$ on every input.

Average-case Given a distribution on inputs $\mu$, we say that a protocol has error $\epsilon$ with respect to $\mu$ if the probability that the protocol makes an error is at most $\epsilon$ when the inputs are sampled from $\mu$.

In both cases, the length of the protocol is always defined to be the maximum depth of all of the deterministic protocol trees that the protocol may generate.

When a protocol has error $\epsilon < 1/2$ in the worst case, we can run it several times and output the most common outcome. This reduces the probability of making an error. If we run the protocol $k$ times, and output the most frequent output in all of the runs, there will be an error in the output only if at least $k/2$ of the runs computed the wrong answer. By the Chernoff bound, the probability of error is thus reduced to at most $2^{-\Omega(k(1/2-\epsilon)^2)}$.

The worst-case and average-case complexity of a problem are related by via Yao’s minimax principle:

**Theorem 3.3.** The communication complexity of a function $g$ in the worst-case with error at most $\epsilon$ is equal to the maximum, over all distributions $\mu$, of the average-case communication complexity of $g$ with error at most $\epsilon$ with respect to $\mu$.

To prove Theorem 3.3, we appeal to a famous minimax principle due to von Neumann:

**Theorem 3.4.** Let $M$ be an $m \times n$ matrix with entries that are real numbers. Let $A$ denote the set of $1 \times m$ row vectors with non-negative entries, such that $\sum_i x_i = 1$, and let $B$ denote the set of $n \times 1$ column vectors with non-negative entries such that $\sum_j y_j = 1$. Then

$$\min_{x \in A} \max_{y \in B} x^T M y = \max_{y \in B} \min_{x \in A} x^T M y.$$  

Theorem 3.4 has a very intuitive interpretation in terms of zero-sum games. The matrix $M$ encodes the rules of the game, and $x, y$ represent strategies for playing the game. There are two players: a row player and a column player. The row player privately chooses a row $i$ of the matrix, and the column player privately chooses a column $j$. The outcome of the game is determined by the choices of $i$ and $j$: the column player gets a payoff of $M_{ij}$, and the row player gets a payoff of $-M_{ij}$. So, the column player wishes to maximize $M_{ij}$, and the row player wishes to minimize $M_{ij}$.

The vector $x$ in the theorem corresponds to a distribution on the rows that the row player may use to choose his row, and $y$ corresponds to a distribution that the column player may use. Then
\[ \min_{x \in A} \max_{y \in B} xMy \] gives the expected value of the payoff, if the row announces his choice for \( x \) first and commit to it, before the column player picks \( y \). In this case, the row player will pick \( x \) to minimize \( \max_{y \in B} xMy \), and the column player will pick \( y \) to maximize \( xMy \). Similarly, the quantity \( \max_{y \geq 0} \min_{x \geq 0} xMy \) measures the expected payoff if the column player commits to a strategy \( y \) first, and the row player \( x \) gets to pick the best strategy \( x \) after seeing \( y \). The first quantity can only be larger than the second—the column player wishes to maximize the value, and in the first case he has more information available to help make his choice. However, the theorem states that they are equal. In other words, there is a strategy \( y^* \) for the column player that guarantees a payoff that is equal to the amount he would get if he knew the strategy of the row player.

Now we leverage this powerful theorem to prove Yao’s minmax principle:

**Proof of Theorem 3.3.** If there is a protocol that computes \( g \) with error \( \epsilon \) in the worst case, then the same protocol must compute \( g \) with error \( \epsilon \) in the average case, no matter what the input distribution is. So, the average-case complexity is at most the worst-case complexity.

Conversely, suppose that for every distribution on inputs, the average-case complexity of the problem with error \( \epsilon \) is at most \( c \). Consider the matrix \( M \), where every row corresponds to an input to the protocol, and every column corresponds to a deterministic communication protocol of length at most \( c \), defined by

\[
M_{i,j} = \begin{cases} 
1 & \text{if protocol } j \text{ computes } g \text{ correctly on input } i, \\
0 & \text{otherwise.}
\end{cases}
\]

A distribution on the inputs corresponds to a choice of \( x \). Since a randomized protocol can be thought of as a distribution on deterministic protocols, a randomized protocol corresponds to a choice of \( y \). The success probability of a fixed randomized protocol \( y \) on inputs distributed according to \( x \) is exactly \( xMy \). So, by assumption, we know that \( \min_x \max_y xMy \geq 1 - \epsilon \). Theorem 3.4 implies that \( \max_y \min_x xMy \geq 1 - \epsilon \) as well—there is a fixed randomized protocol \( y^* \) that has error at most \( \epsilon \) under every distribution on inputs. \( \square \)

**Public Coins vs Private Coins**

Every private coin protocol can certainly be simulated by a public coin protocol, by making the private randomness visible to both parties. It turns out that every private coin protocol can also
It is known that computing whether or not two $n$-bit strings are equal requires $\Omega(\log n)$ bits of communication if only private coins are used. This shows that Theorem 3.5 is tight.

**Theorem 3.5.** If $g : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$ can be computed with $c$ bits of communication, and error $\epsilon$ in the worst case, then it can be computed by a private coin protocol with $c + \log(n/\epsilon^2) + O(1)$ bits of communication, and error $2\epsilon$ in the worst case.

**Proof.** We use the probabilistic method to find the required private coin protocol. Suppose the public coin protocol uses a random string $r$ as the source of all randomness, drawn from some distribution $\mu$. To design the private coin protocol, we start by picking $t$ independent random strings $r_1, \ldots, r_t$ sampled from $\mu$.

For any fixed input $(x, y)$, some of these $t$ random strings lead to the public coin protocol computing the right answer, and some of the lead to the protocol computing the wrong answer. However, the probability that $r_i$ gives the right answer is at least $1 - \epsilon$. Thus, by the Chernoff bound, the probability that $1 - 2\epsilon$ fraction of the $t$ strings lead to the wrong answer is at most $2^{-\Omega(\epsilon^2 t)}$. We set $t = O(2n/\epsilon^2)$ to be large enough so that this probability is less than $2^{-2n}$. Then by the union bound, we get that the probability that more than $2\epsilon t$ of these strings give the wrong answer for any input is less than 1. Thus, there must be some fixed strings with this property.

The private coin protocol is now simple. We fix $r_1, \ldots, r_t$ with the property that for any input $(x, y)$, the fraction of strings giving the wrong answer is at most $2\epsilon$. Alice samples a uniformly random element $i \in \{1, 2, \ldots, t\}$, and sends $i$ to Bob, which takes at most $\log(n/\epsilon^2) + O(1)$ bits. Alice and Bob then run the original protocol using the randomness $r_i$. \qed

**Nearly Monochromatic Rectangles**

Monochromatic rectangles proved to be a very useful concept for understanding deterministic protocols. A similar role is played by nearly monochromatic rectangles when trying to understand randomized protocols.

Given a randomized protocol, and a distribution on inputs $\mu$, one can always fix the randomness of the protocol in the way that minimizes the probability of error under $\mu$. The result is a deterministic protocol whose error under $\mu$ is at most the error of the randomized protocol.

The following theorem describes some aspects of the connection between randomized protocols and nearly monochromatic rectangles. This is studied in greater detail in future chapters, where lower
bounds on randomized communication complexity are proved.

**Theorem 3.6.** If there is a deterministic $c$-bit protocol $\pi$ with error at most $\epsilon$ under a distribution $\mu$, and a set $S$ such that

$$\Pr[\pi(X, Y) \in S] > 2\sqrt{\epsilon},$$

then there exists a rectangle $R$ such that

- $\pi$ has the same outcome for all inputs in $R$, and this outcome is in $S$.
- $\Pr[(X, Y) \in R] \geq \sqrt{\epsilon} \cdot 2^{-c}$.
- $\Pr[\pi \text{ makes an error} | (X, Y) \in R] \leq \sqrt{\epsilon}$.

**Proof.** By Theorem 1.6, we know that the protocol induces a partition of the space into $t \leq 2^c$ rectangles $R_1, R_2, \ldots, R_t$, and in each of these rectangles, the outcome of the protocol is determined.

For each rectangle $R_i$ in the collection, define the number

$$e(R_i) = \Pr[\text{the protocol makes an error} | (X, Y) \in R_i].$$

Let $\rho(R_i)$ denote the number $\Pr[(X, Y) \in R_i]$. If $R$ denotes the rectangle that the inputs $X, Y$ belong to, we have that $E[e(R)] \leq \epsilon$, so Markov’s inequality gives $\Pr[e(R) > \sqrt{\epsilon}] \leq \sqrt{\epsilon}$. Thus,

$$E[1/\rho(R)] = \sum_{i=1}^{t} \Pr[R = R_i] \cdot \frac{1}{\Pr[(X, Y) \in R_i]} = t,$$

so by Markov’s inequality, we get $\Pr[1/\rho(R) > t/\sqrt{\epsilon}] \leq \sqrt{\epsilon}$.

Since

$$\Pr[\pi(X, Y) \in S] > 2\sqrt{\epsilon} > \Pr[e(R) > \sqrt{\epsilon}] + \Pr[1/\rho(R) > t/\sqrt{\epsilon}],$$

there must be a rectangle $R^*$ in the collection corresponding to an outcome in $S$, with $e(R^*) \leq \sqrt{\epsilon}$ and $\rho(R^*) \geq \sqrt{\epsilon}/t \geq \sqrt{\epsilon} \cdot 2^{-c}$, as required. \qed
When there are only $k = 2$ parties, this model is identical to the model of 2 party communication. In fact, optimal lower bounds in this model would have very interesting consequences to the study of circuit complexity.

A protocol solving this problem would compute both the disjointness function and the inner product function.

In Chapter 5, we prove that at least $n/4^k$ bits of communication are required.

Some Examples of Protocols in the Number-On-Forehead Model

We start with some examples of clever number-on-forehead protocols.

**Equality** We have seen that every deterministic protocol for computing equality in the two party setting must have complexity $n + 1$. Perhaps surprisingly, the complexity of equality is quite different in the number-on-forehead model. Suppose 3 parties each have an $n$ bit string written on their foreheads. Then there is a very efficient protocol for computing whether all three strings are the same: Alice announces whether or not Bob and Charlie’s strings are the same, and Bob announces whether or not Alice and Charlie’s strings are the same. This computes equality with 2 bits of communication.

**Intersection size** Suppose there are $k$ parties, and the $i$’th party has a subset $X_i \subseteq [n]$ on their forehead. The parties want to compute the size of the intersection $\bigcap_i X_i$. We shall describe a protocol that requires only $O(k^4(1 + n/2^k))$ bits of communication.
We start by describing a protocol that requires only \( O(k^2 \log n) \) bits of communication, as long as \( n < \binom{k}{k/2} \). It is helpful to think of the input as a \( k \times n \) boolean matrix. Each of the parties knows all but one row of this matrix, and they wish to compute the number of all 1’s columns. Let \( C_{i,j} \) denote the number of columns containing \( j \) ones that are visible to the \( i \)’th party. The parties compute and announce the values of \( C_{i,j} \), for each \( i, j \). Since this involves each player announcing \( k+1 \) numbers, the communication complexity of the protocol is at most \( (k+1)k \log n \leq O(k^2 \log n) \).

The following claim allows us to prove that the protocol achieves its goal.

**Claim 4.1.** Let \( A_j \) denote the actual number of columns with \( j \) ones in them. If there are two valid solutions \( A_k, \ldots , A_0 \) and \( A'_k, \ldots , A'_0 \) that are both consistent with the values \( C_{i,j} \), then either \( A'_k = A_k \), or for each \( j \),

\[
|A_j - A'_j| \geq \binom{k}{j}.
\]

**Proof.** Suppose \( A'_k \neq A_k \). Then \( |A_k - A'_k| \geq 1 = \binom{k}{k/2} \). We prove the claim by induction on \( j = k, k - 1, k - 2, \ldots , 0 \). Since a column of weight \( j \) is observed as having weight \( j - 1 \) by \( j \) parties, and having weight \( j \) by \( k - j \) parties, we have:

\[
(k - j)A_j + (j + 1)A_{j+1} = \sum_{i=1}^{k} C_{i,j} = (k - j)A'_j + (j + 1)A'_{j+1}
\]

\[
\Rightarrow |A_j - A'_j| \geq \frac{j+1}{k-j} |A_{j+1} - A'_{j+1}|
\]

\[
\geq \frac{j+1}{k-j} \binom{k}{j+1} = \binom{k}{j},
\]

as required. \( \square \)

Claim 4.1 shows that knowing \( C_{i,j} \) for all \( i, j \) allows one to determine the number of all 1’s columns. Indeed, if \( n < \binom{k}{k/2} \), then there can only be one possible value for \( A_k \), since otherwise \( |A'_k - A_k| > n \), which is not possible.

To obtain a protocol for general \( n \), the parties divide the columns of the matrix into blocks of size \( \binom{k}{k/2} - 1 \), and count the number of all 1’s columns in each block using the above idea separately. The total communication is then at most

\[
\left( \frac{n}{\binom{k}{k/2}} + 1 \right) \cdot k^2 \log \left( \frac{k}{k/2} \right) \leq O(k^4(1 + n/2^k)),
\]

as claimed.

**Exactly n** Suppose 3 parties each have a number from \([n]\) written on their forehead, and want to know whether these numbers...
sum to \( n \) or not. A trivial protocol is for one of the parties to announce one of the numbers she sees, and then the relevant party announces the answer, which takes \( \log n \) bits. Here we use ideas of Behrend\(^3\) to show that one can do it with just \( O(\sqrt{\log n}) \) bits of communication. Behrend’s ideas lead to a coloring of the integers that avoids monochromatic three-term arithmetic progressions:

**Theorem 4.2.** One can color the set \([m]\) with \(2^{O(\sqrt{\log m})}\) colors, such that for any \(a, b \in [m]\), if the numbers \(a, a + b, a + 2b\) are all in \([m]\), then they do not have the same color.

First we explain how the coloring from Theorem 4.2 can be used to get a protocol for the exactly \( n \) problem with communication \( O(\sqrt{\log n}) \). Suppose the three inputs are \(x, y, z\). Alice computes the number \(x' = n - y - z\), and Bob computes \(y' = n - x - z\).

If \(x + y + z = n\), then we have \(x = x'\) and \(y = y'\). In general, \(x - x' = y - y' = x + y + z - n\). Alice announces the color of \(x' + 2y\) in the coloring promised by Theorem 4.2, when applied to the set \([3n]\). Bob and Charlie just send a bit to indicate whether the announced color is the same as the color of \(x + 2y'\) or \(x + 2y\).

If all three colors are the same, the players conclude that the sum of their numbers is \( n \). If the colors are not the same, they conclude that the sum is not \( n \). The communication complexity of the protocol is at most \( O(\sqrt{\log n}) \).

The reason the protocol works is that if \(x + y + z \neq n\), then \(x - x' = y - y' \neq 0\), and

\[
x + 2y, \quad x + 2y + x' - x = x' + 2y, \quad x' + 2y + 2(y' - y) = x + 2y'
\]

form an arithmetic progression. Thus, by the property of the coloring, all three colors cannot be the same. On the other hand, if \(x + y + z = n\), then \(x' = x\) and \(y' = y\), and all three colors must be the same.

Now we turn to proving Theorem 4.2. A triple of points \(a, a + b, a + 2b\) can be also thought of as a triple of the form \(x, (x + y)/2, y\). In other words, we want to find a coloring of \([m]\) so that if \(x, y\) are of the same color, then \((x + y)/2\) has a different color. The basic observation is that the points on a sphere satisfy a similar property—if \(x, y\) are two vectors that lie on the same sphere, then \((x + y)/2\) must be shorter, and so cannot lie on the same sphere. The idea for the proof is to discretize this property of Euclidean length.

**Proof of Theorem 4.2.** We shall choose parameters \(d, r\) with \(d^r > m\)
and $d$ is divisible by 4. To carry out the above intuition, we need to convert each number $x \in [m]$ into a vector. To do this, we write each number $x \in [m]$ in base $d$, using at most $r$ digits. We express $x = \sum_{i=0}^{r-1} x_i d^i$, where $x_i \in [d-1]$. One can thus interpret $x \in [m]$ as a vector $v(x) \in \mathbb{R}^r$ whose $i$'th coordinate $x_i$. We approximate each of these vectors using a vector where every coordinate is off by at most $d/4$: let $w(x)$ be the vector where the $i$'th coordinate is the largest number of the form $jd/4$ such that $jd/4 \leq x_i$ and $j$ is an integer.

Color each number $x \in [n]$ by the vector $w(x)$ and the integer $\|v(x)\|^2 = \sum_{i=0}^{r-1} x_i^2$. The number of choices for $w(x)$ is at most $2^{O(r)}$, and the number of possible values for $\|v(x)\|^2$ is at most $O(rd^2)$, so the total number of possible colors is at most $2^{O(r+\log d)}$. Setting $r = \sqrt{\log m/\log 2}$ gives the required bound.

It only remains to check that the coloring avoids arithmetic progressions. For the sake of finding a contradiction, suppose $a, b \in [m]$ are such that $a, a + b, a + 2b$ all get the same color. Then we must have $\|v(a)\| = \|v(a + b)\| = \|v(a + 2b)\|$, so the three vectors $v(a), v(a + b), v(a + 2b)$ all lie on a sphere. We will get a contradiction by proving that $v(a + b) = \frac{v(a) + v(a + 2b)}{2}$.

To prove this, we need to use the fact that the points also satisfy: $w(a) = w(a + b) = w(a + 2b)$. We get a contradiction by proving that the following quantity is 0:

$$v(a + 2b) + v(a) - 2v(a + b)$$

$$= v(a + 2b) + v(a) - 2v(a + b) - w(a + 2b) - w(a) + 2w(a + b)$$

$$= (v(a + 2b) - w(a + 2b)) + (v(a) - w(a)) - 2(v(a + b) - w(a + b)).$$

Let $W(x) = \sum_{i=0}^{r-1} w(x_i) d^i$. Then we see that for any $x$, $v(x) - w(x)$ is just the base $d$ representation of $x - W(x)$, and this number is at most $d/4$ in each coordinate. This means that $(v(a + 2b) - w(a + 2b)) + (v(a) - w(a)) - 2(v(a + b) - w(a + b))$ is simply the base $d$ representation of $a + 2b - W(w(a + 2b)) + a - W(w(a)) - 2(a + b - W(w(a + b))) = 0$. So, it must be 0, as required.

\[\square\]

### Cylinder Intersections

The basic building blocks of protocols in the number-on-forehead model are cylinder intersections. They play the same role that rectangles play in the case that the number of parties is 2. Any set $S \subseteq X_1 \times \cdots \times X_k$ can be described using its characteristic func-
We can then define cylinder intersections as:

**Definition 4.3.** $S \subseteq \mathcal{X}_1 \times \cdots \times \mathcal{X}_k$ is called a cylinder if $\chi_S$ does not depend on one of its inputs. $S$ is called a cylinder intersection if it can be expressed as an intersection of cylinders.

If $S$ is a cylinder intersection, we can always express

$$\chi_S(x_1, \ldots, x_k) = \prod_{i=1}^{k} \chi_i(x_1, \ldots, x_k),$$

where $\chi_i$ is a boolean function that does not depend on the $i$'th input.

**Fact 4.4.** The intersection of two cylinder intersections is also a cylinder intersection.

Just as for rectangles, we say that a cylinder intersection is monochromatic with respect to a function $g$, if $g(x) = g(y)$ for every two inputs $x, y$ in the cylinder intersection. In analogy with the 2 party case, we have the following theorem:

**Theorem 4.5.** If the deterministic communication complexity of $g : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \rightarrow \{0, 1\}$ is $c$, then $\mathcal{X}_1 \times \cdots \times \mathcal{X}_k$ can be partitioned into at most $2^c$ monochromatic cylinder intersections with respect to $g$.

Indeed, for every outcome of the protocol $m$, it is easy to verify that the set of inputs that are consistent with that outcome form a cylinder intersection.
Lower Bounds from Ramsey Theory

Cylinder intersections are more complicated than rectangles, and this makes proving lower bounds in the number-on-forehead model challenging. Here we explore some lower bounds in this model by appealing to arguments from Ramsey Theory.

Let us consider the Exactly $n$ problem: Alice, Bob and Charlie are each given a number from $[n]$, written on their foreheads, and want to know if their numbers sum to $n$. We have shown that there is a protocol that computes this function using $O(\sqrt{\log n})$ bits of communication. Here we show that $\Omega(\log \log \log n)$ bits of communication are required.

Denote by $c_n$ the deterministic communication complexity of the exactly $n$ problem. To understand the behavior of $c_n$, we need to relate it to a combinatorial structure studied in Ramsey Theory. Three points in $[n] \times [n]$ form a corner if they are of the form $(x, y), (x + d, y), (x, y + d)$ for some integer $d$. A coloring of $[n] \times [n]$ with $C$ colors is a function $g : [n] \times [n] \to [C]$. We say that the coloring avoids monochromatic corners if there is no corner such that all three points get the same color. Let $C_n$ be the minimum number of colors required to avoid monochromatic corners in $[n] \times [n]$. We claim that $C_n$ essentially captures the value of $c_n$:

**Lemma 4.6.** $\log C_n/3 \leq c_n \leq 2 + \log C_n$.

**Proof.** To prove $\log C_n \geq c_n$, suppose there is a coloring with $C_n$ colors that avoids monochromatic corners. As in the protocol discussed at the beginning of this chapter, Alice can compute $x' = n - y - z$, and Bob can compute $y' = n - x - z$. Alice will then announce the color of $(x', y)$, and Bob and Charlie will say whether this color is the same
as the color of \((x, y')\) and \((x, y)\). The three points \((x, y'), (x', y), (x, z)\) form a corner, since \(x' - x = n - x - y - z = y' - y\). So all three points have the same color if and only if all three points are the same, which can only happen when \(x + y + z = n\).

To prove that \(\log C_{n/3} \le c_n\), suppose there is a protocol solving the exactly-\(n\) problem with \(c\) bits of communication. Then by Theorem 4.5, every input to the protocol can be colored by one of \(2^c\) colors that is the name of the corresponding cylinder intersection. This induces a coloring of \([n/3] \times [n/3]\): color \((x, y)\) by the name of the cylinder intersection containing the point \((x, y, n - x - y)\). We claim that this coloring avoid monochromatic corners. Indeed, if \((x, y), (x + d, y), (x, y + d)\) is a monochromatic corner with \(d \neq 0\), then \((x, y, n - x - y), (x + d, y, n - x - y - d), (x, y + d, n - x - y - d)\) must all belong to the same cylinder intersection. But then \((x, y, n - x - y - d)\) must also be in the same cylinder intersection, since it agrees with each of the three points in two coordinates. That contradicts the correctness of the protocol, since \(x + y + n - x - y = n\) but \(x + y + n - x - y - d \neq n\).

Next we prove:\footnote{Graham, 1980; and Graham et al., 1980}

**Theorem 4.7.** \(C_n \ge \Omega\left(\frac{\log \log n}{\log \log \log n}\right)\).

We prove the theorem by induction. However, to carry out the proof, we need to first strengthen the statement in order for the induction to go through. We prove that the matrix must either contain a monochromatic corner, or a structure called a rainbow-corner. A rainbow-corner with \(r\) colors and center \((x, y)\) is specified by a set of \(r\) distinct colors, and numbers \(d_1, \ldots, d_r\), such that \((x + d_i, y)\) and \((x, y + d_i)\) are both colored using the \(i\)th color, and \((x, y)\) is colored by the \(r\)th color.

**Proof of Theorem 4.7.** We shall prove by induction that as long as \(C > 3\), if \(n \ge 2^{2r}\), then any coloring of \([n] \times [n]\) with \(C\) colors must contain either a monochromatic corner, or a rainbow-corner with \(r\) colors. When \(r = C + 1\), this means that if \(n \ge 2^{2(C+1)}\), then \([n] \times [n]\) must contain a monochromatic corner, proving that \(C_n \ge \Omega\left(\frac{\log \log n}{\log \log \log n}\right)\).

For the base case, when \(r = 2\), since \(n \ge 2^{2r} > C\), two of the points of the type \((x, n - x)\) must have the same color. If \((x, n - x)\) and \((x', n - x')\) have the same color, with \(x > x'\), then \((x', n - x), (x, n - x), (x', n - x')\) are either a monochromatic corner, or a rainbow-corner with 2 colors.

For the inductive step, assume \(n = 2^{Cr}\). The set \([n]\) can be partitioned to \(m = 2^{Cr} - 2^{C(r-1)}\) intervals \(I_1, I_2, \ldots, I_m\), each of size exactly...
\[2^{C^2(r-1)}\]. By induction, each of the sets \(I_j \times I_j\) must have either a monochromatic corner, or a rainbow-corner with \(r - 1\) colors. If one of them has a monochromatic corner, we are done, so suppose they all have rainbow-corners with \(r - 1\) colors. Since a rainbow-corner is specified by choosing the center, choosing the colors and choosing the offsets for each color, there are at most
\[
\left(2^{C^2(r-1)}\right)^2 \cdot C^r \cdot \left(2^{C^2(r-1)}\right)^C
\]
potential rainbow-corners in each of these sets. Since the number of possible rainbow corners is less than \(m\), there must be some \(j < j'\) that have exactly the same rainbow corner with the same coloring. These two rainbow corners induce a monochromatic corner centered in the box \(I_j \times I_j\), or a rainbow-corner with \(r\) colors (see Figure 4.11).

**Exercise 4.1**

Define the generalized inner product function \(GIP\) as follows. For \(k\) inputs \(x_1, \ldots, x_k \in \{0, 1\}^n\),
\[
GIP(x) = \sum_{j=1}^{n} \prod_{i=1}^{k} x_{ij} \mod 2.
\]
This exercise outlines a number-on-forehead \(GIP\) protocol using \(O(n/2^k + k)\) bits. It will be convenient to think about the input \(X\) as a \(k \times n\) matrix with rows corresponding to \(x_1, \ldots, x_k\).

- Suppose the players know a string \(z \in \{0, 1\}^k\) with the property that no column of \(X\) is equal to \(z\). Show that the players can use \(z\) to compute the number of all 1’s columns as follows. Assume the first \(t\) coordinates of \(z\) are zeroes and the rest are ones. For \(\ell \in \{0, 1, \ldots, k-1\}\) define \(c_\ell\) as the number of columns in \(X\) with \(\ell\) zeroes, followed by either a one or zero, followed by \(k - \ell - 1\) ones. Prove that \(\sum_{\ell=0}^{t-1} c_\ell = GIP(x) \mod 2\). Use these ideas to find a protocol to compute \(GIP(x)\) using \(O(k)\) bits assuming the players know \(z\).

- Exhibit an overall protocol for \(GIP\) by showing that the players can agree upon a vector \(z\) and communicate to determine \(c_\ell \mod 2\) using \(O(n/2^k + k)\) bits.

**Exercise 4.2**
Given a function $g$, recall that we define $g^r$ to be the function that computes $r$ copies of $g$. This exercise explores, in the number-on-forehead model, what we can say about the communication required by $g^r$, knowing the communication complexity of $g$.

The approach taken in the proof of Theorem 1.32 does not work because cylinder intersections do not tensorize nicely like rectangles do. Fortunately, we can appeal to a powerful result from Ramsey theory called the Hales-Jewett Theorem\footnote{Hales and Jewett, 1963} to prove that the communication complexity of $g^r$ must increase as $r$ increases.

For an arbitrary finite set $S$, the Hales-Jewett theorem gives insight into the structure of the cartesian product $S^n = S \times S \times \cdots S$ as $n$ grows large. For the precise statement we need the notion of a combinatorial line. The combinatorial line specified by a nonempty set of indices $I \subseteq [n]$ and a vector $v \in S^n$ is the set of all $x \in S^n$ so that $x_i = v_i$ for every $i \notin I$ and $x_i = x_j$ for every $i, j \in I$. For example, when $S = [3]$ and $n = 4$ then the set $\{1132, 2232, 3332\}$ is a combinatorial line with $I = \{1, 2\}$ and $v_3 = 3, v_4 = 2$.

Given a set $S$ and a number $t$, the Hales-Jewett theorem says that as long as $n$ is large enough, any coloring of $S^n$ with $t$ colors must contain a monochromatic combinatorial line.

Assume that the communication complexity of $g$ is strictly greater than the number of players. Define $c_n$ to be communication required to compute the AND of $n$ copies of $g$. Prove that

$$\lim_{n \to \infty} c_n = \infty.$$

**Exercise 4.3**

A three player NOF puzzle demonstrates that unexpected efficiency is sometimes possible.

**Inputs:** Alice has a number $i \in [n]$ on her forehead, Bob has a number $j \in [n]$ on his forehead, and Charlie has a string $x \in \{0, 1\}^n$ on his forehead.

**Output:** On input $(i, j, x)$ the goal is for Charlie to output the bit $x_k$ where $k = i + j \mod n$.

**Question:** Find a deterministic protocol in which Bob sends one bit to Charlie, and Alice sends $\lfloor \frac{n}{2} \rfloor$ bits to Charlie; Alice and Bob must send Charlie their message simultaneously. Charlie is then able to output the correct answer.

**Exercise 4.4**

Show that any degree $d$ polynomial over $\mathbb{F}_2$ over the variables $x_1, \ldots, x_n$ can be computed by $d + 1$ players with $O(d)$ bits in the number-on-forehead model, for any partition of $x_1, \ldots, x_n$ to $d + 1$
parts, where each party has $n/(d+1)$ bits on their forehead (you may assume $d + 1$ divides $n$).
5 Discrepancy

The techniques we have developed for proving lower bounds in prior chapters all rely on the fact that efficient communication protocols lead to small partitions of the space into monochromatic sets. In this chapter, we shall prove that some functions cannot even be partitioned into large nearly monochromatic rectangles. The ideas we develop will lead to lower bounds for randomized protocols, and the best known lower bounds in the number-on-forehead model. To prove the stronger lower bounds, we start by developing techniques to bound the discrepancy of functions.

Discrepancy measures the degree to which a fixed object resembles a truly random object. In our context, a truly random 0/1 valued function will color approximately the same number of entries of a set 0 and 1. If we can prove that a fixed function $g$ shares this property, then we will make progress towards proving that $g$ has large communication complexity. Suppose $g : D \to \{0, 1\}$ is a function and $\mu$ is a distribution on the inputs to $g$. Suppose $S \subseteq D$ is a subset of the domain, and let $\chi_S$ be the characteristic function of the set $S$, so $\chi_S(x) = 1$ if $x \in S$, and $\chi_S(x) = 0$ if $x \notin S$. The discrepancy of $g$ with respect to $S$ and $\mu$ is defined as

$$\left| \mathbb{E}_\mu \left[ \chi_S(x) \cdot (-1)^{g(x)} \right] \right|.$$

When the distribution $\mu$ is understood from the context, we often call the above quantity the discrepancy of $g$ with respect to $S$. A large, nearly monochromatic rectangle (or cylinder intersection) must have high discrepancy. To prove this, recall that the bias of a set $S$ is defined to be

$$\text{bias}(S) = \max_b \Pr_{x \in S} [g(x) = b].$$

A large set with large bias must lead to high discrepancy:

**Fact 5.1.** If $S$ is a set of inputs with $\Pr_{x \in S} [x \in S] \geq \delta$, and $\text{bias}(S) \geq 1 - \epsilon$, then the discrepancy of $g$ with respect to $S$ is at least $(1 - 2\epsilon)\delta$. 

Besides being useful in communication complexity, the concept of discrepancy shows up in several other fields, like geometry and learning theory.
Proof. Only points inside \( S \) contribute to its discrepancy. Since a \((1 - \epsilon)\) fraction of these points have the same value under \( g \), the discrepancy is at least \( \delta(1 - \epsilon - \epsilon) = \delta(1 - 2\epsilon) \). \( \square \)

Functions that have small discrepancy must have large randomized communication complexity, as we show in the following theorem.

**Theorem 5.2.** For a fixed distribution \( \mu \), if the discrepancy of \( g \) with respect to every rectangle is at most \( \gamma \), then any protocol computing \( g \) with error \( \epsilon \) when the inputs are drawn from \( \mu \) must have communication complexity at least \( \log \left( \frac{1 - 2\epsilon}{\gamma} \right) \).

Proof. Suppose \( \pi \) is a deterministic protocol of communication complexity \( c \), and let \( \pi(x, y) \) denote its output. By Lemma 1.4, the leaves of the protocol \( \pi \) correspond to some rectangles \( R_1, \ldots, R_t \), with \( t \leq 2^c \). Then we have

\[
1 - 2\epsilon \leq \mathbb{E}_\mu \left[ (-1)^{\pi(x, y) + g(x, y)} \right] \\
= \mathbb{E}_\mu \left[ (-1)^{\pi(x, y)} \cdot (-1)^{g(x, y)} \right] \\
= \mathbb{E}_\mu \left[ \sum_{i=1}^t \chi_{R_i}(x, y) \cdot \varphi(R_i) \cdot (-1)^{g(x, y)} \right],
\]

where here \( \varphi(R_i) = -1 \) if the protocol outputs 1 on inputs from \( R_i \), and \( \varphi(R_i) = 1 \) if the protocol outputs 0. We can continue to bound:

\[
1 - 2\epsilon \leq \sum_{i=1}^t \mathbb{E}_\mu \left[ \chi_{R_i}(x, y) \cdot (-1)^{g(x, y)} \right] \\
\leq 2^c \cdot \max_R \left| \mathbb{E}_\mu \left[ \chi_R(x, y) \cdot (-1)^{g(x, y)} \right] \right|,
\]

where the maximum is taken over all choices of rectangles. Rearranging, we get

\[
2^c \geq \frac{1 - 2\epsilon}{\max_R \left| \mathbb{E}_\mu \left[ \chi_R(x, y) \cdot (-1)^{g(x, y)} \right] \right|}.
\]

The same result applies in the number-on-forehead model when we have a bound on the discrepancy with respect to cylinder intersections.

Given Theorem 5.2, we only need to bound the discrepancy of functions with respect to rectangles and cylinder intersections to prove communication lower bounds. In this chapter, we shall explore two techniques to bound the discrepancy. First we show how to appeal to Jensen’s inequality to bound discrepancy. Later, we show one can use concentration bounds from probability theory to control the discrepancy. To begin, let us explore some examples where Jensen’s inequality is useful in combinatorics.
Jensen’s Inequality in Combinatorics

Jensen’s inequality plays a key role in understanding many combinatorial properties of graphs. One kind of question where it is very useful is: how many edges must a graph have before it is forced to contain a small cycle? Given a graph on \( n \) vertices, the maximum number of edges it can have is \( \binom{n}{2} \). We say that the graph has edge density \( \epsilon \) if it has \( \epsilon \binom{n}{2} \) edges.

While there are graphs with constant edge density that have no 3-cycles, there are no graph with constant \( \epsilon \) that avoid 4-cycles.

**Lemma 5.3.** The number of 4-cycles in an \( n \)-vertex graph with \( \epsilon \binom{n}{2} \) edges is least \( n^4 \cdot \frac{(\epsilon - 1/n)^4 - 2/n}{4} \).

**Proof.** Let \( G \) be a graph with \( n \) vertices. For two vertices \( x, y \), let \( E(x, y) = 1 \) when there is an edge between the vertices \( x \) and \( y \), and \( 0 \) otherwise. Then if \( x, x', y, y' \) are 4 vertices chosen independently and uniformly at random from the graph, we use Jensen’s inequality to estimate

\[
\mathbb{E}_{x, x', y, y'} \left[ E(x, y) \cdot E(x', y') \cdot E(x, y') \cdot E(x', y') \right] \\
= \mathbb{E}_{x, x'} \mathbb{E}_{y} \left[ E(x, y) \cdot E(x', y') \right]^2 \\
\geq \mathbb{E}_{x, x'} \mathbb{E}_{y} \left[ E(x, y) \cdot E(x', y') \right]^2 \\
= \mathbb{E}_{y} \left[ \mathbb{E}_{x} \left[ E(x, y) \right]^2 \right]^2 \\
\geq \mathbb{E}_{y} \left[ E(x, y) \right]^4.
\]

This last quantity is at least \( (\epsilon - 1/n)^4 \), since \( E(x, y) \) is 0 only when \( x = y \) or \( (x, y) \) is not an edge.

Now, for any \( x, x', y, y' \), as long as \( x \neq x', y \neq y' \) and we have \( E(x, y)E(y, x')E(x', y')E(x, y') = 1 \) then \( x, x', y, y' \) form a 4-cycle. For uniformly random \( x, y, x', y' \), the probability that \( x = x' \) or \( y = y' \) is at most \( 2/n \). Moreover, each 4-cycle can be counted 4 times. This gives the stated bound. \( \square \)

We can use similar ideas as above to prove that every dense bipartite graph must contain a reasonably large bipartite clique. Here is a slightly different way to find a clique in a bipartite graph:

**Lemma 5.4.** Suppose \( G \) is a bipartite graph with edge density \( \epsilon \), and bipartition \( A, B \), with \( |A| = m, |B| = n \), and let \( k \leq \frac{\log n}{2 \log(2\epsilon/e)} \). Then if

For example the complete bipartite graph has \( 2n \) vertices and \( n^2 \) edges, but no 3-cycles.
\( e \geq 2k/m \), there are subsets \( Q \subseteq A, R \subseteq B \) with
\[
|Q| \geq k, \ |R| \geq \sqrt{n}
\]
such that every pair of vertices \( q \in Q, r \in R \) is connected by an edge.

**Proof.** Pick a uniformly random subset \( Q \subseteq A \) of size \( k \), and let \( R \) be all the common neighbors of \( Q \). Given any vertex \( b \in B \) that has degree \( d \geq k \), the probability that \( b \) is included in \( R \) is exactly
\[
\left( \frac{d}{k} \right)^k \geq \left( \frac{d/k}{e/m} \right)^k = \left( \frac{d}{e \cdot m} \right)^k.
\]
Let \( d_b \) be the degree of the vertex \( b \in B \). The expected size of the set \( R \) is at least
\[
\mathbb{E} |R| \geq \sum_{b \in B, d_b \geq k} \left( \frac{d_b}{e \cdot m} \right)^k \geq n \cdot \left( \frac{1}{n} \cdot \sum_{b \in B, d_b \geq k} \frac{d_b}{e \cdot m} \right)^k
\]
\[
= n \cdot \left( \frac{1}{e \cdot n} \cdot \sum_{b \in B, d_b \geq k} d_b \right)^k.
\]
Observe that \( \sum_{b \in B, d_b \geq k} d_b \) counts all the edges of the graph, except those that touch vertices of degree less than \( k \), so this quantity is at least \( e \cdot m \cdot n - k \cdot n \geq e \cdot m \cdot n / 2 \), since \( e \geq 2k/m \). Thus,
\[
\mathbb{E} |R| \geq n \cdot \left( \frac{1}{e \cdot m \cdot n} \cdot \frac{e \cdot m \cdot n}{2} \right)^k = n \cdot \left( \frac{e}{2e} \right)^k \geq \sqrt{n}.
\]
So there must be some choice of \( Q, R \) that proves the lemma. \( \Box \)

**Lower Bounds for Inner-Product**

**Say Alice and Bob are given** \( x, y \in \{0, 1\}^n \) **and want to compute** \( \langle x, y \rangle \mod 2 \). **We have seen that this requires** \( n + 1 \) **bits of communication using a deterministic protocol. Here we show that it requires at least** \( n/2 \) **bits of communication even using a randomized protocol. To prove this, we shall prove that discrepancy of the inner product function with respect to the uniform distribution is exponentially small, for every rectangle.**

**Lemma 5.5.** For any rectangle \( R \), the discrepancy of inner product with respect to \( R \) and the uniform distribution is at most \( 2^{-n/2} \).

**Proof.** Since \( R \) is a rectangle, we can write its characteristic function as the product of two functions \( A : \{0, 1\}^n \to \{0, 1\} \) and \( B : \{0, 1\}^n \to \{0, 1\}^n \) with
\[
\langle x, y \rangle = A(x) \cdot B(y).
\]
{0,1}. Thus we can write:
\[
\left( \mathbb{E}_{x,y} \left[ \chi_R(x,y) \cdot (-1)^{\langle x, y \rangle} \right] \right)^2 = \left( \mathbb{E}_{x,y} \left[ A(x) \cdot B(y) \cdot (-1)^{\langle x, y \rangle} \right] \right)^2
\]
\[
= \left( \mathbb{E}_x \left[ A(x) \mathbb{E}_y \left[ B(y) \cdot (-1)^{\langle x, y \rangle} \right] \right] \right)^2
\]
\[
\leq \mathbb{E}_x \left[ A(x)^2 \left( \mathbb{E}_y \left[ B(y) \cdot (-1)^{\langle x, y \rangle} \right] \right)^2 \right].
\]

Since \(0 \leq A(x) \leq 1\), we can drop \(A(x)\) from this expression to get:
\[
\left( \mathbb{E}_{x,y} \left[ \chi_R(x,y) \cdot (-1)^{\langle x, y \rangle} \right] \right)^2 \leq \mathbb{E}_x \left[ \left( \mathbb{E}_y \left[ B(y) \cdot (-1)^{\langle x, y \rangle} \right] \right)^2 \right]
\]
\[
= \mathbb{E}_{x,y,y'} \left[ B(y)B(y') \cdot (-1)^{\langle x, y \rangle + \langle x, y' \rangle} \right]
\]
\[
= \mathbb{E}_{x,y,y'} \left[ B(y)B(y') \cdot (-1)^{\langle x, y+y' \rangle} \right].
\]

In this way, we have completely eliminated the dependence on the set \(A\) from the calculation! We can also eliminate the set \(B\) too and write:
\[
\left( \mathbb{E}_{x,y} \left[ \chi_R(x,y) \cdot (-1)^{\langle x, y \rangle} \right] \right)^2 \leq \mathbb{E}_{x,y,y'} \left[ B(y)B(y') \cdot (-1)^{\langle x, y+y' \rangle} \right]
\]
\[
\leq \mathbb{E}_{y,y'} \left[ \mathbb{E}_x \left[ (-1)^{\langle x, y+y' \rangle} \right] \right].
\]

Now, whenever \(y+y'\) is not 0 modulo 2, the expectation over \(x\) is 0. On the other hand, the probability that \(y+y'\) is 0 modulo 2 is exactly \(2^{-n}\). So
\[
\mathbb{E}_{y,y'} \left[ \mathbb{E}_x \left[ (-1)^{\langle x, y+y' \rangle} \right] \right] = 2^{-n},
\]
proving the lemma.

\(\square\)

Lemma 5.5 and Theorem 5.2 together imply:

**Theorem 5.6.** Any 2-party protocol that computes the inner product function with error at most \(\epsilon\) over the uniform distribution must have communication at least \(n/2 - \log(1/(1-2\epsilon))\).

Similar ideas can be used to show that the communication complexity of the generalized inner product must be large in the number-on-forehead model\(^1\). Here each of the \(k\) players is given a binary string \(x_i \in \{0,1\}^n\). They want to compute \(\text{GIP}(x) = \sum_{i=1}^n \prod_{j=1}^k x_{ij} \mod 2\). We can show:

**Lemma 5.7.** The discrepancy of the generalized inner product function with respect to the uniform distribution and any cylinder intersection is at most \(e^{-n/4^{k-1}}\).

\(^1\) Babai et al., 1989

Each vector \(x_i\) can be interpreted as a subset of \([n]\). The protocol for computing the set intersection size gives a protocol for computing generalized inner product with communication \(O(k^4n/2^k)\).
Proof. Let $S$ be a cylinder intersection. Then its characteristic function can be expressed as the product of $k$ 0/1 valued functions $\chi_S = \prod_{i=1}^{k} \chi_i$, where $\chi_i$ does not depend on $x_i$. Thus we can write:

\[
\left( \mathbb{E}_x \left[ \chi_S(x) \cdot (-1)^{\text{GIP}(x)} \right] \right)^2 = \left( \mathbb{E}_{x_1, \ldots, x_{k-1}} \left[ \mathbb{E}_{x_k} \left[ \prod_{i=1}^{k} \chi_i(x) \cdot (-1)^{\text{GIP}(x)} \right] \right] \right)^2 \leq \mathbb{E}_{x_1, \ldots, x_{k-1}} \left[ \mathbb{E}_{x_k} \left[ \prod_{i=1}^{k-1} \chi_i(x) \cdot (-1) \sum_{j=1}^{k} \prod_{i=1}^{k\prime} x_i \right] \right].
\]

Now we can drop $\chi_k(x)$ from this expression to get:

\[
\left( \mathbb{E}_x \left[ \chi_S(x) \cdot (-1)^{\text{GIP}(x)} \right] \right)^2 \leq \mathbb{E}_{x_1, \ldots, x_{k-1}} \left[ \mathbb{E}_{x_k} \left[ \prod_{i=1}^{k-1} \chi_i(x) \cdot (-1) \sum_{j=1}^{k} \prod_{i=1}^{k\prime} x_i \right] \right],
\]

where here $x'_k$ is uniformly distributed and independent of $x_1, \ldots, x_{k-1}$, and $x' = (x_1, \ldots, x_{k-1}, x'_k)$. In this way, we have completely eliminated the function $\chi_k$ from the calculation! Repeating this trick $k-1$ times gives the bound

\[
\left( \mathbb{E}_x \left[ \chi_S(x) \cdot (-1)^{\text{GIP}(x)} \right] \right)^2 \leq \prod_{i=2}^{k-1} \left[ \mathbb{E}_{x_i} \left[ (-1)^{\sum_{j=1}^{k} \prod_{i=1}^{k\prime} x_i} \right] \right] \cdot \prod_{i=2}^{k-1} \left[ \mathbb{E}_{x_i} \left[ (-1)^{\sum_{j=1}^{k} \prod_{i=1}^{k\prime} x_i} \right] \right].
\]

Whenever there is a coordinate $j$ for which

\[
\prod_{i=2}^{k-1} (x_{i,j} + x_{i',j}) = 1 \quad \text{(mod 2)},
\]

the inner expectation is 0. When

\[
\prod_{i=2}^{k-1} (x_{i,j} + x_{i',j}) = 0 \quad \text{(mod 2)}
\]

for all coordinates $j$, then the inner expectation is 1. On the other hand, the probability that $\prod_{i=2}^{k-1} (x_i + x'_i) = 0 \quad \text{(mod 2)}$ is exactly $(1 - 2^{-k+1})^n$. So we get

\[
\left( \mathbb{E}_x \left[ \chi_S(x) \cdot (-1)^{\text{GIP}(x)} \right] \right)^{2^{k-1}} \leq (1 - 2^{-k+1})^n \leq e^{-n/2^{k-1}}.
\]

Fact: $1 - x \leq e^{-x}$ for all $x$. 

\[\square\]
By Lemma 5.7 and Theorem 5.2:

**Theorem 5.8.** Any randomized protocol for computing the generalized inner product in the number-on-forehead model with error \( \epsilon \) over the uniform distribution requires at least \( n/4^{k-1} - \log(1/(1-2\epsilon)) \) bits of communication.

**Lower Bounds for Disjointness in the Number-on-Forehead Model**

The discrepancy method may seem too weak to prove lower bounds against functions like disjointness, which do have large monochromatic rectangles. Suppose Alice and Bob are given two sets \( X, Y \subseteq [n] \) and want to compute disjointness. If we use a distribution on inputs that gives intersecting sets with probability at most \( \epsilon \), then there is a trivial protocol with error at most \( \epsilon \)—the parties can conclude that the sets intersect without communicating. On the other hand, if the probability of intersection is at least \( \epsilon \), then by averaging there must be some fixed coordinate \( i \) such that an intersection occurs in coordinate \( i \) with probability at least \( \epsilon/n \).

Setting \( R = \{ (X, Y) : i \in X, i \in Y \} \), we get

\[
\left| \mathbb{E} \left[ \chi_R(X, Y) \cdot (-1)^{\text{Disj}(X, Y)} \right] \right| \geq \frac{\epsilon}{n},
\]

so the discrepancy method can only give a lower bound of \( \Omega(\log n) \) using the approach we used for the inner product function.

Nevertheless, one can use discrepancy to give a lower bound on the communication complexity of disjointness\(^*\), even when the protocol is allowed to be randomized, by studying the discrepancy of a function that is related to disjointness under a carefully chosen distribution.

Consider the following distribution on sets. Let the universe consist of disjoint sets \( I_1, \ldots, I_m \). Alice gets \( m \) independently sampled sets \( X_1, \ldots, X_m \), where \( X_j \) is a uniformly random subset of \( I_j \), and Bob gets \( m \) independent random sets \( Y_1, \ldots, Y_m \), each of size 1, where \( Y_j \subseteq I_j \) for all \( j \). Let \( X = \bigcup_{j=1}^{m} X_j \) and \( Y = \bigcup_{j=1}^{m} Y_j \). We start by proving a weak bound on the discrepancy:

**Lemma 5.9.** For any rectangle \( R \),

\[
\left| \mathbb{E} \left[ \chi_R(X, Y) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(X_j, Y_j)} \right] \right| \leq \sqrt{\frac{1}{\prod_{j=1}^{m} |I_j|}}.
\]

**Proof.** As usual, we express \( \chi_R(X, Y) = A(X) \cdot B(Y) \) and carry out a convexity argument. We get:

\[\text{Sherstov, 2012; and Rao and Yehudayoff, 2015}\]

In fact, this is the only known method to prove lower bounds on the communication complexity of disjointness in the number-on-forehead model.
Whenever we can actually use it to give a linear lower bound on the communication of two-party deterministic protocols. Suppose a deterministic protocol for disjointness has communication $c$, and the intersection contains exactly 1 element. Set $I_i$ is picked uniformly at random, subject to the constraint that their intersection contains exactly 1 element. Set $X_i = \bigcup_{j=1}^{m} X_{i,j}$.

Thus, we get $\sum_{j=1}^{m} \text{Disj}(X_i, Y_j)$, for any fixing of $Y, Y'$, the inner expectation is 0 as long as $Y \neq Y'$, since $|Y_j| = |Y'_j| = 1$ for each $j$. The probability that $Y = Y'$ is exactly $1/\prod_{j=1}^{m} |I_j|$.  

Lemma 5.9 may not seem useful at first, because under the given distribution, the probability that $X, Y$ are disjoint is $2^{-m}$. However, we can actually use it to give a linear lower bound on the communication of two-party deterministic protocols. Suppose a deterministic protocol for disjointness has communication $c$. Then there must be at most $2^c$ monochromatic 1-rectangles $R_1, \ldots, R_T$ that cover all the 1's. Whenever $X, Y$ are disjoint, we have that $\sum_{j=1}^{m} \text{Disj}(X_i, Y_j) = m$. On the other hand, the probability that $X, Y$ are disjoint is exactly $2^{-m}$. Thus, we get

$$2^{-m} \leq \left| \sum_{t=1}^{T} \chi_{R_t}(X, Y) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(X_i, Y_j)} \right|$$

$$\leq \sum_{t=1}^{T} \left| \chi_{R_t}(X, Y) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(X_i, Y_j)} \right|$$

$$\leq \frac{2^c}{\prod_{j=1}^{m} |I_j|}.$$ 

Setting $|I_i| = 16$, and rearranging gives $c \geq m$, a linear lower bound on the deterministic two-party communication complexity of disjointness. While we have already seen several approaches for proving linear lower bounds on disjointness, this approach has a unique advantage: it works even in the number-on-forehead model.

Consider the distribution where for each $j = 1, 2, \ldots, m$, the set $X_{1,j} \subseteq I_i$ is picked uniformly at random, and $X_{2,j}, \ldots, X_{k,j} \subseteq I_i$ are picked uniformly at random, subject to the constraint that their intersection contains exactly 1 element. Set $X_i = \bigcup_{j=1}^{m} X_{i,j}$. Suppose the $i$'th player has the set $X_i$ written on his forehead.
Lemma 5.10. For any cylinder intersection $S$,  
\[ |E \left[ \chi_S(X) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(X_{1,j}, \ldots, X_{k,j})} \right] | \leq \prod_{j=1}^{m} \frac{2^{k-1} - 1}{|I_j|}. \]

Proof. We prove the lemma by induction on $k$. When $k = 2$, the statement was already proved in Lemma 5.9.

For ease of notation, we write 
\[ T_j = X_{1,j}, \ldots, X_{k,j} \]
to denote the input in the $j$th part of the universe. Suppose $\chi_S = \prod_{i=1}^{k} \chi_i$, where $\chi_i$ does not depend on $X_i$. As usual, we apply a convexity argument to bound:
\[
\left( E \left[ \chi_S(X) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(X_{1,j}, \ldots, X_{k,j})} \right] \right)^2 
\leq \left( E \left[ \chi(X) \cdot \left( E \left[ \prod_{i=1}^{k-1} \chi_i(X) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(T_j)} \right] \right) \right] \right)^2 
\leq \left( E \left[ \prod_{i=1}^{k-1} \chi_i(X) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(T_j)} \right] \right)^2 
= E \left[ \prod_{i=1}^{k-1} \chi_i(X) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(T_j) + \text{Disj}(T_j')} \right],
\]
where $X_k'$ is an independent copy of $X_k$ conditioned on $X_1, \ldots, X_{k-1}$.

Now, let $v_j$ denote the common intersection point of $X_{2,j}, \ldots, X_{k,j}$, and let $v_j'$ denote the common intersection point of $X_{2,j}, \ldots, X_{k-1,j}, X_k'$. Whenever $v_j = v_j'$, we have $\text{Disj}(T_j) = \text{Disj}(T_j')$, and so the $j$th term of the sum is 0 modulo 2. On the other hand, when $v_j \neq v_j'$, then any intersection in $T_j$ must take place in the set $X_{k,j} \setminus X_{k,j}'$, and any intersection in $T_j'$ must take place in $X_{k,j}' \setminus X_{k,j}$. So we can ignore the part of the sets $X_1, \ldots, X_{k-1}$ in $X_k \cap X_k'$ and in $X_k' \cap X_k$ and use induction to bound the discrepancy as follows. Let $Z_j$ be the random variable defined as
\[ Z_j = \begin{cases} 1 & \text{if } v_j = v_j', \\ \frac{(2^{k-2} - 1)^2}{|X_k \setminus X_k'| \cdot |X_k'| \setminus X_k|} & \text{otherwise}. \end{cases} \]

Since the $Z_j$'s are independent of each other, the inductive hypothesis gives:
\[ (5.1) \leq E \left[ \prod_{j=1}^{m} Z_j \right] \leq \prod_{j=1}^{m} E \left[ Z_j \right]. \]

We need a technical claim next:
Claim 5.11. Suppose a set \( Q \subseteq I_j \) is sampled by including a random element \( v \in I_j \) and adding every other element to \( Q \) independently with probability \( \gamma \neq 0 \). Then \( \mathbb{E} \left[ \frac{1}{|Q|} \right] \leq \frac{1}{\gamma |I_j|} \).

Proof.

\[
\mathbb{E} \left[ \frac{1}{|Q|} \right] = \sum_{Q \in \mathcal{Q}^{I_j}} \frac{(1/|I_j|) \cdot \gamma^{|Q|}(1 - \gamma)^{|I_j| - |Q|}}{|Q|} = \frac{1}{\gamma |I_j|} \sum_{Q \neq \emptyset} \gamma^{|Q|}(1 - \gamma)^{|I_j| - |Q|} \leq \frac{1}{\gamma |I_j|} (1 - \gamma)^{|I_j|} = \frac{1}{\gamma |I_j|}.
\]

We now use the claim. To bound \( \prod_{j=1}^m \mathbb{E} \left[ Z_j \right] \). Conditioned on the value of \( Q_j = X_{2,j} \cap \ldots \cap X_{k-1,j} \), the probability that \( v_j = v_j' \) is exactly \( 1/|Q_j| \). Given the common intersection point, every element of \( I_j \) is included in \( Q_j \) independently with probability \( \frac{1}{2} \). So by Claim 5.11,

\[
\Pr[v_j = v_j'] = \mathbb{E} \left[ \frac{1}{|Q_j|} \right] \leq \frac{2^{k-1} - 1}{|I_j|}.
\]

On the other hand, when \( v_j \neq v_j' \), we can bound

\[
Z_j = \frac{(2^{k-2} - 1)^2}{\sqrt{|X_{k,j} \setminus X_{k,j}'| \cdot |X_{k,j}' \setminus X_{k,j}|}} \leq \frac{(2^{k-2} - 1)^2}{2} \cdot \left( \frac{1}{|X_{k,j} \setminus X_{k,j}'|} + \frac{1}{|X_{k,j}' \setminus X_{k,j}|} \right) \text{ by the arithmetic mean - geometric mean inequality.}
\]

In this case, let \( Q_j = X_{k,j} \setminus X_{k,j}' \). Once again we see that \( Q_j \) is sampled by picking the value of \( v_j \) uniformly, and then every other element is included in \( Q_j \) independently with probability \( \frac{2^{k-2} - 1}{2^{k-2} - 1} \). So using Claim 5.11 again,

\[
\mathbb{E} \left[ \frac{1}{|X_{k,j} \setminus X_{k,j}'|} \right] \leq \frac{2(2^{k-1} - 1)}{(2^{k-2} - 1)|I_j|}.
\]

By symmetry, we also get:

\[
\mathbb{E} \left[ \frac{1}{|X_{k,j}' \setminus X_{k,j}|} \right] \leq \frac{2(2^{k-1} - 1)}{(2^{k-2} - 1)|I_j|}.
\]

Combining these bounds, we get

\[
\mathbb{E} \left[ Z_j \right] \leq \Pr[v = v'] + \frac{(2^{k-2} - 1)^2}{2} \cdot \left( \frac{1}{|X_{k,j} \setminus X_{k,j}'|} + \frac{1}{|X_{k,j}' \setminus X_{k,j}|} \right) \leq \frac{2^{k-1} - 1}{|I_j|} + \frac{2(2^{k-1} - 1)(2^{k-2} - 1)^2}{(2^{k-2} - 1)|I_j|} = \frac{(2^{k-1} - 1)^2}{|I_j|}.
\]
Lemma 5.10 can be used to prove a lower bound of $\Omega(n/4^k)$ on the number-on-forehead deterministic communication complexity of disjointness. Suppose a deterministic protocol for disjointness has communication $c$. Then there are at most $2^c$ monochromatic cylinder intersections $S_1, \ldots, S_T$ that cover all the 1’s. Whenever $X_1, \ldots, X_k$ are disjoint, we have that $\sum_{j=1}^m \text{Disj}(X_{1,j}, X_{2,j}, \ldots, X_{k,j}) = m$. On the other hand, the probability that $X_1, \ldots, X_k$ are disjoint is exactly $2^{-m}$. Thus, we get

$$2^{-m} \leq \mathbb{E} \left[ \sum_{t=1}^T \chi_{S_t}(X) \cdot (-1)^{\sum_{j=1}^m \text{Disj}(X_{1,j}, \ldots, X_{k,j})} \right]$$

$$\leq \sum_{t=1}^T \left| \mathbb{E} \left[ \chi_{S_t}(X) \cdot (-1)^{\sum_{j=1}^m \text{Disj}(X_{1,j}, \ldots, X_{k,j})} \right] \right|$$

$$\leq 2^c \cdot \left( \prod_{j=1}^m \frac{2^{k-1} - 1}{|I_j|} \right).$$

Setting $|I_j| = 16 \cdot (2^{k-1} - 1)^2$, we get that

$$c \geq m = \frac{n}{16(2^{k-1} - 1)^2}.$$ 

**Theorem 5.12.** Any deterministic protocol for computing disjointness in the number-on-forehead model requires $\frac{n}{16(2^{k-1} - 1)^2}$ bits of communication.

The best lower bound on the randomized communication complexity is given by the following theorem:

**Theorem 5.13.** Any randomized protocol for computing disjointness in the worst case with error $1/3$ in the number-on-forehead model requires $\Omega\left(\frac{\sqrt{n}}{2^k}\right)$ bits of communication.

Using Concentration Bounds to Control Discrepancy

Several tools of probability theory give useful ways to control discrepancy. Here we explore two examples: one using Chernoff-Hoeffding bounds, and the other using Talagrand’s inequality.

Lower bounds on the randomized communication complexity of Disjointness

We start by trying to prove lower bounds on the randomized communication complexity of 2 party protocols computing the disjointness function. We have already discussed a major obstacle to using discrepancy to prove lower bounds on disjointness—for any distribution
on inputs, either the distribution gives disjoint sets with high probability, or there are large nearly monochromatic rectangles contain inputs that mostly intersect. Nevertheless, we can prove some lower bounds by estimating the discrepancy of disjointness with respect to rectangles that mostly contain disjoint inputs.

We shall prove\textsuperscript{4}:

**Theorem 5.14.** Any randomized 2-party protocol computing disjointness with error $1/3$ must have communication $\Omega(\sqrt{n})$.

To prove this theorem, we start by defining a hard distribution on inputs. For a parameter $\gamma$, we independently sample sets $X, Y \subseteq [n]$ by including each element in each set independently with probability $\gamma$. We set $\gamma$ so that the probability that the sets are disjoint is exactly $(1 - \gamma^2)^n = \frac{1}{2}$.

We have the estimates:

$$\frac{1}{\sqrt{2n}} \leq \gamma \leq \ln 2 \sqrt{\frac{1}{n}}.$$  

The heart of the proof is the following lemma, which shows that we cannot have a large rectangle with too many disjoint pairs of inputs:

**Lemma 5.15.** There are constants $0 < \alpha, \beta < 1$, such that for any rectangle $R$, if

$$\Pr[(X,Y) \in R] \geq e^{-\frac{\sqrt{n}}{4}},$$

then

$$\Pr[X, Y are disjoint | (X,Y) \in R] < 1 - \beta.$$  

Lemma 5.15 implies the lower bound in Theorem 5.14. Indeed, we can always reduce the error of the protocol by repeating it a constant number of times, so suppose for the sake of contradiction that there is a protocol with communication $c$ that computes disjointness with error at most $\epsilon$, for some small constant $\epsilon$. Since the probability that $X,Y$ are disjoint is $1/2$, the protocol must output that the sets are disjoint with probability at least $1/2 - \epsilon$. Then Theorem 3.6 implies that there is a rectangle of density $\sqrt{\epsilon} \cdot 2^{-c}$ that consists almost entirely of disjoint rectangles. By the Lemma, any such rectangle must have density at most $e^{\sqrt{n}}$, proving that $c$ must be at least $\Omega(\sqrt{n})$.

To prove Lemma 5.15, we repeatedly use the Chernoff-Hoeffding bound:

**Proof of Lemma 5.15.** We shall set $\alpha, \beta$ to be small enough constants during the proof. Let $R = A \times B$ be any rectangle of density $e^{-\alpha} / \sqrt{n}$. Define:

$$A' = \{ X \in A : \Pr_{Y}[X,Y \text{ are intersecting} | Y \in B] \leq 2\beta \}.$$  

For $\gamma \leq 1/2$,

$$2^{-2\gamma^2 n} \leq (1 - \gamma^2)^n \leq e^{-\gamma^2 n}.$$  

Can you think of an example showing that the Lemma is tight?
By Markov’s inequality, at most 1/2 of the rows in $A$ are not in $A'$. So, we must have that
\[
\Pr[X \in A'] \geq \Pr[X \in A'|X \in A] \cdot \Pr[X \in A]
\geq (1/2) \cdot \Pr[(X, Y) \in R] \geq e^{-\gamma n}/2.
\]

**Claim 5.16.** Let $\ell = \lceil \frac{1}{\gamma^2} \rceil$. If $\alpha$ is small enough, there are sets

\[X_1, X_2, \ldots, X_\ell \in A'\]

such that for all $i$,
\[
\frac{1}{2} \cdot \gamma n \leq |X_i| \leq \frac{3}{2} \cdot \gamma n,
\]

and
\[
\left| X_i - \bigcup_{j=1}^{i-1} X_j \right| \geq \frac{\gamma n}{4}.
\]

**Proof.** We find the sequence of sets $X_1, \ldots, X_\ell$ inductively. In the $i$'th step, consider the experiment of picking $X_i$ according to the distribution where each element is included in $X_i$ independently with probability $\gamma$. The expected size of $X_i$ is $\gamma n$. So by the Chernoff-Hoeffding bound,
\[
\Pr[||X_i| - \gamma n| > \gamma n/2] \leq 2e^{-(1/2)^2 \gamma n/3}.
\]

$\bigcup_{j=1}^{i-1} X_j$ is a set of size at most $\frac{1}{\gamma^2} \cdot \frac{3 \gamma n}{2} = \frac{n}{6}$, so if $\mu$ is the the expected number of elements in $X_i$ that are not in $\bigcup_{j=1}^{i-1} X_j$, we have $\mu \geq \gamma 5n/6$. So, the probability that fewer than $\gamma n/4$ elements are in $X_i$ but not in the union is at most
\[
\Pr \left[ \left| X_i - \bigcup_{j=1}^{i-1} X_j \right| < \gamma n/4 \right] \leq e^{-\left(\frac{5}{6} - \frac{1}{4}\right)^2 \gamma n/18}.
\]

We set $\alpha$ to be small enough so that
\[
\Pr[X \in A'] > 2e^{-(1/2)^2 \gamma n/3} + e^{-\left(\frac{5}{6} - \frac{1}{4}\right)^2 \gamma n/18}.
\]

This ensures that there will be some $X_i \in A$ satisfying all three conditions for every step.

Let $X_1, \ldots, X_\ell$ be as promised by Claim 5.16, and for each $i$, define
\[
Z_i = X_i - \bigcup_{j=1}^{i-1} X_j.
\]

The sets $Z_1, \ldots, Z_\ell$ are disjoint, and each is of size at least $\gamma n/4$. If we pick $Y$ at random by including each element in $Y$ with probability $\gamma$, the probability that $Y$ is disjoint from $Z_i$ is at most
\[
(1 - \gamma)\frac{\gamma n}{4} \leq e^{-\gamma^2 n/4} \leq e^{-1/8}.
\]
So, the expected number of the sets $Z_1, \ldots, Z_\ell$ that $Y$ intersects is at least $\ell(1 - e^{-1/8})$. Applying the Chernoff-Hoeffding bound once more, we get that the probability $Y$ intersects at most $(1 - e^{-1/8})\ell/2$ of the $Z_i$'s is at most
\[
e^{-\frac{1}{2} \ell (1 - e^{-1/8})/3} < e^{-\frac{a^2}{72}/2},
\]
if we choose $a$ to be a small enough constant.

Set $\beta$ so that $4\beta \ell < (1 - e^{-1/8})\ell/2$, and define
\[B' = \{Y \in B : Y \text{ intersects at most } 4\beta \ell \text{ of the sets } Z_1, \ldots, Z_\ell\}.
\]
By the definition of $A'$, a random element $Y \in B$ intersects at most $2\beta \ell$ of the sets $Z_1, \ldots, Z_\ell$ in expectation. Thus, the probability that $Y$ intersects more than $4\beta \ell$ fraction of the sets is at most $1/2$ by Markov’s inequality. We get:
\[
\Pr[Y \in B'] = \Pr[Y \in B' | Y \in B] \cdot \Pr[Y \in B] \\
\geq (1/2) \cdot \Pr[(X, Y) \in R] > e^{-\frac{a^2}{72}/2}.
\]
This is a contradiction, because the probability that $Y$ intersects so many of the $Z_i$’s was shown to be smaller.

The Gap-Hamming Problem

Although the bounds we obtained for the randomized communication complexity of disjointness are not tight, a similar approach does give tight bounds for the Gap-Hamming problem. Suppose Alice and Bob have inputs $x, y \in \{+1, -1\}^n$. They wish to estimate the inner-product $\langle x, y \rangle$ of the two strings. This task is equivalent to computing the Hamming distance, defined to be $|\{i \in [n] : x_i \neq y_i\}|$. Indeed, the Hamming distance is exactly $1 - \frac{\langle x, y \rangle}{2}$.

The fooling set method can be used to show that the deterministic communication complexity of this problem is $\Omega(n)$. In Chapter 3, we showed that there is a randomized protocol that can estimate the hamming distance up to an additive factor of $\sqrt{n}/e$, with communication $O(e^2n)$. Here we prove that this protocol is essentially the best we can hope for. We say that a randomized protocol $\pi$ approximates the inner-product up to a parameter $m$ if for every $x, y$, $\Pr[|\langle x, y \rangle - \pi(x, y)| > m] \leq 2/3$.

We shall prove:

**Theorem 5.17.** Any protocol that approximates the hamming distance up to $\sqrt{n}$ must have communication complexity $\Omega(n)$.

Let $X, Y \in \{+1, -1\}^n$ be independent and uniformly random. As in the lower bound for disjointness, the key step in the argument is to prove that there are no large rectangles where the magnitude of the inner product is small. We shall prove:
Lemma 5.18. There are constants $0 < \alpha, \beta < 1$, and a number $t > 0$ such that if $R$ is a rectangle with

$$\Pr[(X, Y) \in R] > 2^{-\alpha n},$$

then

$$\Pr \left[ |\langle X, Y \rangle| \leq \sqrt{n}/t |(X, Y) \in R \right] \leq 1 - \beta.$$

To prove the lemma, we appeal to a beautiful result from probability called Talagrand’s inequality, as well as the singular value decomposition of matrices. Before discussing how to use these ideas to prove the Lemma, let us first show how to use the Lemma to prove Theorem 5.17.

We start by arguing that the approximation factor of any protocol can be made small by increasing the communication by a constant factor. Let $\ell = 9t^2n$, where $t$ is as in Lemma 5.18. Suppose there is a protocol $\pi$ with communication complexity $o(\ell)$ that approximates the inner product of $x', y' \in \{+1, -1\}^\ell$ up to $\sqrt{\ell}$. Then, by repeating the protocol a small number of times, and taking the majority outcome, we obtain a protocol with communication $o(\ell)$ that approximates the inner-product up to $\sqrt{\ell}$, and makes an error with probability $o(1)$. If $x, y \in \{+1, -1\}^n$, Alice and Bob can repeat each coordinate of their input $9t^2$ times to obtain inputs $x', y' \in \{+1, -1\}^\ell$, and run the protocol on these inputs. We have $\langle x', y' \rangle = 9t^2 \cdot \langle x, y \rangle$, so if the estimated inner product is $z$, and the protocol does not make an error, we have:

$$3t\sqrt{n} = \sqrt{\ell} \geq |z - \langle x', y' \rangle| = 9t^2 \cdot \left| \frac{z}{9t^2} - \langle x, y \rangle \right|,$$

so $\left| \frac{z}{9t^2} - \langle x, y \rangle \right| \leq \sqrt{n}/3t$. Alice and Bob can compute $z/9t^2$ to obtain a strong estimate for the inner-product.

There is a significant probability that the inner-product will have a magnitude that is at most $\sqrt{n}/3t$. To see this, let $Z_1, \ldots, Z_n$ be bits defined by

$$Z_i = \begin{cases} 
1 & \text{if } X_i \neq Y_i, \\
0 & \text{otherwise}.
\end{cases}$$

Then $\sum_{i=1}^n Z_i = \frac{n - \langle X, Y \rangle}{2}$. Moreover, the expected value of the sum is $n/2$. So, by the Chernoff-Hoeffding bound,

$$\Pr[|\langle X, Y \rangle| > \sqrt{n}/3t] = \Pr \left[ \left| \sum_{i=1}^n Z_i - \frac{n}{2} \right| > \sqrt{n}/6t \right] \leq e^{-\frac{(1/3t\sqrt{n/2})^2}{256}} = e^{-\frac{1}{3t}}.$$
This means that the protocol must compute an outcome of magnitude at most $\sqrt{n}/3t$ with probability at least $1 - e^{-\frac{1}{54} - o(1)}$. So by Theorem 3.6, there must be a rectangle $R$ of density at least $2^{-o(n)}$, where the protocol computes an estimate of magnitude at most $\sqrt{n}/3t$, and yet makes an error with probability $o(1)$. Lemma 5.18 implies that this is impossible.

It only remains to prove Lemma 5.18. The first technical tool we need is called Talagrand’s inequality. Talagrand’s inequality concerns the length of the projection of a uniformly random vector $x \in \{+1, -1\}^n$ to a given vector space $V$. If $V$ is a $d$-dimensional vector space, let $\text{proj}_V(x)$ denote the projection of $x$ to $V$. Figure 5.4, shows various ways of projecting the cube $\{+1, -1\}^n$ to different 2-dimensional subspaces.

It is easy to see that the expected value of the square of the length of the projection $\|\text{proj}_V(x)\|^2$ is $d$: if $e_1, e_2, \ldots, e_d$ is an orthonormal basis for $V$, then the expected squared length of the projection is

$$E \left[ \sum_{i=1}^d (x,e_i)^2 \right] = \sum_{i=1}^d E \left[ (x,e_i)^2 \right],$$

and for each $e_i$, we have:

$$E \left[ (x,e_i)^2 \right] = E \left[ (\sum_{j=1}^n e_{ij}x_j)^2 \right] = \sum_{j=1}^n E \left[ (e_{ij}x_j)^2 \right] + \sum_{j \neq j'} E \left[ (e_{ij}x_j)(e_{ij'}x_{j'}) \right] = \sum_{j=1}^n E \left[ (e_{ij}x_j)^2 \right] = \|e_i\|^2 = 1.$$

This might lead us to guess that length of the projection should typically be about $\sqrt{d}$. Talagrand’s inequality shows that the length of the projection is concentrated around this quantity:

**Theorem 5.19.** There is a constant $\gamma > 0$ such that for any $d$-dimensional vector space $V \subseteq \mathbb{R}^n$,

$$\Pr_{x \in \{+1,-1\}^n} \left[ \|\text{proj}_V(x)\| - \sqrt{d} \geq s \right] < 4e^{-\gamma s^2}.$$

Intuitively Talagrand’s inequality suggests that a statement like Lemma 5.18 ought to hold. If $R = A \times B$ is not exponentially small, then for $k = n/16$, say, one should be able to find vectors $x_1, \ldots, x_k \in A$ are essentially orthogonal to each other—a uniformly random vector $x_i \in \{+1,-1\}^n$ will have a small projection onto the previous vectors except with exponentially small probability, so the probability that it falls in $A$ and has small projection onto the previous vectors
will be positive. Now if \( x_1, \ldots, x_k \) were perfectly orthogonal, we could apply Talagrand’s inequality once again to prove the Lemma. Consider the experiment of picking a uniformly random \( Y \). Since \( Y \)'s projection to the space spanned by \( x_1, \ldots, x_k \) must be of length about \( \sqrt{k} \approx \sqrt{n} \) by Talagrand’s inequality, we must have \( |\langle y, x_i \rangle| > \sqrt{k} \) for most coordinates \( i \), except with exponentially small probability. Since \( B \) is not exponentially small, the probability that \( Y \) is in \( B \) and still has the above properties is significant.

Of course, \( x_1, \ldots, x_k \) need not be perfectly orthogonal, so to turn these intuitions into a proof, we need to use the concept of singular value decompositions. Let \( M \) be an arbitrary \( m \times n \) matrix with real valued entries and \( m \leq n \). Then, one can always express

\[
M = \sum_{i=1}^{m} \sigma_i \cdot u_i v_i^\top,
\]

where \( u_1, \ldots, u_m \) are orthogonal \( m \times 1 \) column vectors with \( \|u_i\| = 1 \), \( v_1, \ldots, v_m \) are orthogonal \( m \times 1 \) column vectors with \( \|v_i\| = 1 \), and \( \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m \geq 0 \) are real numbers called the singular values of \( M \).
The singular value decomposition gives a nice way to interpret the action of $M$ on a $n \times 1$ column vector $y$:

$$My = \sum_{i=1}^{m} \sigma_i u_i v_i^\top y = \sum_{i=1}^{m} \sigma_i \langle v_i, y \rangle \cdot u_i.$$

In words, $My$ can be viewed as the result of scaling and rotating $y$. To compute $My$, express $y$ in the basis given by $v_1, \ldots, v_m$ and scale the coefficients of $y$ using the singular values. Then rotate the result to the basis given by $u_1, \ldots, u_m$.

The singular values characterize how much the matrix $M$ can stretch an $n \times 1$ column vector $y$. Indeed, the properties of the decomposition allow us to conclude:

$$\|My\|^2 = y^\top M^\top My = y^\top \left( \sum_{i=1}^{m} \sigma_i v_i u_i^\top \right) \left( \sum_{j=1}^{m} \sigma_j u_j v_j^\top \right) y = y^\top \left( \sum_{i=1}^{m} \sigma_i^2 v_i u_i^\top \right) y = \sum_{i=1}^{m} \sigma_i^2 \cdot (y^\top v_i)^2.$$  \hspace{1cm} (5.2)

Moreover, the $i$'th term in this sum is at most $\sigma_i^2 \cdot \|y\|^2$, and this quantity is at most $\sigma_1^2 \cdot \|y\|^2$. Thus, $\|My\| \leq \sigma_1 \cdot \|y\|$, and for a given value of $\|y\|$, the length $\|My\|$ is maximized when $y$ is proportional to $v_1$.

**Proof of Lemma 5.18.** Without loss of generality, we assume that $n$ is a power of 4. This will ensure that many of the ratios in the proof are integers. Let $R = A \times B$ be the given rectangle. We shall set $\alpha, \beta, t$ as needed in the proof. We assume towards a contradiction that $R$ has density at least $2^{-\alpha n}$, and yet a random $(X, Y)$ in $R$ satisfy $|\langle X, Y \rangle| \leq \sqrt{n}/t$ with probability at least $1 - \beta$. Define:

$$A' = \left\{ X \in A : \Pr_{Y \in B} \left[ |\langle X, Y \rangle| > \sqrt{n}/t \right] \leq 2\beta \right\}.$$

By Markov’s inequality, we must have that

$$\Pr[X \in A'] \geq \frac{1}{2} \cdot \Pr[\langle X, Y \rangle \in R] \geq 2^{-\alpha n - 1}.$$

The first step of the proof is to use Talagrand’s inequality to find a set of nearly orthogonal vectors in $A'$:

**Claim 5.20.** If $k = \frac{n}{16}$, there are strings $x_1, x_2, \ldots, x_k \in A'$ such that for all $i$, if $V_i$ denotes the span of $x_1, \ldots, x_i$, then

$$\|\text{proj}_{V_{i-1}}(x_i)\| \leq \frac{\sqrt{n}}{2}.$$
Proof. We find the sequence $x_1, \ldots, x_k \in \{+1, -1\}^n$ inductively. In the $i$'th step, consider the experiment of picking $x_i$ according to the uniform distribution. The dimension of $V_{i-1}$ is at most $k$, so by Theorem 5.19, the probability that the length of the projection exceeds $\sqrt{n}/2$ is at most $4e^{-\gamma(\sqrt{n}/4)^2}$. We set $\alpha$ to be a small enough constant such that $\Pr[X \in A'] \geq 2^{-\alpha n} > 4e^{-\gamma(\sqrt{n}/4)^2}$, to guarantee that there must be some $x_i \in A'$ satisfying the requirements for each $i$. □

For each subset $S \subseteq [k]$ of size $|S| = k - 4\beta$, we define the set

$$B_S = \{ y \in B : \text{for all } i \in S, \langle x_i, y \rangle \leq \sqrt{n}/t \}.$$ 

At least half the $y$'s in $B$ must satisfy the property that the number of coordinates $i$ for which $|\langle x_i, y \rangle| > \sqrt{n}/t$ is at most $4\beta$. So, by averaging, there must be one set $S$ for which

$$\Pr[Y \in B_S] \geq 2^{an} \gamma(\frac{k}{\beta})^2.$$ 

Without loss of generality, we may assume that $\{1, 2, \ldots, m\}$ is such a set.

Now, we use the singular value decomposition. Let $x_1, \ldots, x_m$, and $V_1, \ldots, V_m$ be as in Claim 5.20. Define the $m \times n$ matrix $M$ whose rows are $x_1, \ldots, x_m$. Suppose $M$ has the singular value decomposition:

$$M = \sum_{i=1}^{m} \sigma_i \cdot u_i^T v_i.$$ 

We claim that the largest singular values of $M$ cannot be too different from each other, by establishing two bounds:

$$\sum_{i=1}^{m} \sigma_i \geq m \sqrt{n}/2, \quad (5.3)$$

and

$$\sum_{i=1}^{m} \sigma_i^2 = mn. \quad (5.4)$$

Let us see how to use these bounds to completely the proof of Lemma 5.18.

We claim that there must be at least $m/16$ singular values of magnitude at least $\sqrt{n}/4$, since

$$\sum_{i>m/16} \sigma_i = \sum_{i=1}^{m} \sigma_i - \sum_{i=1}^{m/16} \sigma_i \geq \frac{m \sqrt{n}}{2} - \sqrt{\sum_{i=1}^{m/16} \sigma_i^2 \cdot \sqrt{m/16}} \geq \frac{m \sqrt{n}}{2} - \sqrt{mn \cdot \sqrt{m}/4} = m \sqrt{n}/4.$$ 

by the Cauchy-Schwartz inequality, and (5.3)
So, $\sigma_{m/16+1} \geq \sqrt{n}/4$. Let $V$ denote the span of $v_1, \ldots, v_{m/16}$. If $y \in \{+1, -1\}^n$ is uniformly random, then by Theorem 5.19,
\[
\Pr[|\|\text{proj}_V(y)\|-\sqrt{m}/4| \geq \sqrt{n}/4] < 4e^{-\gamma m/64} < \frac{2^{-an-1}}{\binom{k}{\beta k}},
\]
if we choose $\alpha, \beta$ to be small enough constants. Thus, there must be a $y \in B_2^n$ with $\|\text{proj}_V(y)\| \geq \sqrt{m}/8$. On the other hand, by (5.2),
\[
\frac{mn}{t^2} \geq \sum_{i=1}^m \langle x_i, y \rangle^2 = \|My\|^2 = \sum_{i=1}^{m/16} \sigma_i^2 \cdot (y^Tv_i)^2 \geq \frac{n}{16} \sum_{i=1}^{m/16} (y^Tv_i)^2 \geq \frac{n}{16} \cdot \|\text{proj}_V(y)\|^2 \geq \frac{n}{16} \cdot \frac{m}{64},
\]
which is a contradiction if $t \geq 32$.

To prove (5.3), let $z_1, \ldots, z_m$ be the orthogonal vectors obtained by setting $z_1 = x_1$, and for $i > 1$,
\[
z_i = x_i - \text{proj}_{V_{i-1}}(x_i).
\]
Now if $Z$ is the matrix with rows $z_1, \ldots, z_m$, then on the one hand, we have:
\[
\text{tr}(MZ^T) = \sum_{i=1}^m \langle x_i, z_i \rangle = \sum_{i=1}^m \langle x_i, x_i \rangle - \sum_{i=1}^m \langle x_i, \text{proj}_{V_{i-1}}(x_i) \rangle \geq mn - \sum_{i=1}^m \|x_i\| \cdot \|\text{proj}_{V_{i-1}}(x_i)\| \geq mn - m \cdot \sqrt{n} \cdot \sqrt{n} = mn/2.
\]
On the other hand, we have
\[
\text{tr}(MZ^T) = \text{tr}((\sum_{i=1}^m \sigma_i \cdot u_i v_i^T)Z^T) = \sum_{i=1}^m \sigma_i \cdot \text{tr}(u_i v_i^T Z^T) \leq \sum_{i=1}^m \sigma_i \cdot \|u_i\| \cdot \|Z v_i\|.
\]
The rows of $Z$ are orthogonal, and of length at most $\sqrt{n}$, so we get:
\[
\|Z v_i\|^2 = v_i Z^T v_i^T = \sum_{j=1}^n v_{i,j}^2 \|z_i\|^2 \leq \sum_{j=1}^n v_{i,j}^2 n \leq n.
\]
Thus, we have
\[
\text{tr}(MZ^T) \leq \sqrt{n} \cdot \sum_{i=1}^{m} \sigma_i.
\]

To prove (5.4), consider that on the one hand,
\[
\text{tr}(MM^T) = \sum_{i=1}^{m} x_i^\top x_i = mn.
\]

On the other hand, it is the same as the trace
\[
\text{tr} \left( \sum_{i=1}^{m} \sigma_i^2 v_i v_i^\top \right) = \sum_{i=1}^{m} \sigma_i^2 \cdot \text{tr} (v_i v_i^\top) = \sum_{i=1}^{m} \sigma_i^2.
\]
6

Information

Shannon’s seminal work defining entropy\(^1\) has had a big impact on many areas in mathematics. Shannon’s goal was to quantify the amount of information or entropy contained in a random variable \(X\). His definition leads to a theory that is both elegant and useful in many areas, including communication complexity. We begin this chapter with some simple examples that help to demonstrate the usefulness of the concepts of information. Later, we show how these concepts can be used to understand communication complexity.

Entropy

In many cases, the amount of information contained in a message is not the same as the length of the message. Here are some examples:

- Consider a protocol where Alice’s first message to Bob is a \(c\)-bit string that is always \(0^c\), no matter what her input is. This message does not convey any information to Bob. Alice and Bob may as well imagine that this first message has already been sent, and so reduce the communication of the first step to 0.

- Consider a protocol where Alice’s first message to Bob is a uniformly random string from a set \(S \subseteq \{0, 1\}^c\), with \(|S| \ll 2^c\). In this case, the parties could use \(\log |S|\) bits to index the elements of the set, reducing the communication from \(c\) to \(\log |S|\).

- Consider a protocol where Alice’s first message to Bob is the string \(0^n\) with probability \(1 - \epsilon\), and is a uniformly random \(n\) bit string with the probability \(\epsilon\). In this case, one cannot encode every message using fewer than \(n\) bits. However, Alice can send the bit 0 to encode the string \(0^n\), and the string 1\(x\) to encode the \(n\) bit string \(x\). Although the first message is still quite long in the worst case, the expected length of the message is \(1 - \epsilon + \epsilon(n + 1) = 1 + \epsilon n \ll n\).

\(^1\) Shannon, 1948
Shannon’s definition of entropy gives a general way to compute the optimal encoding length for messages. Given a random variable $X$ with probability distribution $p(x)$, the entropy of $X$ is defined to be

$$H(X) = \sum_x p(x) \cdot \log \frac{1}{p(x)} = \mathbb{E}_{p(x)} \left[ \log \frac{1}{p(x)} \right].$$

This definition may seem technical at first sight, but as we shall see, it enjoys some simple, intuitive and useful properties. One simple property is that the entropy is always non-negative, since every term in the sum is non-negative.

Suppose $X$ is a uniformly random element of $[n]$. Then its entropy is

$$H(X) = \sum_{x \in [n]} \frac{1}{n} \log n = \log n.$$ 

In fact, the uniform distribution has the maximum entropy of any distribution on a finite set: if $X \in [n]$ then

$$H(X) = \mathbb{E}_{p(x)} \left[ \log \frac{1}{p(x)} \right] \leq \log \mathbb{E}_{p(x)} \left[ \frac{1}{p(x)} \right] = \log n, \quad (6.1)$$

where the inequality follows by Jensen’s inequality applied to the convex function $\log$. This property of the entropy function makes it particularly useful as a tool to count the size of sets, since it relates entropy to the size of a set.

### Axiomatic definition

A fundamental property of the entropy function is that it can be axiomatically defined\(^2\). Let $H(X)$ be any notion of the entropy of a random variable $X$. Then it is natural to require this notion of entropy to satisfy the following axioms:

**Symmetry** $H(\pi(X)) = H(X)$ for all permutations $\pi$ of $[n]$.\(^*\)

![Figure 6.1: The entropy of a bit with $p(1) = \epsilon$.](image)

By convention $0 \log(1/0) = 0$. In fact, most of the properties of entropy follow by Jensen’s inequality.

\(^*\)Shannon, 1948
Several other axiomatic definitions of the entropy are known.

The entropy is always non-negative.

by Markov’s inequality

Continuity  $H$ should be continuous in the distribution of $X$.

Monotonicity  If $X$ is uniform over a set of size $n$, then $H(X)$ increases as $n$ increases.

Chain-Rule  If $X = (Y, Z)$ then

$$H(X) = H(Y) + \sum_y p(Y = y) \cdot H(Z|Y = y).$$

Shannon proved that any $H$ satisfying these axioms must be proportional to the entropy function that he defined.

Coding

Shannon showed that the entropy of $X$ characterizes the expected number of bits that need to be transmitted to encode $X$. On the one hand, if there is an encoding of $X$ that has expected length $k$, then $X$ can be encoded by a string of length at most $10k$ most of the time. So, one would expect that $X$ takes one of $2^{O(k)}$ values most of the time, and thus the expected value of $\log(1/p(x))$ should be at most $O(k)$. On the other hand, if $H(X) = k$, we may assume that $X \in [n]$ and $p(1) \geq p(2) \geq \ldots$. Since $\sum_{j=1}^{n} p(j) \leq 1$ for all $i$, it follows that $p(i) \leq 1/i$. Writing the integer $i$ takes about $\log i$ bits, so the expected length of the encoding should be about $\sum_{i \in [n]} p(i) \cdot \log i \leq H(X)$.

Formally, we can prove:\n
\textbf{Theorem 6.1.}  $X$ can be encoded using a message whose expected length is at most $H(X) + 1$. Conversely, every encoding of $X$ has expected length at least $H(X)$.

\textbf{Proof.}  Without loss of generality, suppose that $X$ is an integer from $[n]$, with $p(i) \geq p(i+1)$ for all $i$. Let $\ell_i = \lceil \log(1/p(i)) \rceil$. To prove that $X$ can be encoded using messages of length $H(X) + 1$, we shall construct a protocol tree for Alice to send $X$ to Bob. Each $i$ in the tree will correspond to a vertex $v_i$ at depth $\ell_i$. The expected length of the message will thus be

$$\sum_i p(i) \cdot \ell_i \leq \sum_i p(i)(\log(1/p(i)) + 1) = H(X) + 1.$$

The encoding is done greedily. In the first step, we pick a vertex $v_1$ of the complete binary tree at depth $\ell_1$ and let that vertex represent 1. We delete all of $v_1$’s descendants, so that $v_1$ becomes a leaf. Next, we find an arbitrary vertex $v_2$ at depth $\ell_2$ that has not been deleted, and use it to represent 2. We continue in this way, until every element of $[n]$ has been encoded.

$\text{Shannon, 1948}$

Can you think of an example that shows that the expected length needs to be at least $H(X) + 1$?
This process can fail only if for some $j$ there are no available leaves at depth $\ell_j$. We show that this never happens. For $i < j$, the number of vertices at depth $\ell_j$ that are deleted in the $i$'th step is exactly $2^{j-\ell_i}$. So, the number of vertices at depth $j$ that are deleted before the $j$'th step is

$$\sum_{i=1}^{j-1} 2^{j-\ell_i} = 2^j \left( \sum_{i=1}^{j-1} 2^{-\ell_i} \right) \leq 2^j \sum_{i=1}^{j-1} p(i) < 2^j.$$ 

This proves that some vertex will always be available at the $j$'th step.

To show that no encoding can have expected length less than $H(X)$, suppose $X$ can be encoded in such a way that $i$ is encoded using $\ell_i$ bits. Then the expected length of the encoding is:

$$\mathbb{E}_{p(i)} [\ell_i] = \mathbb{E}_{p(i)} [\log(1/p(i))] - \mathbb{E}_{p(i)} [\log(2^{-\ell_i}/p(i))] \geq H(X) - \log \left( \mathbb{E}_{p(i)} \left[ 2^{-\ell_i}/p(i) \right] \right) = H(X) - \log \left( \sum_i 2^{-\ell_i} \right).$$

We claim that $\sum_i 2^{-\ell_i} \leq 1$. Imagine sampling a random path by starting from the root of the protocol tree, and picking one of the two children uniformly at random, until we reach a leaf. This random path hits the leaf encoding $i$ with probability $2^{-\ell_i}$. Thus, the sum $\sum_i 2^{-\ell_i}$ is the probability of hitting a leaf encoding some $i$—this is certainly at most 1. This proves that $\mathbb{E}_{p(i)} [\ell_i] \geq H(X)$.

\[\square\]

**Chain Rule and Conditional Entropy**

The entropy function has several properties that make it particularly useful. To illustrate some of the properties, we use an example from geometry. Let $S$ be a set of $n^3$ points in $\mathbb{R}^3$, and let $S_x, S_y, S_z$ denote the projections of $S$ onto the $x, y, z$ axes.

**Claim 6.2.** One of $S_x, S_y, S_z$ must have size at least $n$.

This claim has an easy proof:

$$n^3 = |S| \leq |S_x| \cdot |S_y| \cdot |S_z|,$$

and so one of the three projections must be of size $n$. However, to introduce some properties of the entropy function, we seek a proof using entropy.

Let $(X, Y, Z)$ be a uniformly random point from $S$. We prove the claim by arguing on $H(XYZ)$ and exploiting properties of entropy.

Later, we will use these properties to show that the projections onto one of the $xy, yz$ or $zx$ planes must be large; this proof is much easier using the entropy function.
The property of the entropy function we need is subadditivity. For any random variables $A, B$, we have:

$$H(AB) \leq H(A) + H(B). \quad (6.2)$$

This follows from the concavity of the log function:

$$
egin{align*}
H(A) + H(B) &- H(AB) \\
&= \mathbb{E}_{p(ab)} \left[ \log \frac{1}{p(a)} \right] + \mathbb{E}_{p(ab)} \left[ \log \frac{1}{p(b)} \right] - \mathbb{E}_{p(ab)} \left[ \log \frac{1}{p(ab)} \right] \\
&= - \mathbb{E}_{p(ab)} \left[ \log \frac{p(a \cdot p(b)}{p(ab)} \right] \\
&\geq - \log \mathbb{E}_{p(ab)} \left[ \frac{p(a \cdot p(b)}{p(ab)} \right] = - \log \sum_{a,b} p(a \cdot p(b) = - \log 1 = 0.
\end{align*}
$$

Subadditivity is a very powerful property. It can be used even when many variables are involved:

$$H(A_1A_2 \ldots A_k) \leq H(A_1) + H(A_2 \ldots A_k) \leq H(A_1) + H(A_2) + H(A_3 \ldots A_k) \leq \ldots \leq \sum_{i=1}^k H(A_i).$$

To prove Claim 6.2, recall that $(X, Y, Z)$ is a uniformly random element of $S$. So,

$$3 \log n = \log |S| = H(XYZ) \leq H(X) + H(Y) + H(Z),$$

by subadditivity, proving that one of the three terms $H(X), H(Y), H(Z)$ must be at least $\log n$. By (6.1), the projection onto the corresponding coordinate must be supported on at least $n$ points. This proves Claim 6.2.

Of course, this is not very surprising—we did not need the definition of entropy to reach this conclusion. Things become more interesting when we study the projections of $S$ to the $xy$, $yz$ and $zx$ planes. Let $S_{xy}, S_{yz}, S_{zx}$ denote these projections. Then we claim:

Claim 6.3. One of $S_{xy}, S_{yz}, S_{zx}$ must have size at least $n^2$.

To proceed, we need to define the notion of conditional entropy. For two random variables $A$ and $B$, the entropy of $B$ conditioned on $A$ is

$$H(B|A) = \mathbb{E}_{p(a,b)} \left[ \log \frac{1}{p(b|a)} \right].$$

This is the expected entropy of $B$, conditioned on the event $A = a$, where the expectation is over $a$. The chain rule for entropy states that:

$$H(AB) = H(A) + H(B|A).$$

There is a subtle notational difference between conditioning on a random variable $A$ and on an event $E$. For example, $H(B|A, E)$ is the entropy of $B$ conditioned on $A$, where the distributions of both $B$ and $A$ are conditioned on the event $E$. This claim is a special case of the Loomis-Whitney inequality.
The proof follows from Baye’s rule and linearity of expectation:

\[ H(AB) = \mathbb{E}_{p(ab)} \left( \log \frac{1}{p(ab)} \right) \]
\[ = \mathbb{E}_{p(ab)} \left( \log \frac{1}{p(a) \cdot p(b|a)} \right) \]
\[ = \mathbb{E}_{p(ab)} \left( \log \frac{1}{p(a)} + \log \frac{1}{p(b|a)} \right) = H(A) + H(B|A). \]

The chain rule shows that the entropy of \( AB \) is the entropy of \( A \) plus the entropy of \( B \) given that we know \( A \). The chain rule is an extremely useful property of entropy. It allows us to express the entropy of a collection of random variables in terms of the entropy of individual variables.

To better understand why the conditioning is necessary, consider the following example. Suppose \( A, B \) are two uniformly random bits that are always equal. Then \( H(AB) = H(A) = H(B) = 1 \), so \( H(AB) > H(A) + H(B) \). Nevertheless, \( H(B|A) = 0 \) and hence \( H(AB) = H(A) + H(B|A) < H(A) + H(B) \).

Conditional entropy satisfies another intuitive property—conditioning can only decrease entropy:

\[ H(B|A) \leq H(B) \quad (6.3) \]

Indeed, using the chain rule and subadditivity, we have

\[ H(A) + H(B|A) = H(AB) \leq H(A) + H(B), \]

which proves (6.3).
We now have enough tools to prove Claim 6.3. As before, let \((X, Y, Z)\) be a uniformly random point of \(S\). Then \(H(XYZ) = \log |S| = 3 \log n\). Repeatedly using the fact that conditioning cannot increase entropy, (6.3) we have:

\[
\begin{align*}
H(X) + H(Y|X) &\leq H(X) + H(Y|X) \\
H(X) + H(Z|XY) &\leq H(X) + H(Z|X) \\
H(Y|X) + H(Z|XY) &\leq H(Y) + H(Z|Y)
\end{align*}
\]

Adding these inequalities together and applying the chain rule gives

\[
6 \log n = 2 \cdot H(XYZ) \leq H(XY) + H(XZ) + H(YZ).
\]

Thus, one of three terms on the right hand side, must be at least \(2 \log n\). The projection onto the corresponding plane must be of size at least \(n^2\).

**Combinatorial applications**

The entropy function has found many applications in combinatorics, where it can be used to give simple proofs. Here we give a few examples that illustrate its power and versatility.

**Counting paths and cycles in a graph** Suppose we have a graph on \(n\) vertices with \(m\) edges. Since the sum of the degrees of the vertices is \(2m\), the average degree of the vertices is \(d = 2m/n\). Here we prove lower bounds on the number of paths and cycles in the graph.

Let \(X, Y, Z\) be a random path of length 2 in the graph, obtained by sampling a uniformly random edge \(X, Y\), and then a uniformly random neighbor \(Z\) of \(Y\). After fixing \(Y\), the vertex \(Z\) is independent of the choice of \(X\). Thus, we can use the chain rule to write:

\[
H(XYZ) = H(XY) + H(Z|XY) = \log m + H(Z|Y).
\]

To bound \(H(Z|Y)\), we use convexity. If \(d_v\) denotes the degree the vertex \(v\), we have:

\[
H(Z|Y) = \sum_v \frac{d_v}{2m} \cdot \log d_v
= \frac{n}{2m} \cdot \sum_v \frac{1}{n} \cdot d_v \log d_v
\geq \frac{n}{2m} \cdot d \cdot \log d = \log \frac{2m}{n}.
\]

Thus, we have

\[
H(XYZ) \geq \log \frac{2m^2}{n},
\]
proving that the support of $XYZ$ must contain at least $\frac{2m^2}{n}$ elements. Some of the elements in the support of $XYZ$ do not correspond to paths of length 2, since we could have $x = z$. However, there are at most $2m$ sequences $x, y, z$ that correspond to such a redundant choice. Moreover, each path of length 2 can be expressed in two ways as a sequence $X, Y, Z$. After correcting for these counts, we are left with at least

$$\left(\frac{2m^2}{n} - 2m\right) / 2 = m \left(\frac{m}{n} - 1\right)$$

paths of length 2.

Next, we turn to proving a lower bound on the number of 4-cycles. Sample $X, Y, Z$ as before, and then independently sample $W$ using the distribution of $Y$ conditioned on the values of $X, Z$. Then,

$$H(XYZW) = H(XYZ) + H(W | XZ)$$
$$= H(XYZ) + H(XWZ) - H(XZ)$$
$$\geq 2 \cdot H(XYZ) - 2 \log n.$$

Combining this with our bound for $H(XYZ)$, we get

$$H(XYZW) \geq \log 4 \frac{m^4}{n^4}.$$ 

This does not quite count the number of 4-cycles, because there could be some settings of $XYZW$ where two of the vertices are the same. We could have $X = Z$ or $Y = W$. However, there are at most $2n^3$ possible elements in the support of $XYZW$ where that can happen. Each cycle can be expressed in at most 4 different ways as $XYZW$. After accounting for these facts, we are left with at least

$$\left(\frac{4m^4}{n^4} - 2n^3\right) / 4 = \frac{m^4}{n^4} - \frac{n^3}{2}$$

distinct cycles.

**Bounding the girth of a graph** Another interesting parameter of a graph is its girth: the length of the smallest cycle. Suppose we have a graph with $n$ vertices and $m$ edges, and let $d = 2m/n$ denote the average degree. Here we show how to give an upper bound on the girth $g$ in terms of $n$ and $m$, using an argument based on entropy\(^5\). For simplicity, let us assume that $g$ is an odd number $g$. We also make the necessary assumption that $d > 2$, since if $d \geq 2$, there are graphs which have very large girth.

If every vertex in the graph has the same degree, then the vertices at distance $\frac{g-1}{2}$ from any fixed vertex must form a rooted

---

\(^5\) Babu and Radhakrishnan, 2010; and Alon et al., 2002

The ideas given here can easily be modified to apply when $g$ is even.

Can you think of a graph with $d = 2$ that has large girth?
(d – 1)-ary tree, or else the graph would have a cycle of length less than g. This proves that \( (d – 1)^{\frac{g-1}{2}} \leq n \) and so \( g \leq \frac{2 \log n}{\log (d-1)} + 1 \).

We shall prove that the same bound holds even when the vertices do not all have the same degree. We start by observing that we may assume that \( d_v \geq 2 \) for each vertex \( v \). Indeed, if \( v \) is a vertex with \( d_v = 1 \), then by deleting \( v \) from the graph we obtain a graph with fewer vertices, larger average degree and yet the same girth.

Let \( X = (X_0, X_1, \ldots, X_{g-1}) \) be a random path in the graph, sampled as follows. Let \( X_0, X_1 \) be a random edge, and for \( i > 1 \), let \( X_i \) be a random neighbor of \( X_{i-1} \) that is not the same as \( X_{i-2} \). This is a non-backtracking path—the path never returns along an edge that it took in the last step.

Given \( X_1 \), we see that \( X_2 \) and \( X_0 \) are identically distributed, but may be dependent. Indeed, each edge of this path is identically distributed. The chain rule gives

\[
H(X|X_0) = \sum_{i=1}^{g-1} H(X_i|X_{i-1}).
\]

If \( d_v \) denotes the degree of the vertex \( v \), we can calculate each term \( H(X_i|X_{i-1}) \) as

\[
H(X_i|X_{i-1}) = \sum_v \frac{d_v}{2m} \cdot \log(d_v - 1) = \frac{1}{d} \cdot \sum_v \frac{d_v}{n} \cdot \log(d_v - 1) \geq \frac{1}{d} \cdot d \log(d - 1) = \log(d - 1).
\]

Putting these bounds together, we get:

\[
H(X|X_0) \geq \frac{g-1}{2} \cdot \log(d - 1).
\]

Since the girth of the graph is \( g \), there can be at most \( n \) distinct paths of length \( \frac{g-1}{2} \) that begin at \( X_0 \). Thus,

\[
\log n \geq H(X|X_0) \geq \frac{g-1}{2} \cdot \log(d - 1),
\]

proving that \( g \leq \frac{2 \log n}{\log (d-1)} + 1 \).

An isoperimetric inequality for the hypercube An isoperimetric inequality identifies the shape of a given volume that has the smallest boundary. For example, if we are working with the geometry of the plane, the shape of unit area with the smallest boundary is a disc. Here we prove a similar fact, albeit in a discrete geometric space.
The n-dimensional hypercube is the graph whose vertex set is \(\{0, 1\}^n\), and whose edge set consists of pairs of vertices that disagree in exactly one coordinate. The hypercube contains \(2^n\) vertices and \(\frac{n}{2} \cdot 2^n\) edges. Given a subset \(S\) of the edges in the hypercube, we define its volume to be \(|S|\). The boundary of \(S\) consists of the set of edges that go from inside \(S\) to outside \(S\). We write \(\delta(S)\) to denote the size of the boundary of \(S\). We want to understand the sets \(S\) that minimize \(\delta(S)\) for a given value of \(|S|\).

A \(k\)-dimensional subcube of the hypercube is a subset of the vertices given by fixing \(n - k\) coordinates of the vertices to some fixed value and, allowing \(k\) of the coordinates to take any value. The volume of such a subcube is exactly \(2^k\). Each vertex of the subcube has \(n - k\) edges that leave the subcube, so the boundary of the subcube is of size \((n - k)2^k\).

We shall prove that the subcube has the smallest boundary of any set of the same volume:

**Theorem 6.4.** For any subset \(S\) of the vertices, if \(|S| \geq 2^k\), then \(\delta(S) \geq (n - k)2^k\).

**Proof.** Let \(e(S)\) be the number of edges contained in \(S\). Then we have
\[
\delta(S) = n|S| - 2e(S),
\]
so it suffices to prove that the subcube maximizes \(e(S)\) when \(|S| = 2^k\).

Let \(X\) be a uniformly random element of \(S\). For every vertex \(x \in S\) and \(y\) such that \(\{x, y\}\) is an edge of the hypercube, and \(x, y\) disagree in the \(i\)'th coordinate, we have
\[
H(X_i | X_{-i} = x_{-i}) = \begin{cases} 
1 & \text{if } \{x, y\} \subset S, \\
0 & \text{otherwise}.
\end{cases}
\]

So,
\[
\sum_{i=1}^n H(X_i | X_{-i}) = \sum_{i \in [n]} \sum_{x \in S} \frac{1}{|S|} H(X_i | X_{-i} = x_{-i}) = \frac{2e(S)}{|S|}.
\]

By subadditivity, and since conditioning does not increase the entropy,
\[
\log |S| = H(X) = \sum_{i=1}^n H(X_i | X_{<i}) \geq \sum_{i=1}^n H(X_i | X_{-i}) = \frac{2e(S)}{|S|}.
\]

This proves that \(e(S) \geq \frac{|S|\log |S|}{2}\), which is exactly the parameter achieved by a subcube. \(\square\)
Shearer’s Inequality

Shearer’s inequality is a generalization of the subadditivity of entropy. Suppose $X = X_1, \ldots, X_k$ is a collection of $n$ jointly distributed random variables, and $S \subseteq [k]$ is a set of coordinates sampled independently of $X$. Then we write $X_S$ to denote the collection of variables that correspond to $S$. One way to interpret the subadditivity of entropy (6.2) is that when $S$ is a uniformly random subset of size 1, then

$$H(X_S|S) \geq H(X).$$

Shearer’s inequality is a generalization of this fact:

**Lemma 6.5.** If $p(i \in S) \geq \epsilon$ for every $i \in [n]$, then $H(X_S|S) \geq \epsilon \cdot H(X)$.

**Proof.** Since conditioning does not increase entropy,

$$H(X_S|S) \geq \sum_{i \in S} H(X_i|X_{<i})$$

$$= \sum_{i=1}^{n} p(i \in S) \cdot H(X_i|X_{<i}) \geq \epsilon \cdot H(X).$$

We shall show how Shearer’s inequality can be used to understand families of graphs that have structured intersections. First, a warmup: suppose $\mathcal{F}$ is a family of sets of $[n]$ such that any two sets from $\mathcal{F}$ intersect. How large can such a family be? We claim

**Claim 6.6.** $|\mathcal{F}| \leq 2^{n-1}$.

**Proof.** For any set $T \in \mathcal{F}$, its complement cannot be in $\mathcal{F}$. So only half of all the sets can be in $\mathcal{F}$. □

Now, let us try to study the same kind of question for families of graphs. Let $\mathcal{G}$ be a family of graphs on $n$ vertices such that every two graphs intersect in a triangle. Such a family can be obtained by including a fixed triangle, which gives a family with $2^{\binom{n}{2}}/8$ graphs. This bound is known to be tight, but here we give a simple argument that provides a partial converse:

**Theorem 6.7.** $|\mathcal{G}| \leq 2^{\binom{n}{2}}/4$.

**Proof.** Let $G$ be a uniformly random graph from the family. $G$ can be described by a binary vector of length $\binom{n}{2}$, where each bit indicates whether a particular edge is present or not. Let $S$ be a uniformly random subset of the vertices, so that each vertex $v$ is in $S$ with probability $1/2$, independently of all other vertices.

Can you think of an example of $\mathcal{F}$ satisfying this bound?

* A cycle of length 3
* Ellis et al., 2010
* Chung et al., 1986
Let $G_S$ denote the graph obtained from $G$ by deleting all edges that go from $S$ to the complement of $S$. Since the probability that any particular edge is retained is exactly $1/2$, Shearer’s inequality gives

$$\mathbb{E}_S [H(G_S | S)] \geq H(G) / 2.$$ 

Now, since every two graphs $G, G'$ in the family intersect in a triangle, we must have that $G_S, G'_S$ must share an edge, no matter what $S$ is, because at least one of the edges of the triangle is not deleted. This means that the number of graphs of the form $G_S$ is at most half of all possible options, by Claim 6.6. Writing

$$e(S) = \binom{|S|}{2} + \binom{n - |S|}{2}$$

for the total number of edges possible in the graph $G_S$, this means that $H(G_S | S) \leq e(S) - 1$. The expected value of $\mathbb{E} [e(S)]$ is exactly $(n^2)/2$, since each edge of the complete graph is counted in the expectation with probability $1/2$. Thus, we have

$$\frac{1}{2} \cdot \binom{n}{2} = \mathbb{E}_S [e(S)] \geq \mathbb{E}_S [H(G_S | S)] + 1 \geq \frac{1}{2} \cdot H(G) + 1.$$ 

So $H(G) \leq \binom{n}{2} - 2$, which implies that $|G| \leq 2^{\binom{n}{2} - 2} = 2^{\binom{n}{2} / 4}$.

\section*{Divergence and Mutual Information}

The concepts of divergence and mutual information are closely related to the concept of entropy. They provide a toolbox that helps to understand the flow of information in a variety of situations.

The divergence between two distributions $p(x)$ and $q(x)$ is defined to be

$$\frac{p(x)}{q(x)} = \sum_x p(x) \log \frac{p(x)}{q(x)} = \mathbb{E}_{p(x)} \left[ \log \frac{p(x)}{q(x)} \right].$$

The divergence can be thought of as a measure of distance between the two distributions $p$ and $q$. In line with this intuition, we have

\textbf{Fact 6.8.} $\frac{p(x)}{q(x)} \geq 0$, and equality holds if and only if $p$ and $q$ are identical.

\textbf{Proof.} We have

$$\frac{p(x)}{q(x)} = - \sum_x p(x) \log \frac{q(x)}{p(x)} \geq - \log \sum_x p(x) \frac{q(x)}{p(x)} = \log 1 = 0.$$ 

The inequality follows from the convexity of the log function.
Since the log function is strictly convex, the inequality is a strict inequality unless \( p(x)/q(x) \) is the same for every \( x \). This can happen only when \( p(x) \) and \( q(x) \) are the same distribution.

The divergence is, however, not symmetric; it is possible that

\[
\frac{p(x)}{q(x)} \neq \frac{q(x)}{p(x)}.
\]

Moreover, the divergence can be infinite, for example if \( p \) is supported on a point that has 0 probability under \( q \). If \( X \) is an \( \ell \)-bit string, we see that:

\[
H(X) = \mathbb{E}_{p(x)} \left[ \log \left( \frac{1}{p(x)} \right) \right] = \ell - \mathbb{E}_{p(x)} \left[ \log \left( \frac{p(x)}{2^{-\ell}} \right) \right] = \ell - \frac{p(x)}{2^{-\ell}} = \ell - \frac{p(x)}{q(x)},
\]

where \( q(x) \) is the uniform distribution on \( \ell \)-bit strings. So we see that the entropy of a string is just the divergence from uniform.

We can use divergence to quantify the dependence between two random variables. If \( p(a,b) \) is a joint distribution of two random variables \( A \) and \( B \), we define the mutual information between \( A \) and \( B \) to be

\[
I(A : B) = \mathbb{E}_{p(a,b)} \left[ \log \left( \frac{p(a,b)}{p(a)p(b)} \right) \right] = \mathbb{E}_{p(a)} \left[ \log \left( \frac{p(b|a)}{p(b)} \right) \right] = H(B) - H(B|A).
\]

By Baye’s rule, we can write this as

\[
I(A : B) = \mathbb{E}_{p(a,b)} \left[ \log \left( \frac{p(b|a)}{p(b)} \right) \right] = \mathbb{E}_{p(a)} \left[ \frac{p(b|a)}{p(b)} \right] = H(B) - H(B|A).
\]

As explained in the conventions chapter, \( p(a) \) denotes the marginal distribution of \( A \).
The third expression is the expected divergence between $p(b|a)$ and $p(b)$; it measures the distance of $p(b|a)$ from $p(b)$, for an average $a$. The fourth expression says that the information measures the decrease in the entropy of $B$ when conditioning on $A$. Of course, since the information is symmetric, we have $I(A : B) = H(A) - H(A|B)$ as well.

When $A = B$, we have $I(A : B) = H(B)$. In the other extreme, $I(A : B) = 0$ exactly when $A$ and $B$ are independent. In general, the mutual information satisfies

$$0 \leq I(A : B) = H(A) - H(A|B) \leq H(A).$$

The first inequality follows from the fact that conditioning can only decrease entropy, and the second from the fact that entropy is always non-negative.

**Lower Bound for Indexing**

We have gathered enough tools to begin discussing our first lower bound on communication using information theory. We shall prove a lower bound on the indexing problem.

Suppose Alice has a uniformly random $n$ bit string $x$, and Bob is given an independent uniformly random index $i \in [n]$. The goal of the players is to compute the $i$'th bit $x_i$, but they are only allowed to execute a protocol of a specific form—the protocol must start with a message from Alice to Bob, and then Bob must output the answer. We prove that $\Omega(n)$ bits of communication are necessary, even if the parties are allowed to use randomness and make errors with some small probability.

Suppose there is a protocol for this problem where Alice sends a message $M$ that is $\ell$ bits long. Intuitively, since $M$ is only $\ell$ bits long, it can have only $\ell$ bits of information about $x$. So, we should be able to argue that $M$ carries only $\ell/n$ bits of information about a typical coordinate $x_i$. If indeed this is the case and $\ell/n \ll 1$, then $M$ should not be useful to help Bob determine $x_i$.

**Chain Rules for Divergence and Mutual Information**

We have already learnt a chain rule for the entropy function, and used it a few times. Divergence and mutual information have similar chain rules that are equally useful. Since $M$ has only $\ell$ bits, the total amount of information $M$ has about $X$ is at most $\ell$. We shall use a chain rule to argue that $M$ can convey only $\ell/n$ bits of information about $X_i$. 

By Fact 6.8.

The entropy, mutual information and divergence are all expectations over the universe of various log-ratios.
The chain rule for divergence states that for every two distributions \( p(a, b) \) and \( q(a, b) \),

\[
\frac{p(a, b)}{q(a, b)} = \frac{p(a)}{q(a)} + \mathbb{E}_{p(a)} \left[ \frac{p(b|a)}{q(b|a)} \right].
\]

The proof is a straightforward calculation using Baye’s rule:

\[
\frac{p(a, b)}{q(a, b)} = \mathbb{E}_{p(a, b)} \left[ \log \frac{p(a) \cdot p(b|a)}{q(a) \cdot q(b|a)} \right] = \mathbb{E}_{p(a, b)} \left[ \log \frac{p(a)}{q(a)} \right] + \mathbb{E}_{p(a, b)} \left[ \log \frac{p(b|a)}{q(b|a)} \right].
\]

In words, the total divergence is the sum of the divergence from the first variable, plus the expected divergence from the second variable.

Before we state the chain rule for information, it is worthwhile to think about a simple example. Suppose \( A, B, C \) are three random bits that are all equal to each other. Then \( I(AB : C) = 1 < 2 = I(A : C) + I(B : C) \). On the other hand, if \( A, B, C \) are three random bits satisfying \( A + B + C = 0 \) mod 2, we have \( I(AB : C) = 1 > 0 = I(A : C) + I(B : C) \). Nevertheless, a chain rule does hold for mutual information—we need to use the right definitions. For three random variables \( A, B \) and \( C \), define

\[
I(B : C|A) = \mathbb{E}_{p(a,b,c)} \left[ \log \frac{p(b,c|a)}{p(b|a) \cdot p(c|a)} \right].
\]

The chain rule for mutual information is then

\[
I(AB : C) = I(A : C) + I(B : C|A).
\]

This chain rule also has an intuitive meaning: the information \( AB \) give about \( C \) is the information \( A \) gives about \( C \) plus the information \( B \) gives about \( C \) when we already know \( A \). As usual, the proof is a straightforward application of Baye’s rule:

\[
I(AB : C) = \mathbb{E}_{p(a,b,c)} \left[ \log \frac{p(a, c) \cdot p(b|a,c)}{p(a)p(b|a) \cdot p(c)} \right] = I(A : C) + \mathbb{E}_{p(a,b,c)} \left[ \log \frac{p(b|a,c)}{p(b|a)} \right] = I(A : C) + I(B : C|A).
\]

Subadditivity

Unlike the entropy, the mutual information can go up under conditioning. For example, if \( A, B, C \) are three random bits subject to
$A + B + C = 0 \mod 2$, then $0 = I(A : B) < I(A : B | C) = 1$. Nevertheless, we can prove a subadditivity bound, under the assumption that the variables under consideration are independent.

**Theorem 6.9.** Let $A_1, \ldots, A_n$ be independent random variables, and $B$ be jointly distributed. Then,

$$\sum_{i=1}^{n} I(A_i : BA_{<i}) \leq I(A_1, \ldots, A_n : B) \leq H(B).$$

**Proof.** We have

$$H(B) \geq I(A_1, \ldots, A_n : B) = H(A_1, \ldots, A_n) - H(A_1, \ldots, A_n | B).$$

The first term is exactly equal to $\sum_{i=1}^{n} H(A_i)$, since $A_1, \ldots, A_n$ are independent. On the other hand, the chain rule gives that

$$H(A_1, \ldots, A_n | B) = \sum_{i=1}^{n} H(A_i | BA_{<i}).$$

So, we get

$$I(A_1, \ldots, A_n : B) \geq \sum_{i=1}^{n} H(A_i) - H(A_i | BA_{<i}) = \sum_{i=1}^{n} I(A_i : BA_{<i}),$$

as required.

Returning to the indexing problem, we can apply Theorem 6.9 to $X_1, \ldots, X_n$ and $M$, to obtain

$$\mathbb{E} [I(X_i : M)] \leq \mathbb{E} [I(X_i : MX_{<i})] = (1/n) \sum_{i=1}^{n} I(X_i : MX_{<i}) \leq \ell/n.$$

This inequality captures our intuition about the information in the message—the message does not contain much information about the bit that Bob cares about. However, the proof is not yet complete: we need to prove that if $M$ has low mutual information with $X_i$ then Bob cannot use $M$ to guess the value of $X_i$ with high probability. We need one more technical tool to prove this—Pinsker’s inequality.

**Pinsker’s Inequality**

Pinsker’s inequality bounds the statistical distance between two distributions in terms of the divergence between them.

**Lemma 6.10.**

$$\frac{p(x)}{q(x)} \geq \frac{2}{\ln 2} \cdot |p - q|^2.$$

This inequality is tight.

See the notational remarks in Probability section of the Conventions Chapter.
Proof. Let $T$ be the set that maximizes $p(T) - q(T)$, and define

$$x_T = \begin{cases} 
1 & \text{if } x \in T, \\
0 & \text{otherwise.}
\end{cases}$$

By the chain rule,

$$\frac{p(x)}{q(x)} \geq \frac{p(x_T)}{q(x_T)}.$$

Since $|p - q| = p(T) - q(T) = p(x_T = 1) - q(x_T = 1)$, it remains to prove that

$$\frac{p(x_T)}{q(x_T)} \geq 2 \ln 2 \cdot (p(x_T = 1) - q(x_T = 1))^2.$$

Let $\epsilon = p(x_T = 1)$ and $\gamma = q(x_T = 1)$. It is enough to prove that

$$\epsilon \log \frac{\epsilon}{\gamma} + (1 - \epsilon) \log \frac{1 - \epsilon}{1 - \gamma} - \frac{2}{\ln 2} \cdot (\epsilon - \gamma)^2 \geq 0 \quad (6.4)$$

is always non-negative. (6.4) is 0 when $\epsilon = \gamma$, and its derivative with respect to $\gamma$ is

$$\frac{-\epsilon}{\gamma \ln 2} + \frac{1 - \epsilon}{(1 - \gamma) \ln 2} - \frac{4(\gamma - \epsilon)}{\ln 2} = \frac{1 - \epsilon - \gamma}{\gamma(1 - \gamma) - 4} \cdot \left( \frac{1}{\gamma(1 - \gamma) - 4} \right).$$

Since $\frac{1}{\gamma(1 - \gamma)}$ is always at most 4, the derivative is non-positive when $\gamma \leq \epsilon$, and non-negative when $\gamma \geq \epsilon$. This proves that (6.4) is indeed always non-negative. □
Pinsker’s inequality implies that two variables that have low information with each other are statistically close to being independent:

**Corollary 6.11.** If \( A, B \) are random variables then on average over \( b \),
\[
p(a|b) \approx p(a),
\]
where \( \epsilon = \sqrt{\frac{\ln 2 \cdot I(A:B)}{2n}} \).

Another useful corollary is that conditioning on a low entropy random variable cannot change the distribution of many other independent random variables:

**Corollary 6.12.** Let \( A_1, \ldots, A_n \) be independent random variables, and \( B \) be jointly distributed. Let \( i \in [n] \) be uniformly random and independent of all other variables. Then on average over \( i, b, a_{<i} \),
\[
p(a_i|b, a_{<i}) \approx p(a_i),
\]
where \( \epsilon \leq \sqrt{\frac{H(B) \ln 2}{2n}} \).

**Proof.** By Theorem 6.9, \( H(B) \geq \sum_{j=1}^{n} I(A_j : BA_{<j}) \). Thus we get that for a uniformly random coordinate \( i \),
\[
\mathbb{E} [I(A_i : BA_{<i})] \leq H(B)/n.
\]
The bound then follows from Corollary 6.11.

We are finally ready to prove the lower bound we wanted. By Corollary 6.12, on average over \( m \) and \( i \),
\[
p(x_i|m) \approx p(x_i),
\]
with \( \epsilon = \sqrt{\frac{\ln 2}{2n}} \). Since \( p(x_i) \) is uniform for each \( i \), the probability that Bob makes an error in the \( i \)th coordinate must be at least \( 1/2 - \left| p(x_i|m) - p(x_i) \right| \). So the probability that Bob makes an error is at least \( 1/2 - \sqrt{\frac{\ln 2}{2n}} \), proving that at least \( \Omega(n) \) bits must be transmitted if the protocol has a small probability of error.

**Randomized Communication of Disjointness**

One of the triumphs of information theory is its ability to prove optimal lower bounds on the randomized communication complexity of functions like disjointness\(^{10} \), which we do not know how to prove any other way.

**Theorem 6.13.** Any randomized protocol that computes disjointness function with error \( 1/2 - \epsilon \) must have communication \( \Omega(\epsilon^2 n) \).

The square-root dependence is tight: If Alice sends the majority of all her bits, that bit is equal to a random coordinate with probability \( 1/2 + \Omega(1/\sqrt{n}) \). See Exercise 6.2.

Note that if Alice has a random set from a family of sets of size \( 2^{\Omega(n)} \), the lower bound for indexing would still hold. The lower bound even extends to the case that Bob knows \( x_1, \ldots, x_{i-1} \).

\(^{10}\) Kalyanasundaram and Schnitger, 1992; Razborov, 1992; Bar-Yossef et al., 2004; and Braverman and Moitra, 2013

This result is very important because many other lower bounds in various models (more in Part II) rely on Theorem 6.13.
Obstacles to Proving Theorem 6.13

The most natural way to prove lower bounds on randomized protocols is to find a hard distribution on the inputs, such that any protocol with low communication must make an error a significant fraction of the time. If we adopt this approach, we need not worry about the protocol being randomized, since any randomized protocol that works on average over the hard distribution implies the existence of a deterministic protocol as well.

This is the approach we took when we proved lower bounds on the inner-product function (Theorem 5.6), where we used the uniform distribution, and the same distribution works to understand the pointer-chasing problem (Theorem 6.16). In those cases, the uniform distribution on inputs is a hard distribution. However, the uniform distribution is not a hard distribution for disjointness: two uniformly random sets \( X, Y \) will intersect with very high probability, so the protocol can output 0 without communicating and still have very low error. In fact, it can be shown that any distribution where \( X \) and \( Y \) are independent cannot be used to prove a strong lower bound. The hard distribution, if it exists, must involve strong correlations between \( X \) and \( Y \).

Given these constraints, we shall use a natural distribution on correlated sets. \( X, Y \) will be a convex combination of two uniformly random disjoint sets, and two sets that intersect in exactly one element. Once we restrict our attention to such a distribution, we have a second challenge: the pairs of variables \( X_i, Y_i \) and \( X_j, Y_j \) are not independent for \( i \neq j \). This makes arguments involving subadditivity much harder to carry out, because subadditivity of information crucially relies on independence. The subtleties in the proof arise from circumventing these obstacles.

Proving Theorem 6.13

Given a randomized protocol with error \( 1/2 - \epsilon \), one can make the error an arbitrarily small constant by repeating the protocol \( O(1/\epsilon^2) \) times and outputting the majority outcome. This means that, it suffices to show that any protocol with error \( \frac{1}{32} \) must have communication \( \Omega(n) \).

We start by defining a hard distribution on inputs. View the sets \( X, Y \) as \( n \)-bit strings, by setting \( X_i = 1 \) if and only if \( i \in X \). Pick an index \( T \in [n] \) uniformly at random, and let \( X_T, Y_T \) be uniformly random and independent bits. For \( i \neq T \), sample \( (X_i, Y_i) \) to be one of \((0,0), (0,1), (1,0)\) with equal probability, and independent of all other pairs \((X_j, Y_j)\). The random sets \( X \) and \( Y \) intersect in at most 1 element, and they intersect with probability \( \frac{1}{8} \).
Let $M$ denote the messages of a deterministic protocol whose communication complexity is $\ell$ and probability of error is at most $1/32$. We shall prove that the protocol conveys a significant amount of information about $X_t$ or $Y_t$, when the sets are disjoint. Let $D$ denote the event that $X, Y$ are disjoint. The key claim of the proof is that the following sum of informations must be large:

$$I(X_T : M|T, X_{<T}Y_{\geq T}, D) + I(Y_T : M|T, X_{<T}Y_{\geq T}, D) \geq \Omega(1). \quad (6.5)$$

Before proving (6.5), we use it together with the subadditivity of information to prove that $\ell \geq \Omega(n)$.

We start by using the chain rule to prove:

**Lemma 6.14.** Let $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_n)$ be random variables such that the $n$ tuples $(X_1, Y_1), \ldots, (X_n, Y_n)$ are mutually independent. Let $M$ be an arbitrary random variable. Then

$$\sum_{i=1}^n I(X_i : M|X_{<i}Y_{\geq i}) \leq I(X : M|Y),$$

and

$$\sum_{i=1}^n I(Y_i : M|X_{<i}Y_{\geq i}) \leq I(Y : M|X).$$

**Proof.** Using the chain rule:

$$\sum_{i=1}^n I(X_i : M|X_{<i}Y_{\geq i}) \leq \sum_{i=1}^n I(X_i : MY_{<i}|X_{<i}Y_{\geq i})$$

$$= \sum_{i=1}^n (I(X_i : Y_{<i}|X_{<i}Y_{\geq i}) + I(X_i : M|X_{<i}Y))$$

$$= \sum_{i=1}^n I(X_i : M|X_{<i}Y) = I(X : M|Y).$$

The second bound is proved similarly. \(\square\)

We see that $X, Y, M|D$ satisfy the assumptions of Lemma 6.14. Moreover $T$ is uniform and independent of $X, Y, M$, conditioned on $D$, so Lemma 6.14 gives:

$$\frac{2\ell}{n} \geq \frac{I(X : M|YD) + I(Y : M|XD)}{n} \geq I(X_T : M|TX_{<T}Y_{\geq T}D) + I(Y_T : M|TX_{<T}Y_{\geq T}D) \geq \Omega(1),$$

since $M$ has at most $\ell$ bits by Lemma 6.14, by (6.5)

which proves that $\ell \geq \Omega(n)$.

It only remains to prove (6.5). Let $Z = (M, T, X_{<T}, Y_{\geq T})$. The intuition for the proof is to show that if the information is small, then if we sample $Z$ conditioned on the event that the sets are disjoint, then
with high probability the resulting value $z$ will have the property that $p(x_t, y_t|z)$ is close to the distribution of two uniformly random bits. However, this will lead to a high probability of errors for the protocol, because the protocol must output that the sets are either disjoint or not disjoint. For any $z$, let $\alpha_z$ be the statistical distance of $p(x_t, y_t|z)$ from uniform. Let

$$I(X_T : M | T, X < T Y \geq T, D) + I(Y_T : M | T, X \leq T, Y > T, D) = 2\gamma^4 / 3.$$ 

Let $G$ be the set of $z$ such that $\alpha_z \leq 2\gamma$. We shall use Pinsker’s inequality to prove:

**Claim 6.15.** $p(z \in G) \geq \frac{1 - 4\gamma}{4}$.

When $z \in G$, the output of the protocol is determined, but $x_t, y_t$ are close to uniform, so the probability that the protocol makes an error in this case is at least $\geq 1/4 - 2\gamma$. Thus, the probability that the protocol makes an error overall is at least

$$1/32 \geq p(\text{error}) \geq p(z \in G) \cdot p(\text{error}|z \in G) = \frac{1 - 4\gamma}{4} \cdot (1/4 - 2\gamma),$$

proving that $\gamma \geq \Omega(1)$.

To prove Claim 6.15, observe that since $X_T, Y_T$ are independent and $M$ defines a rectangle, for all $z$,

$$p(x_t | z) = p(x_t | z, y_t = 0) = p(x_t | z, y_t = 0, D).$$

Let $\beta_z$ be the statistical distance of $p(x_t | z)$ from uniform. We have

$$\frac{2}{3} \cdot I(X_T : M | T, X < T Y \geq T, Y_T = 0, D) \leq I(X_T : M | T, X \leq T Y > T, D) \leq 2\gamma^4 / 3,$$

so by convexity and Pinsker’s inequality (Lemma 6.10),

$$\mathbb{E}_{p(z|y_t=0)} [\beta_z] \leq \sqrt{\mathbb{E}_{p(z|y_t=0)} [\beta_z^2]} \leq \sqrt{\gamma^4} = \gamma^2.$$

In particular,

$$\gamma \geq p(\beta_z > \gamma | y_t = 0) \geq p(x_t = 0 | y_t = 0) \cdot p(\beta_z > \gamma | y_t = 0 = x_t) \geq \frac{p(\beta_z > \gamma | x_t = 0 = y_t)}{2},$$

so $p(\beta_z > 0 | x_t = 0 = y_t) \leq 2\gamma$. A symmetric argument proves that the probability that the statistical distance of $p(y_t | z)$ from uniform exceeds $\gamma$ is at most $2\gamma$, conditioned on $x_t = 0 = y_t$. Thus by the union bound, the probability that either $p(x_t | z)$ or $p(y_t | z)$ has distance from uniform $\gamma$ is at most $4\gamma$. Since $p(x_t, y_t | z)$ is a product distribution, its distance from uniform is at most $2\gamma$ in this case. Claim 6.15 then follows.

Intuitively, if $\gamma$ is small, then $x_t, y_t$ must be close to uniform for most fixings of $m$. This leads to a high probability of error in the protocol.
Lower Bound for Number of Rounds

Are interactive protocols more powerful than protocols that do not have much interaction?\(^{11}\) Here we show that a protocol with many rounds can have significantly less communication than a protocol with fewer rounds.

**Randomized Pointer-Chasing**

The pointer chasing problem is a natural problem where having many rounds of communication is very useful. Alice is given \(x \in [n]^n\) and Bob is given \(y \in [n]^n\). The vectors \(x, y\) define a bipartite directed graph with \(2n\) vertices, in which each vertex has exactly one edge coming out of it. The edges emanating from the vertices on the left are specified by \(x\), and the edges from the right are specified by \(y\). An example is shown in Figure 6.8. The graph defines a path \(z(0), z(1), z(2), \ldots\) by setting \(z(0) = 1, z(1) = xz(0), z(2) = yz(1), \) and so on. Namely, \(z(i)\) is the vertex obtained by following \(i\) edges in the graph starting at the vertex \(z(0) = 1\). The goal of the parties is to output whether or not \(z(k)\) is even.

There is an obvious deterministic protocol that takes \(k\) rounds and \(k \lceil \log n \rceil\) bits of communication—in round \(i\), the relevant player announces the value of \(z_i\).

There is also a randomized protocol with \(k - 1\) rounds and \(O((k + n/k) \log n)\) bits of communication.\(^{12}\) In the first step, Alice and Bob use shared randomness to pick \(10n/k\) vertices in the graph and announce the edges that originate at these vertices. Alice and Bob then continue to use the deterministic protocol, but do not communicate if one of the edges they need has already been announced. This protocol will have \(< k\) rounds with high probability.

We shall prove that any randomized or deterministic protocol with \(k - 1\) rounds must have much more communication than the \(k\)-round protocol. Actually we will prove that it is hard to compute any information about \(z_k\) in at most \(k - 1\) rounds of communication.

The key idea here is quite similar to the lower bound for the indexing problem. Assume \(x, y\) are chosen uniformly at random and independently, and suppose the protocol has small communication complexity and at most \(k - 1\) rounds. We argue, by induction, that \(z(k)\) is “close to uniform” even after conditioning on the messages in the first \(k - 1\) rounds of the protocol.

**Theorem 6.16.** Any randomized \((k - 1)\)-round protocol for the \(k\)-step pointer chasing problem that is correct with probability \(1/2 + \epsilon\) requires \(\frac{\epsilon^2 n}{k^2} - k \log n\) bits of communication.

---

\(^{11}\) Yao, 1983; Duris et al., 1987; Halstenberg and Reischuk, 1993; and Nisan and Wigderson, 1993

Since the information about the number of rounds is lost once we move to viewing a protocol as a partition into rectangles, it seems hard to prove a separation between a few rounds and many rounds using the techniques we have seen before. A protocol with low communication will have a large rectangle, so we cannot bound the size of rectangles to get a separation between interactive protocols and non-interactive protocols.

---

\(^{12}\) Nisan and Wigderson, 1993

Figure 6.8: An example of an input to pointer chasing, with \(n = 8, k = 5\).
The probability that none of the announced values help to save a round of communication is exponentially small in \( k \), as long as \( \Omega(k) \) of the values \( z_i \) are distinct. If the values are not distinct, then less than \( k \) rounds of communication are required anyway.

Try to prove this basic fact.

**Proof.** Let \( X, Y \) be distributed uniformly and independently. Let \( M_t \) be the message sent at the \( t \)’th round of some protocol. Let \( R_{k-1} \) denote the random variable

\[
R_{k-1} = (M_1, \ldots, M_{k-1}, Z(1), \ldots, Z(k-1)).
\]

We shall prove inductively that on average over \( r_{k-1} \), the distribution \( p(z(k)|r_{k-1}) \) is \( (k \cdot \epsilon) \)-close to uniform with

\[
\epsilon = \sqrt{\frac{\ell + k \log n}{n}}.
\]

When \( k = 1 \), the statement is trivial. Suppose \( k \geq 2 \) and \( k \) is even. The proof is similar when \( k \) is odd. We shall repeatedly use the following fact about statistical distance: If \( U, V \) are independent and \( p(u) \approx p(v) \), then \( p(g(u)) \approx p(g(v)) \), for any function \( g \).

Since \( R_{k-2}, M_{k-1} \) contains at most \( \ell + k \log n \) bits of information, Corollary 6.12 implies that if \( i \) is uniformly random in \([n]\) and independent of all other variables, then on average over \( i, r_{k-2}, m_{k-1}, \)

\[
p(y_i|r_{k-2}) \approx p(y_i|r_{k-2}).
\]

We need to bound the error in the \( k' \)th step by \( k\epsilon \). There are two cases to consider:

**Alice sends the message \( m_{k-1} \).** In this case, \( p(y_i|r_{k-1}) = p(y_i|r_{k-2}) \), since after fixing \( r_{k-2} \), we know that \( Y_i \) is independent of \( M_{k-1} \) and \( Z(k-1) \). By induction,

\[
p(z(k-1)|r_{k-2}) \approx p(i).
\]

After fixing \( r_{k-2} \), if we set \( u = i, v = z(k-1) \), and \( g \) to be the function with \( g(j) = y_j \), then by induction \( p(u) \approx p(v) \), and so

\[
p(g(u)) \approx p(g(v))
\]

implies that

\[
p(y_{z(k-1)}|z(k-1), r_{k-2}) \approx p(y_i|i, r_{k-2}).
\]

So we have:

\[
p(z(k)|r_{k-1}) = p(y_{z(k-1)}|z(k-1), r_{k-2}) \approx p(y_i|i, r_{k-2}).
\]

Combining this with (6.6) gives that \( p(z(k)|r_{k-1}) \approx p(y_i) \).
Bob sends the message $m_{k-1}$. In this case, after fixing $R_{k-2}$, we know that $Z(k-1)$ is independent of $Y$, and therefore also of $M_{k-1}$, which is a function of $Y$. So,

$$p(z(k-1)|r_{k-2}) = p(z(k-1)|m_{k-1}, r_{k-2}).$$

By induction,

$$p(z(k-1)|m_{k-1}, r_{k-2})^{(k-1)e} \approx p(i).$$

So, on average over $i, r_{k-2}$ and $m_{k-1}$. After fixing $r_{k-2}$, if we set $u = i, v = z(k-1)$, and $g$ to be the function with $g(j) = y_j$, then by induction $p(u)^{(k-1)e} \approx p(v)$, and so

$$p(g(u))^{(k-1)e} \approx p(g(v))$$

implies that

$$p(y_{z(k-1)}|z(k-1), r_{k-2})^{(k-1)e} \approx p(y_i|i, r_{k-2}).$$

This proves that

$$p(z(k)|r_{k-1}) = p(y_{z(k-1)}|z(k-1), m_{k-1}, r_{k-2})^{(k-1)e} \approx p(y_i|i, m_{k-1}, r_{k-2}).$$

Combining this with (6.6) gives that $p(z(k)|r_{k-1}) \approx p(y_i)$.

Both of these bounds imply that $p(z_k|r_{k-1})$ is $(k \cdot e)$-close to uniform, as required.

---

**Stronger Bounds for Deterministic Protocols**

Similar intuitions can be used to show that the deterministic communication of the pointer-chasing problem is $\Omega(n)$ if fewer than $k$ rounds of communication are used. Since the proof is too technical, we omit it from this text.

**Theorem 6.17.** Any $k-1$ round deterministic protocol that computes the $k$-step pointer-chasing problem requires $\frac{n}{16} - k$ bits of communication.

---

**Exercise 6.1**

Show that for any two joint distributions $p(x, y), q(x, y)$ with same support, we have

$$\mathbb{E}_{p(y)} \left[ \frac{p(x|y)}{p(x)} \right] \leq \mathbb{E}_{p(y)} \left[ \frac{p(x|y)}{q(x)} \right].$$

---

Nisan and Wigderson, 1993
Exercise 6.2

Suppose \(n\) is odd, and \(x \in \{0, 1\}^n\) is sampled uniformly at random from the set of strings that have more 1’s than 0’s. Use Pinsker’s inequality to show that the expected number of 1’s in \(x\) is at most \(n/2 + O(\sqrt{n})\).

Exercise 6.3

Let \(X\) be a random variable supported on \([n]\) and \(g : [n] \to [n]\) be a function. Prove that

\[
\Pr[X \neq g(X)] \geq \frac{H(X|g(X)) - 1}{\log n}.
\]

Use this bound to show that if Alice has a uniformly random vector \(y \in [n]^n\), Bob has uniformly random and independent input \(i \in [n]\), and Alice sends Bob a message \(M\) with that contains \(\ell\) bits, the probability that Bob guesses \(y_i\) is at most \(\frac{1+\ell/n}{\log n}\).

Exercise 6.4

Let \(G\) be a family of graphs on \(n\) vertices, such that in each graph in the family, every two vertices share a clique on \(r\) vertices. Show that the number of graphs in the family is at most \(2^{\binom{n}{2}}/2^{r-1}\).

*Hint:* Partition the graph into \(r\) parts uniformly at random and throw away all edges that do not stay within a part. Analyze the entropy of the resulting distribution on graphs.

Exercise 6.5

Prove the data processing inequality: if \(A \rightarrow B \rightarrow C\) then \(I(A : B) \geq I(A : C)\).
Is there a way to define the information of a protocol in analogy with Shannon’s definition of the entropy of a single message? Extending Shannon’s ideas, we would like to measure the information contained in all the messages of the protocol. In this chapter, we explore how to do this for 2-party communication protocols. Somewhat surprisingly, the definitions lead to several results in communication complexity that do not concern information at all, though these definitions were motivated by understanding combinatorial problems.

Suppose we are working in the distributional setting, where the inputs $X, Y$ to Alice and Bob are sampled from some known distribution $\mu$, and the protocol is randomized.

- Consider a protocol where all the messages of the protocol are 0, not matter what the inputs are. Then messages of the protocol are known ahead of time, and Alice and Bob may as well not send them. These messages do not convey any information.

- Suppose $X, Y$ are independent uniformly random $n$ bit strings. Alice sends $X$ as her first message and Bob sends $Y$ in response. In this case, the protocol cannot be simulated with less than $2n$ bits.

- Suppose $X, Y$ are independent uniformly random $n$ bit strings. In the first message, Alice privately samples $k$ uniformly random bits and sends them to Bob, and follows this message by sending $X$. This protocol can be simulated by a randomized communication protocol with communication $n$: Alice and Bob can use shared randomness to sample the $k$ bit string, so that they do not have to send it.

- Suppose $X, Y$ are independent uniformly random $n$ bit strings. Alice uses private randomness to sample a uniformly random subset $T \subseteq [n]$ of size $k$. Alice sends $n$ bits to Bob, where the $i$th bit is $X_i$ if $i \notin T$, and $1 - X_i$ otherwise. One can simulate this protocol with less than $n$ bits, at least in expectation.
Bob can use public randomness to sample a set \( S \subseteq [n] \) of size \( k \), as well a uniformly random \( n \)-bit string \( R \). Alice computes the number of coordinates \( i \in S \) such that \( R_i \neq X_i \). If there are \( t \) such coordinates, she samples a uniformly random subset \( T' \subseteq [n] - S \) of size \( k - t \). Setting \( M_i = R_i \) for all \( i \in S \), \( M_i = 1 - X_i \) for all \( i \in T' \) and \( M_i = X_i \) for all remaining \( i \), Alice sends the \( n - k \) bits of \( M \) that Bob does not already know. For any fixed value of \( X \), the string \( M \) is identically distributed to how it was in the original protocol, so the simulation succeeds with communication \( n - k \).

• Suppose \( X, Y \) are uniformly random strings that are always equal. Suppose Alice sends \( X \) to Bob in the first message. Then this message can be simulated with 0 communication, since Bob already knows \( X \).

These examples illustrate the difficulties with defining the information of communication protocols. Indeed, motivated by the applications to communication complexity, there are two natural ways to define the information of a protocol. Let \( R \) denote the public randomness of the protocol, and let \( M \) denote the messages that result from executing the protocol. The external information\(^1\) of the protocol is defined to be

\[
I(XY : M|R).
\]

The external information measures the amount of information about the inputs that an external observer may learn about \( X, Y \) from messages and public randomness of the protocol. The second definition is called the internal information\(^2\) of the protocol. It is defined to be

\[
I(X : M|YR) + I(Y : M|XR).
\]

The internal information measures the amount of information that Alice and Bob learn about each other’s inputs from the messages and public randomness of the protocol.

The external information is never larger than the internal information, and the two quantities are equal when \( X, Y \) are independent. To see this, let us apply the chain rule to express the internal information as:

\[
I(X : M|YR) + I(Y : M|XR) = \sum_i I(X : M_i|YRM_{<i}) + I(Y : M_i|XRM_{<i}),
\]

where here \( M_1, M_2, \ldots \) are the bits of \( M \). The definition of communication protocols ensures that the first \( i - 1 \) bits \( M_{<i} \) determine whether
Alice or Bob sends the next bit of the protocol. If Alice sends the next bit, then
\[ I(Y : M_i | XRm_{<i}) = 0, \]
because \( M_i \) is determined by the variables \( XRm_{<i} \). Similarly, if Bob sends the next bit in the protocol, then
\[ I(X : M_i | YRm_{<i}) = 0. \]
Moreover, if Alice sends the next bit, then by the chain rule, we have
\[
I(X : M_i | YRm_{<i}) \\
\leq I(X : M_i | YRm_{<i}) + I(Y : M_i | Rm_{<i}) \\
= I(XY : M_i | Rm_{<i}),
\]
where the inequality is an equality when \( X, Y \) are independent of each other, because in this case \( Y \) is independent of \( M_i \) after fixing \( R, m_{<i} \). Similarly, if Bob sends the next bit, we have
\[
I(Y : M_i | XRm_{<i}) \leq I(XY : M_i | Rm_{<i}),
\]
and the inequality is an equality when \( X, Y \) are independent. Putting all of these observations together, we get that the internal information can be bounded
\[
I(X : M | YR) + I(Y : M | XR) \\
= \sum_i I(X : M_i | YRM_{<i}) + I(Y : M_i | XRM_{<i}) \\
\leq \sum_i I(XY : M_i | RM_{<i}) \\
= I(XY : M | R),
\]
so the internal information never exceeds the external information. The two quantities are equal when \( X, Y \) are independent.

What we are really after is an analogy to Theorem 6.1—we want to show that information characterizes communication. Such a statement would be immensely useful, because the quantities defining information are much easier to work with than communication complexity.

**Correlated Sampling**

Suppose we are given a protocol whose communication complexity is very large, but its internal information is very close to 0. In this case, the protocol teaches Alice and Bob almost nothing about each others inputs, so they should be able to simulate its execution without communicating, and this is what we show here. We use the technique of correlated sampling\(^3\) to achieve this:

The same argument proves that \( I(X : M | YR) \) is at most the expected number of bits sent by Alice in the protocol, and \( I(Y : M | XR) \) is at most the expected number of bits sent by Bob in the protocol.
Lemma 7.1. There is a protocol for Alice and Bob, who are each given distributions $p(m), q(m)$ to use public randomness and no communication in such a way that Alice samples $M^A$ distributed according to $p(m)$, Bob samples $M^B$ distributed according to $q(m)$ and the probability that $M^A \neq M^B$ is at most $2|p - q|$.

Proof. Alice and Bob will use public randomness to sample a sequence $(m_1, \rho_1), (m_2, \rho_2), \ldots$, where $m_i$ is a uniformly random element from the support of $m$, and $\rho_i$ is uniformly random from $[0, 1]$. Alice will set $m = m_i$, where $i$ is the minimum number for which $\rho_i \leq p(m_i)$. Similarly, Bob will set $m' = m_j$ where $j$ is the minimum number such that $\rho_j < p(m_j)$.

Let $r(m^A, m^B)$ denote the joint distribution of the outputs of Alice and Bob. Let $E$ denote the event that Alice sets $i = 1$. Then we claim that $r(M^A = m_A | E) = p(M = m^A)$. Indeed, by the definition of the process, we have $r(M^A = m^A | E) = r(M^A = m^A)$. Since

$$r(M^A = m^A) = r(E)r(M^A = m^A | E) + (1 - r(E))r(M^A = m^A | \neg E),$$

we have $r(M^A = m^A) = r(M^A = m^A | E)$.

This implies that $r(M^A = m^A) = p(M = m^A)$, and $r(M^B = m^B) = q(M = m^B)$. Let $B$ denote the event that either $q(m_i) < \rho_i < p(m_i) \quad \text{or} \quad p(m_i) < \rho_j < q(m_j)$. The event $B$ must happen if $M \neq M'$. Let $F$ denote the event that either $i = 1$ or $j = 1$. Then exactly as before, we

Figure 7.1: The correlated sampling procedure.

The expected values of $i$ and $j$ are proportional to the size of the universe, so the time required to carry out this procedure is also proportional to the size of the universe.

Figure 7.2: An example of the sampling procedure. $(m_4, \rho_4)$ is selected in this case.
have \( r(B|\neg F) = r(B) \), and so \( r(B) = r(B|F) \). But we have

\[
    r(B|F) = \frac{\sum_m |p(m) - q(m)|}{\sum_m \max\{p(m), q(m)\}}
\]

\[
    \leq \frac{\sum_m |p(m) - q(m)|}{\sum_m p(m) + |p(m) - q(m)|} \leq \sum_m |p(m) - q(m)|,
\]

since \( \sum_m p(m) = 1 \)

as required.

\[\square\]

**Compressing a Single Round of Communication**

**External Information**

Suppose we would just like to compress the first message in a protocol down to its external information. If the message \( M \) is sent by Alice, who has the input \( X \), and Bob has the input \( Y \), then the external information can be expressed as

\[
    I(XY : M) = I(X : M) + I(Y : M|X)
\]

\[
    = I(X : M).
\]

In analogy with Theorem 6.1, we prove that there is a way to simulate\(^4\) the sending of the message \( M \) using \( I(X : M) + O(\log I(X : M)) \) bits of communication in expectation. The theorem follows from the following stronger fact:

**Theorem 7.2.** Suppose Alice knows two distributions \( p, q \), and Bob knows \( q \). There is a protocol for Alice and Bob to sample an element according to \( p \) using

\[
    \frac{p(m)}{q(m)} + 2 \log \left( \frac{p(m)}{q(m)} \right) + O(1)
\]

bits of communication in expectation.

As a corollary, we get

**Corollary 7.3.** Alice and Bob can use public randomness to simulate sending \( M \) with expected communication \( I(X : M) + 2 \log I(X : M) + O(1) \).

The protocol we use is inspired by the correlated sampling idea. The public random tape will consist of a sequence of samples \( (m_1, \rho_1), (m_2, \rho_2), \ldots \), where each \( m_i \) is a uniformly random element from the support of \( m \), and \( \rho_i \) is a uniformly random number from \([0, 1]\).

Given this public randomness, Alice finds the minimum index \( r \) such that \( p(M = m_r) \geq \rho_r \). The value \( m_r \) has exactly the right distribution. Unfortunately, communicating \( r \) can be too expensive,
so Alice cannot simply send $r$ to Bob. Instead, Alice computes the positive integer

$$T = \left\lceil \frac{\rho_r}{q(M = m_r)} \right\rceil,$$

and sends $T$ to Bob. Given $T$, Alice and Bob both compute the set

$$S_T = \left\{ j : T = \left\lceil \frac{\rho_j}{q(M = m_j)} \right\rceil \right\}.$$  

Alice sends Bob the number $K$ for which $r$ is the $K$'th smallest element of $S_T$.

We have already shown in Section 7 that the sample $m_r$ has the right distribution. To analyze the expected communication of the protocol, we need two basic claims. The first claim, whose proof we sketch in the margin, is used to encode the integers sent in the protocol.

**Claim 7.4.** One can encode all positive integers in such a way that at most $\log z + 2 \log \log z + O(1)$ bits are used to encode the integer $z$.

To argue that the expected length of $T$ is small, we need the following claim:

**Claim 7.5.** For any two distribution $p(m)$, $q(m)$, the contribution of the terms with $p(m) < q(m)$ to the divergence is at least $-1$:

$$\sum_{m : p(m) < q(m)} p(m) \log \frac{p(m)}{q(m)} > -1.$$  

Figure 7.4: The sampling procedure of Theorem 7.2. Here $T$ is 3 and the sampled point is the 3'rd point of $S_T$.

Figure 7.3: Compressing a single round of communication to its internal information.

Proof of Claim 7.4: A naive encoding would have Alice send a bit to indicate whether there is another bit left to send in the encoding, and then send the bit of data. This would take $2\lceil \log z \rceil + O(1)$ bits. To get a better bound, first send the integer $\lceil \log z \rceil$ using the naive encoding, and then send $\lceil \log z \rceil$ more bits to encode $z$.  

Input: Alice knows $p(m), q(m)$,  
Bob knows $q(m)$.  
Output: $m$ distributed according to $p(m)$.

P. Rand: $(m_1, \rho_1), \ldots$, uniformly and independently from the universe and $[0, 1]$.

Alice sets $r$ to be the minimum index such that $p(M = m_r) > \rho_r$;  
Alice computes $T = \left\lceil \frac{\rho_r}{q(M = m_r)} \right\rceil$;  
and sets $K$ to be the smallest integer such that $r$ is the $K$'th element of $\left\{ j : T = \left\lceil \frac{\rho_j}{q(M = m_j)} \right\rceil \right\}$.

Alice sends Bob $T, K$;
Now, to bound the expected number of bits required to transmit $T$, observe that by Claim 7.4, this is at most

$$\mathbb{E} \left[ \log T + 2 \log \log T + O(1) \right] \leq \mathbb{E} \left[ \log T \right] + 2 \log \mathbb{E} \left[ \log T \right] + O(1),$$

where the inequality follows from Jensen’s inequality. By Claim 7.5, we can bound

$$\mathbb{E} \left[ \log T \right] \leq \sum_m p(m) \log \left( \frac{p(m)}{q(m)} \right)$$

$$\leq \sum_{m: p(m) > q(m)} p(m) \left( \log \frac{p(m)}{q(m)} + 1 \right)$$

$$\leq \frac{p(m)}{q(m)} \sum_{m: p(m) < q(m)} p(m) \log \frac{p(m)}{q(m)} + 1$$

$$\leq \frac{p(m)}{q(m)} + 2,$$

and so the expected number of bits used to transmit $T$ is at most

$$\frac{p(m)}{q(m)} + 2 \log \left( \frac{p(m)}{q(m)} \right) + O(1).$$

Proof of Claim 7.5: Let $E$ denote the subset of $m$’s for which $p(m) < q(m)$. Then we have

$$\sum_{m: p(m) < q(m)} p(m) \log \frac{p(m)}{q(m)}$$

$$\leq -p(E) \cdot \log \frac{p(E)}{q(E)}$$

$$\geq p(E) \cdot \log \frac{q(E)}{p(E)}$$

$$\geq -p(E) \cdot \log \frac{q(E)}{p(E)}$$

$$\geq p(E) \cdot \log \frac{q(E)}{p(E)}.$$

For $0 \leq x \leq 1$, $x \log x$ is maximized when its derivative is 0: $\log e + \log x = 0$. So the maximum is attained at $x = 1/e$, proving that $p(E) \log p(E) \geq -\frac{\log e}{e} > 1$.

It only remains to bound the number of bits required to transmit $K$. Since Jensen’s inequality proves that $\mathbb{E} \left[ \log K \right] \leq \log \mathbb{E} \left[ K \right]$, we shall start by bounding $\mathbb{E} \left[ K \right]$. Consider the event $A$, defined to be $p(M = m_1) \geq \rho_1$. When $A$ happens, $K = 1$. Conditioned on the event that $A$ does not happen, $T$ is independent of $(m_1, \rho_1)$. Define the random variable

$$Z = \begin{cases} 1 & \text{if } 1 \in S_T, \\ 0 & \text{otherwise.} \end{cases}$$

Then we have

$$\mathbb{E} \left[ K \right] = \mathbb{P}[A] + \mathbb{P}[\neg A] \cdot (\mathbb{E} \left[ K \right] + \mathbb{E} \left[ Z | \neg A \right])$$

$$\Rightarrow \mathbb{E} \left[ K \right] = 1 + \frac{\mathbb{P}[\neg A] \cdot \mathbb{E} \left[ Z | \neg A \right]}{\mathbb{P}[A]}.$$

Suppose the space of all $m$’s is of size $u$. Then we can compute

$$\mathbb{P}[A] = \frac{1}{u} \sum_m p(m) = 1/u,$$
and
\[
\mathbb{E}[Z|\neg A] = \Pr[i \in S_T|\neg A] \\
\leq \frac{(1/u) \sum_m Tp(m) - (T-1)p(m)}{(1/u) \sum_m (1-p(m))} \\
\leq \frac{(1/u) \sum_m p(m)}{(1/u) \sum_m (1-p(m))} \\
= \frac{1}{(1/u)(u-1)} = \frac{1}{u-1}.
\]

Thus we get
\[
\mathbb{E}[K] \leq 1 + \frac{(1-1/u)/(u-1)}{1/u} = 2.
\]

So the expected number of bits required to transmit \( K \) is a constant.

**Internal Information**

Now suppose we wish to compress a single message sent from Alice to Bob down to its internal information. This is strictly harder than the problem for external information—when \( Y \) is a constant, the two problems are the same.

**Theorem 7.6.** Suppose Alice knows two distributions \( p \), and Bob knows \( q \). For every \( \epsilon \), there is a protocol for Alice to sample an element according to the distribution \( p \) while communicating an expected
\[
\frac{p(m)}{q(m)} + O\left(\sqrt{\frac{p(m)}{q(m)}}\right) + \log(1/\epsilon) + O(1)
\]
bits, such that Bob also computes the same sample, except with probability \( \epsilon \).

As a corollary, we get

**Corollary 7.7.** Alice and Bob can use public randomness to simulate sending \( M \) with expected communication \( I(X:M|Y) + O(\sqrt{I(X:M|Y)}) + \log(1/\epsilon) \).

We shall use very similar ideas to obtain a protocol as in the previous section. However, our simulating protocol will be interactive, and there will be a small possibility of committing an error.

As in the previous section, Alice and Bob will use public randomness to sample a sequence of points \((m_1, \rho_1), (m_2, \rho_2), \ldots\), where each \( m_i \) is a uniformly random element of the support, and \( \rho_i \) is a uniformly random number in \([0,1]\). As before, Alice picks the smallest index \( r \) such that \( p(M = m_r) > \rho_r \). We would really like to compute \( \left\lceil \frac{\rho_r}{q(M=m_r)} \right\rceil \) with small communication. Unfortunately, Alice does not
know q, so she cannot compute this ratio without interacting with Bob. Instead, Alice and Bob will try to guess the ratio. To do so, they will gradually increase a threshold T until it is larger than this ratio. They will then use hashing to find r.

For each index i, let \( h(i) = h(i)_1, h(i)_2, \ldots \) be an infinite sequence of uniformly random bits, sampled publicly. \( h(i) \) is a hash function that Alice and Bob will try to use to quickly agree on the value of r.

The protocol will proceed in rounds. In round \( k \), Alice and Bob set \( T = 2^k \), and Bob computes the set

\[
Q_T = \left\{ j : T \geq \frac{\rho_j}{q(M = m_j)} \right\}.
\]

In round \( k \), Alice will send Bob all the bits of \( h(r)_{\leq k^2 + \log(1/\epsilon)} \) that she has not already sent him. For each \( i = 1, 2, \ldots, T \), and \( j = 1, 2, \ldots, k^2 \), Bob will compute the value

\[
g(i, j) = \min\{ \ell \in Q_{2^\ell} : h(\ell)_{\leq j} = h(r)_{\leq j} \}.
\]

\( g(i, j) \) is Bob’s best guess for the index of \( Q_{2^\ell} \) that is consistent with the first \( j \) bits of \( h(r) \) that he sees. If there is any index \( s \leq k \) such that \( g(s, k^2 + \log(1/\epsilon)) = g(s, (k - 1)^2) \), then Bob stops the protocol and outputs \( g(s, (k - 1)^2) \) for the smallest such index \( s \). If there is no such index, Bob sends Alice a bit to indicate that the protocol should continue, and the parties begin the next round.

Intuitively, if \( k \) is large enough so that \( Q_T \) contains \( r \), then all indices of \( Q_T \) that are less than \( r \) will eventually become inconsistent with \( h(r) \). If \( T \) is smaller, then the probability that any index will remain consistent with the hashes for many hashes is small.
First, let us analyze the probability that the protocol makes an error. The protocol outputs \( g_{s,(k-1)^2} \neq r \) only if we have \( g_{s,k^2 + \log(1/\epsilon)} = g_{s,(k-1)^2} \neq r \). The probability of this event is at most
\[
2^{-(k^2 + \log(1/\epsilon) - (k-1)^2)} \leq 2^{-2k - 1 - \log(1/\epsilon)}.
\]
Thus the probability of an error is at most
\[
\sum_{k=1}^{\infty} k \cdot 2^{-2k - 1 - \log(1/\epsilon)} \leq \epsilon.
\]

To analyze the expected communication of the protocol, let \( K \) be the smallest non-negative integer such that \( 2K^2 \geq \frac{\rho_r}{q(M=m')} \), and let \( R \) denote the index picked by Alice. Let \( H \) be the minimum integer such that \( g(K,H) = R \) and \( H \geq K^2 \). We shall show that the expected value of \( H \) is small.

**Claim 7.8.** \( \mathbb{E}[H] \leq \frac{p(m)}{q(m)} + 3 \sqrt{\frac{p(m)}{q(m)}} + O(1) \).

Now if \( (k-1)^2 + \log(1/\epsilon) \geq H + \log(1/\epsilon) \), then the protocol will certainly terminate by round \( k \), since after this point, we have \( g(K,k^2 + \log(1/\epsilon)) = r = g(K,(k-1)^2) \). Since the first value of \( k \) satisfying this inequality is \( \lceil \sqrt{H} \rceil + 1 \), the expected communication of the protocol is at most
\[
\mathbb{E}

\left[

\left(\left\lceil \sqrt{H} \right\rceil + 1\right)^2 + \left(\left\lceil \sqrt{H} \right\rceil + 1\right) + \log(1/\epsilon)

\right]

\leq \mathbb{E}[H] + 5 \sqrt{\mathbb{E}[H] + \log(1/\epsilon)} + O(1)

\leq \frac{p(m)}{q(m)} + O \left( \sqrt{\frac{p(m)}{q(m)}} \right) + \log(1/\epsilon) + O(1).
\]

**Proof of Claim 7.8.** We start by proving that
\[
\mathbb{E}[H|K,R] \leq 2 + \log \left( \frac{2K^2 R}{u-1} \right).
\]
We shall prove by induction that if \( L \) elements of \( Q_{2K^2} \) precede \( R \), then the expected number of hashes is needed is at most \( 2 + \log L \). Indeed, when \( L = 1 \), the expected number of hashes needed is 2—it is the same as the number of coin tosses needed to see a heads with a fair coin. For larger \( L \), let \( L' \) denote the number of elements that survive the first hash. Then we have:
\[
\mathbb{E}[H|L] = 1 + \mathbb{E}\left[ \log L' + 2|L \right]

\leq 3 + \log \mathbb{E}\left[ L'|L \right]

\leq 3 + \log(L/2) = \log L + 2.
\]

Here we use the identity:
\[
\sum_{i=1}^{n} z^{i-1} = \frac{(n+2)z^{n+1}}{1-z} + \frac{1-z^{n+2}}{(1-z)^2},
\]
which can be proved by taking the derivative of the identity
\[
\sum_{i=1}^{n} z^i = \frac{1-z^{n+2}}{1-z}.
\]
The expected number of elements that precede $R$ is exactly $R \frac{2K^2}{u-1}$. The claim follows by the convexity of the log function.

The expected number of hashes needed is thus at most $K^2 + \log \frac{R}{u-1}$.

To bound the first term, observe that if $K' = \max \left\{ \log \frac{\rho_r}{q(m=M,m_r)}, 0 \right\}$, then

$$K^2 = \left[ \sqrt{K'} \right]^2 \leq \left( \sqrt{K'} + 1 \right)^2 = K' + 2 \cdot \sqrt{K'} + 1,$$

so by Jensen’s inequality, we can bound

$$\mathbb{E} \left[ K^2 \right] \leq \mathbb{E} \left[ K' \right] + 2 \sqrt{\mathbb{E} \left[ K' \right]} + 1.$$

To bound $\mathbb{E} \left[ K' \right]$, we use Claim 7.5:

$$\mathbb{E} \left[ K' \right] = \sum_{m : p(m) > q(m)} p(m) \log \frac{p(m)}{q(m)}$$

$$= \frac{p(m)}{q(m)} - \sum_{m : p(m) < q(m)} p(m) \log \frac{p(m)}{q(m)} \leq \frac{p(m)}{q(m)} + 1.$$

This gives:

$$\mathbb{E} \left[ K^2 \right] \leq \frac{p(m)}{q(m)} + 2 \sqrt{\frac{p(m)}{q(m)}} + 3.$$

To bound the second term, observe that

$$\mathbb{E} \left[ \log \frac{R}{u-1} \right] \leq \log \frac{\mathbb{E} \left[ R \right]}{u-1} \leq 2,$$

since we have

$$\mathbb{E} \left[ R \right] = (1/u) \sum_m p(m) + (1 - 1/u)(\mathbb{E} \left[ R \right] + 1),$$

so $\mathbb{E} \left[ R \right] = u$. \hfill \Box

**Compressing Entire Protocols with Low Internal Information**

In this section, we describe how to compress any protocol with low internal information\(^5\). Suppose we are given inputs $X, Y$ sampled according to some known distribution, and a protocol $\pi$ with public randomness $R$ and messages $M$. Suppose the communication complexity of the protocol is $C$ and its internal information is

$$I = I(X : M|YR) + I(Y : M|XR).$$

We shall prove:

\(^5\) Barak et al., 2010
**Theorem 7.9.** One can simulate any such protocol π with communication complexity $O(\sqrt{T \cdot C \cdot \log C})$.

The idea for the proof is quite straightforward. Alice and Bob use correlated sampling to repeatedly guess the bits of the messages in the protocol, without communicating. Then, they communicate a few bits to fix the errors in the transmissions.

First observe that without loss of generality, we can assume that there is no public randomness in the protocol we are simulating. This is because for each fixing of the public randomness $R = r$, if the internal information cost is $I_r$, and we obtain a simulating protocol with communication $\sqrt{I_r \cdot C \log C}$, then by convexity, the expected number of bits communicated for average $r$ is

$$
\mathbb{E}_{p(r)} \left[ \sqrt{I_r \cdot C \cdot \log C} \right] \leq \sqrt{\mathbb{E}_{p(r)} [I_r \cdot C \cdot \log C]} = \sqrt{I \cdot C \cdot \log C}.
$$

To carry out the simulation, we use correlated sampling, but since we will be sampling bits and not elements of a large universe, the sampling procedure is particularly simple. Let $\rho_1, \ldots, \rho_C$ be random numbers from the interval $[0, 1]$. For each prefix $m_{<i}$ of messages, define the number

$$
\gamma(m_{<i}) = p(M_i = 1|xym_{<i}).
$$

These numbers define the correct $m$ that our simulation protocol will attempt to compute. To define the correct $m$, for each $i$, set $m_i = 1$ if $\rho_i < \gamma(m_{<i})$, and set $m_i = 0$ otherwise. The correct $m$ has exactly the right distribution—the probability that $m$ is correct is

$$
\prod_{i=1}^C \gamma(m_{<i})^m_i (1 - \gamma(m_{<i}))^{1-m_i} = \prod_{i=1}^C p(m_i|xym_{<i}) = p(m|xy).
$$

Although Alice and Bob cannot compute $\gamma(m_{<i})$ without communicating, they can compute the numbers:

$$
\gamma^A(m_{<i}) = p(M_i = 1|xym_{<i}) \quad \text{and} \quad \gamma^B(m_{<i}) = p(M_i = 1|ym_{<i}).
$$

Moreover, if it is Alice’s turn to speak, then $\gamma^A(m_{<i}) = \gamma(m_{<i})$, and if it is Bob’s turn to speak, then $\gamma^B(m_{<i}) = \gamma(m_{<i})$, so:

**Claim 7.10.** Either $\gamma(m_{<i}) = \gamma^A(m_{<i})$, or $\gamma(m_{<i}) = \gamma^B(m_{<i})$.

So, Alice and Bob use these numbers to try and guess the correct $m$. Alice computes $m^A$ by setting $m^A_i = 1$ if and only if $\rho_i < \gamma^A(m_{<i})$, and Bob computes $m^B$ by setting $m^B_i = 1$ if and only if $\rho_i < \gamma^B(m_{<i})$. Of course, $m^A$ and $m^B$ are likely to be quite different. However, by Claim 7.10, if they are the same, then they must both be equal.
to $m$. To compute $m$, Alice and Bob communicate to find the first index $j$ where $m^A_j \neq m^B_j$. Using the results of Exercise 8.8, this takes $O(\log C/\epsilon)$ communication, if the probability of making an error is $\epsilon$.

If $m^A_{<i}$ dictates that Alice was supposed to send the $j$’th bit, then Bob sets $m^B_j = m^A_j$, otherwise Alice sets $m^A_j = m^B_j$. The two parties then use $\rho_{j+1}, \ldots, \rho_C$ to recompute $m^A, m^B$. They repeat this procedure until $m^A = m^B = m$.

To analyze the correctness of the simulation, we need to argue that Alice and Bob can find $m$ with small communication. To prove that the communication complexity of the protocol is small, we appeal to Pinsker’s inequality. We say that the protocol made a mistake at $i$ if during its execution, $m^A_i$ was found to be not equal to $m^B_i$. This happens exactly when $\rho_i$ lies in between the numbers $\gamma^A(m_{<i})$ and $\gamma^B(m_{<i})$, so given that $m$ is sampled by the protocol, the probability that there is a mistake at $i$ is at most

$$
\mathbb{E}_{p(xym)} [|\gamma^A(m_{<i}) - \gamma^B(m_{<i})|] = \mathbb{E}_{p(xym)} [|p(m_i = 1|x_m < i) - p(m_i = 1|y_m < i)|].
$$

Now for each fixing of $m_{<i}$, if the $i$’th message is supposed to be sent by Alice, we have

$$
\mathbb{E}_{p(xym|m)} [|p(m_i = 1|x_m < i) - p(m_i = 1|y_m < i)|] \leq \sqrt{I(X: M_i|Y_m < i)}, \quad \text{by Corollary 6.11}
$$

and if the $i$’th bit was to be send by Bob, then we have

$$
\mathbb{E}_{p(xym|m)} [|p(m_i = 1|x_m < i) - p(m_i = 1|y_m < i)|] \leq \sqrt{I(Y: M_i|X_m < i)}.
$$

In either case, we get that expected number of mistakes is at most

$$
\sum_{i=1}^{C} \sqrt{I(X: M_i|Y_m < i) + I(Y: M_i|X_m < i)} \leq \sqrt{C} \cdot \sqrt{\sum_{i=1}^{C} I(X: M_i|Y_m < i) + I(Y: M_i|X_m < i)} \quad \text{by the Cauchy-Schwartz inequality}
$$

$$
= \sqrt{C} \cdot \sqrt{I(X: M|Y) + I(Y: M|X)}. \quad \text{by the chain rule}
$$
Figure 7.8: Finding the correct path. In this case the correct path is obtained after 3 mistakes have been fixed.
Setting $\epsilon = 1/C^2$, the communication of the protocol is $O(\sqrt{C-1} \log C)$ in expectation. By Markov’s inequality, the probability that the communication exceeds 10 times this number is at most $1/10$, so we obtain a protocol with small communication overall.

**Direct Sums in Randomized Communication Complexity**

We already proved a direct sum theorem for deterministic communication complexity in Section 1. Here we prove similar results for randomized communication complexity.

**Theorem 7.11.** If the randomized communication complexity of $g$ is $c$, then the randomized communication complexity of $g^k$ is at least $\Omega(c\sqrt{k}\log c)$.

**Proof.** Suppose there is a protocol computing $g^k$ in the worst case, with $\ell$ bits of communication, and success probability $2/3$. Then by repeating the protocol a constant number of times (say 3 times), and taking the majority outcome, the probability of error is at most $\left(\frac{1}{3}\right)^3 + \left(\frac{2}{3}\right)^3 \cdot 3 > \frac{2}{3}$.

By Yao’s min-max theorem, there is a distribution $\mu$ on inputs to Alice and Bob such that every deterministic protocol that computes $g$ with less than $c$ bits of communication must make an error with probability bigger than $1/3$ when the inputs are sampled from this distribution. Let $(X_1, Y_1), (X_2, Y_2), \ldots, (X_k, Y_k)$ be $k$ independent inputs sampled according to the distribution $\mu$. Since the randomized communication complexity of $g^k$ is at most $3\ell$, there is a deterministic protocol computing $g^k$ on these inputs, with error less than $1/3$. Let $M$ denote the messages of this protocol when the inputs are sampled as above.

Setting $X = X_1, \ldots, X_k$, and $Y = Y_1, \ldots, Y_k$, by Lemma 6.14, we have

$$\sum_{i=1}^{k} I(X_i : M | X_{<i}Y_{\geq i}) \leq I(X : M | Y),$$

$$\sum_{i=1}^{k} I(Y_i : M | X_{<i}Y_{\geq i}) \leq I(Y : M | X).$$

Now consider the following protocol for computing $g$ on inputs from the distribution $\mu$. Alice and Bob sample $i \in [k]$ uniformly at random and sample $X_{<i}Y_{\geq i}$ using public randomness. They set their inputs to be $X_i, Y_i$. The internal information cost of the resulting protocol is at most $O(\ell/k)$, and its communication complexity is $O(\ell)$. By Theorem 7.9, the resulting protocol can be simulated with arbitrarily small constant error and communication

$$O(\sqrt{\ell \cdot \ell/k \log \ell}) = O(\ell \sqrt{1/k} \log \ell).$$
Round Lower bounds for Tree Pointer-Chasing

Even though correlated sampling deals with the regime where the information is extremely small, it is still quite useful to prove lower bounds on communication. We give an important example in this section. The tree pointer-chasing problem is a variant of the pointer-chasing problem\(^7\) that is useful to study for applications. In the version we study here, Alice and Bob are allowed to have a lot of common information, which makes proving the lower bound both harder and more fruitful.

Let \( T_{a,b} \) denote a rooted tree of depth \( k \), where every vertex at even depth has \( a \) children, and every vertex at odd depth has \( b \) children. Let \( E \) denote a subset of the edges of the tree, such that every vertex is connected to exactly one of its children in \( E \). The goal of Alice and Bob is to compute the unique leaf in \( T_{a,b} \) that remains connected to the root of the tree. However, Alice only knows a subset \( E_A \subseteq E \) of the edges, and Bob only knows a subset \( E_B \subseteq E \). \( E_A \) is promised to contain all the edges of \( E \) at even depth, and moreover, if a vertex \( v \) at odd depth is to the left of a sibling that is picked by its parent in \( E_A \), then all edges of \( E \) in the subtree rooted at \( v \) are included in \( E_A \). Similarly, \( E_B \) contains all edges of \( E \) at odd depth, and in addition, if a vertex \( v \) at even depth is to the left of a sibling that is picked by its parent in \( E_B \), then all edges of \( E \) in the subtree rooted at \( v \) are included in \( E_B \). A natural hard distribution for this problem is when the edges are sampled uniformly and independently. See Figure 7.9 for an example.

**Theorem 7.12.** Let \( M \) denote the messages in a deterministic \( k - 1 \) round protocol where Alice sends the first message and Alice sends \( a' \) bits in each round, Bob sends \( b' \) bits in each round. Let \( X \) denotes Alice’s input, \( Y \) denotes Bob’s input and \( L_1, \ldots, L_k \) denote the vertices on the unique path from the root to a leaf in the tree pointer chasing problem on \( T_{a,b} \). Then we must have

\[
\mathbb{E}_{p(m|L_{<k})} [ |p(L_k|L_{<k}) - p(L_k|L_{<k}m)| ] \leq (k - 1) \left( \sqrt{\frac{b' \ln 2}{2a}} + \sqrt{\frac{a' \ln 2}{2b}} \right).
\]

**Theorem 7.12** shows that any protocol that has few rounds of communication must make an error:

**Corollary 7.13.** Any randomized \( k - 1 \) round protocol where Alice sends the first message and Alice sends \( a' \) bits in each round, Bob sends \( b' \) bits in each round.

\(^7\) Nisan and Wigderson, 1993; and Klauck et al., 2007

For example, it will be used to prove lower bounds for data structures solving the predecessor search problem.

Including the edges in the left subtrees corresponds to knowing \( x_1, \ldots, x_{i-1} \) in the indexing problem.

The inputs to Alice and Bob are correlated, even though the edges sampled are independent.
round must make an error with probability

\[ \frac{1}{2} + (k - 1) \left( \sqrt{\frac{b' \ln 2}{2a}} + \sqrt{\frac{a' \ln 2}{2b}} \right), \]

when solving tree pointer chasing on a tree \( T_{a,b} \) of depth \( k \).

**Proof of Theorem 7.12.** Suppose the edges \( E \) are sampled uniformly at random, and the edges \( E_A, E_B \) are given to Alice and Bob. We shall prove the theorem by induction on \( k \). When \( k = 1 \), the theorem is trivially true, since there is no communication.

Suppose \( k > 1 \). Let \( X_i \) denote the edges of the tree at even depth, in the subtree rooted at the \( i \)'th child of the root, and let \( Y_i \) denote the edges at odd depth in the same subtree. Let \( M_1 \) denote the first message of the protocol, sent by Alice. We shall prove

\[
\mathbb{E}_{p(x_{<\ell_1}, y_{>\ell_1} | m_1)} \left[ |p(x_{\ell_1}, y_{\ell_1}) - p(x_{\ell_1}, y_{\ell_1} | m_1 \ell_1 x_{<\ell_1}, y_{>\ell_1})| \right] \leq \epsilon, \quad (7.1)
\]

with \( \epsilon = \sqrt{\frac{a' \ln 2}{2b}} \). After the first message has been transmitted, and we have fixed \( \ell_1, x_{<\ell_1}, y_{>\ell_1} \), one can think of the rest of the protocol as a \( k - 2 \) round protocol where Bob speaks first. If \( x_{\ell_1}, y_{\ell_1} \) were truly
uniform after this conditioning, induction would give
\[
\mathbb{E}_{p(x_{<\ell_1}y_{>\ell_1}m_1\ell_1)} \left[ \mathbb{E}_{p(\ell_k|m_1)} \left[ |p(\ell_k|\ell_{<k}) - p(\ell_k|m\ell_{<k})| \right] \right] < (k - 2) \left( \sqrt{\frac{b' \ln 2}{2a}} + \sqrt{\frac{a' \ln 2}{2b}} \right).
\]
However, since the distribution on \(x_{\ell_1}, y_{\ell_1}\) is only \(\epsilon\)-close to uniform, the error has an additional term of \(\epsilon\). This proves the final statement.

It only remains to prove (7.1). By Lemma 6.14, if \(X = X_1, \ldots, X_b\) and \(Y = Y_1, \ldots, Y_b\), we have
\[
\frac{1}{b} \sum_{i=1}^{b} I(X_i : M_1 | X_{<i}Y_{\geq i}) \leq I(X : M_1 | Y) \leq \frac{a'}{b}.
\]
By Pinsker’s inequality (Corollary 6.11), we get that on average over the choice of \(i, x_{<i}y_{\geq i}, m_1\),
\[
p(x_i|x_{<i}y_{\geq i}m_1) \approx p(x_i|x_{<i}y_{\geq i}),
\]
with \(\epsilon \leq \sqrt{\frac{a' \ln 2}{2b}}\). Since \(m_1\) is computed by Alice, and Alice’s inputs are independent of \(\ell_1\), we have \(p(y_{\ell_1} | x_{<\ell_1}y_{>\ell_1}m_1) = p(y_{\ell_1})\). Thus we get that
\[
\mathbb{E}_{p(x_{<\ell_1}y_{>\ell_1}m_1\ell_1)} \left[ |p(x_{\ell_1}y_{\ell_1}) - p(x_{\ell_1}y_{\ell_1}|m_1\ell_1x_{<\ell_1}y_{>\ell_1})| \right] \\
\leq \mathbb{E}_{p(x_{<\ell_1}y_{>\ell_1}m_1\ell_1)} \left[ |p(x_{\ell_1}) - p(x_{\ell_1}|m_1\ell_1x_{<\ell_1}y_{\geq \ell_1})| \right] \leq \epsilon,
\]
as required. \(\Box\)

Round Lower Bound for Greater Than

Suppose Alice and Bob each have \(n\)-bit numbers, and want to know which of their numbers is greater. We have already shown that the deterministic communication complexity of this problem is \(n + 1\), and argued that the randomized communication complexity is \(\Theta(\log n)\). Here we study the randomized communication complexity of bounded round protocols for this problem.

**Theorem 7.14.** Any \(k\) round protocol for computing the greater than function on \(x, y \in [2^n]\) must transmit at least \(\Omega(n^{1/k}/k^2)\) bits in some round.

**Proof.** We prove the theorem by appealing to the lower bound for the tree pointer chasing problem (Theorem 7.12). Say we have an input to the tree pointer chasing problem on a tree \(T_{a, a}\) of depth \(k\). We set \(n = a^k\).
We show how Alice and Bob can transform the tree into inputs $x \in \{0, 1, \ldots, 2^n - 1\}$ and $y \in \{0, 1, \ldots, 2^n - 1\}$ for the greater-than problem, without communicating. The numbers $x, y$ are best thought of as $a^{k-1}$-digit numbers written in base $a$. The transformation will guarantee that the greater than function will reveal undue information about the identity of the last leaf in the tree.

We describe how to carry out the reduction. If the tree is of depth 1, with the edge coming out of the root in Alice’s input, we set $x \in \{0, 1, \ldots, a-1\}$, and $y = \lceil a/2 \rceil$. If the edge from the root is in Bob’s input, we set $y$ based on the edge, and set $x = \lceil a/2 \rceil$. Viewing $x, y$ as numbers written in base $a$, these are single digit numbers.

If the tree is of depth $k > 1$, with Alice knowing the edge coming out of the root, we first compute $x_0, \ldots, x_{a-1}, y_0, \ldots, y_{a-1}$, which are the inputs to predecessor search determined by the $a$ subtrees of depth $k-1$ that lie just below the root. By induction, these correspond to numbers with $a^{k-2}$ digits. Suppose $i \in \{0, 1, \ldots, a-1\}$ corresponds to the edge coming out of the root. In words, we shall view $x, y$ as $a$-digit numbers with $a^{k-1}$ digits. The digits of $x, y$ can be thought of as broken up into $a$ consecutive blocks, each with $a^{k-2}$ digits. For all $j$, we set the $j$’th block of $y$ to be $y_j$. We set the first $i-1$ blocks of $x$
to be the same as \( y \). For \( j \geq i \), we set the \( j \)'th block of \( x \) to be \( x_j \). Since Alice knows all the edges in the first \( i - 1 \) subtrees, Alice can compute \( x \), and Bob can compute \( y \).

This construction has the property that if \( \ell_k \) is the \( k \)'th leaf, and the last edge on the root to leaf path is visible only to Alice, then \( x > y \) if and only if this last leaf is the \( r \)'th child of its parent, with \( r > a/2 \). Similarly, if this last edge is visible to Bob, then \( y > x \) if and only if this last edge is greater than \( a/2 \). By Theorem 7.12, the protocol can succeed only if the number of bits communicated in each round is \( \Omega(a/k^2) \). Since \( a = n^{1/k} \), the proves the required bound. \( \square \)
Part II

Complexity
8
Circuits and Proofs

Although communication complexity ostensibly studies the amount of communication needed between parties that are far apart, it has had quite an impact in understanding many other concrete computational models and discrete systems. In this chapter, we discuss the applications of communication complexity to boolean circuits and proof systems.

Boolean Circuits

Boolean circuits are the most natural model for computing boolean functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$. A boolean circuit is a directed acyclic graph whose vertices, often called gates, are associated with boolean operators or input variables. Every gate with in-degree 0 corresponds to an input variable, the negation of an input-variable, or a constant bit, and all other gates compute either the logical AND (denoted by the symbol $\land$) or the OR (denoted by the symbol $\lor$) of the inputs that feed into them. Usually, the fan-in of the gates is restricted to being at most 2. We adopt this convention, unless we explicitly state otherwise.

Every gate $v$ in a circuit computes a boolean function $f_v$ of the inputs to the circuit. We say that a circuit computes a function $f$ if $f = f_v$ for some gate $v$ in it. Every circuit is associated with two standard complexity measures: The size of the circuit is the number of gates, and the depth of the circuit is the length of the longest directed path in the underlying graph. The size corresponds to the number of operations the circuit performs, and the depth to the parallel-time it takes the computation to end, using many processors.

Understanding the complexity of computation with boolean circuits is extremely important, because they are a universal model of computation. Any function that can be computed by an algorithm...
in $T(n)$ steps can also be computed by circuits of size $\tilde{O}(T(n))$. So, to prove lower bounds on the time complexity of algorithms, it is enough to prove that there are no small circuits that can carry out the computation. Counting arguments imply that almost every function requires circuits of exponential size\(^1\). However, we know of no explicit function for which we can prove a super-linear lower bound, highlighting the difficulty in proving such lower bounds.

We focus on two well-known types of circuits. A formula is a circuit whose underlying graph is a tree. Every circuit of depth $d$ can always be turned into a formula whose size is at most $2^d$, and depth is at most $d$. A monotone circuit is a circuit that does not use any negated variables. A monotone circuit computes a monotone function—$f(y) \geq f(x)$ whenever $y_i \geq x_i$ for all $i$. Every boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ can be computed by a circuit\(^2\) of depth $n$ and size at most $O(2^n/n)$. A similar statement holds for the size and depth of monotone circuits computing monotone functions.

We now describe a general connection between circuit complexity and communication complexity.

### Karchmer-Wigderson Games

Every boolean function that is not identically 0 or 1 defines a communication problem via its Karchmer-Wigderson game\(^3\). In the game defined by $f$, Alice gets $x \in f^{-1}(0)$, Bob gets $y \in f^{-1}(1)$, and they seek to find $i \in [n]$ such that $x_i \neq y_i$. When $f$ is monotone, one can define the monotone Karchmer-Wigderson game to be the problem where the inputs are $x \in f^{-1}(0), y \in f^{-1}(1)$ as before, but now Alice and Bob want to find an $i$ such that $x_i < y_i$.

The basic observation is that if there is a circuit of depth $d$ computing $f$, then the communication complexity of the associated game is at most $d$. Indeed, if the gate computing $f$ is computed as $f = g \land h$, then either $g(x) = 0$ or $h(x) = 0$, while $g(y) = h(y) = 1$. Alice can announce whether $g(x)$ or $h(x)$ is 0, and the parties can continue the protocol using $g$ or $h$. Similarly if $f = g \lor h$, Bob can announce whether $g(y) = 1$ or $h(y) = 1$, and the parties then continue with either $g$ or $h$. After at most $d$ steps, the parties identify an index $i$ for which $x_i \neq y_i$. When the circuit is monotone, the same simulation finds an index $i$ such that $x_i = 0, y_i = 1$, since there are no negated variables.

The topology of the circuit determines the topology of the protocol tree. Every AND gate corresponds to a node in the protocol tree where Alice speaks, and every OR gate corresponds to a node where Bob speaks. Moreover, a circuit of depth $d$ gives a protocol whose

---

\(^1\) Shannon, 1949

The number of circuits of size $s$ can by bounded by $2^{O(s \log s)}$, while the number of functions $f$ is $2^n$, so if $s \ll 2^n/n$, one cannot hope to compute every function with a circuit of size $s$.

Prove this fact by induction on the depth.

\(^2\) Lupanov, 1958

The size of the circuit captures the total number of basic operations needed to evaluate it. The depth captures the number of parallel steps needed to evaluate it: if a circuit has size $s$ and depth $d$, then it can be evaluated by $s$ processors in $d$ time steps.

Several restricted classes of circuits are not discussed in this book. We restrict our attention to methods related to communication complexity. For example, if the circuit is allowed to have gates of arbitrarily large fan-in, then it makes sense to talk about circuits of bounded depth. Today, we know that constant depth circuits require exponential size to compute functions like parity and majority. These lower bounds are sometimes proved by taking a random restriction of the inputs, or by using methods based on polynomials.

\(^3\) Karchmer and Wigderson, 1990

Note that Alicei and Bob are not computing a function of their inputs—in general there are many indices $i$ with the property that they want.

Show that when $f$ is monotone, there must be an $i$ with $x_i < y_i$. 
communication complexity is at most $d$.

Conversely, we have:

**Lemma 8.1.** If the Karchmer-Wigderson game for a non-constant function $f$ can be solved with $d$ bits of communication, then there is a circuit of depth $d$ computing $f$. If $f$ is monotone, and the monotone game can be solved with $d$ bits of communication, then there is a monotone circuit of depth $d$ computing $f$.

**Proof.** We shall prove, by induction on $d$, that for any non-empty sets $A \subseteq f^{-1}(0), B \subseteq f^{-1}(0)$, the following holds. If there is a protocol such that whenever $x \in A$ is given to Alice and, $y \in B$ is given to Bob, they can exchange $d$ bits to find $i$ such that $x_i \neq y_i$, then there is a circuit of depth $d$ computing a boolean function $g$ with $g(A) = 0$, and $g(B) = 1$. When $A = f^{-1}(0), B = f^{-1}(1)$, this implies the lemma. If we are working with the monotone game, we shall prove that the resulting circuit is monotone.

When $d = 0$, the protocol must have a fixed output $i$, and so we must have that $x_i \neq y_i$ (or $x_i = 0, y_i = 1$ in the monotone case) for every $x \in A, y \in B$. Thus, setting $g$ to be the $i$'th variable or its negation works.

Suppose $d > 0$ and Alice speaks first. Then, her message partitions the set $A$ into two non-empty disjoint sets $A = A_0 \cup A_1$, where $A_0$ is the set of inputs that lead her to send 0 as the first message, and $A_1$ is the set of inputs that lead her to send 1. Both sets must be non-empty, since otherwise the first bit of the protocol need not be transmitted, and the lemma follows by induction.

By induction, the two children of the root node in the protocol tree correspond to boolean functions $g_0$ and $g_1$, with $g_0(A_0) = g_1(A_1) = 0$ and $g_0(B) = g_1(B) = 1$. Consider the circuit that takes the AND of the two gates obtained inductively, and denote the function it computes by $g$. Then for all $y \in B$, $g(y) = g_0(y) \land g_1(y) = 0 \land 1 = 1$. For all $x \in A$, either $x \in A_0$ or $x \in A_1$. In either case $g(x) = g_0(x) \land g_1(x) = 0$. If the first bit of the protocol is sent by Bob, the proof is similar, except we take the OR of the gates obtained by induction. 

One immediate consequence of Lemma 8.1 is regarding the circuit depth required to compute the majority and parity functions. In Section 1 and Exercise 1.3, we proved that solving the Karchmer-Wigderson games for these functions requires at least $2 \log n - O(1)$ bits of communication. This shows that both of these functions requires circuits of depth $2 \log n - O(1)$, and hence the smallest formulas for these functions have size $\Omega(n^2)$.

As we shall see, the Karchmer-Wigderson connection between circuit complexity and communication complexity is a powerful tool for proving lower bounds. As exercise ?? shows, one can use information complexity to prove a similar lower bound in the distributional setting.
Lower bounds on the Depth of Monotone Circuits

One of topics we do not yet understand in circuit complexity is the power of depth:

Open Problem 8.2. Can every function that is computable using circuits of size polynomial in $n$ be computed by circuits of depth $O(\log n)$?

However, we do know how to prove interesting results when the underlying circuits are monotone.

Matching

One of the most studied combinatorial problems is finding the largest matching in a graph. A matching is a set of disjoint edges. Today, we know of several polynomial time algorithms that can find the matching of largest size in a given graph$^4$.

Given a graph $G$ on $n$ vertices, define

$$
\text{Match}(G) = \begin{cases} 
1 & \text{if } G \text{ has a matching of size at least } n/3 + 1, \\
0 & \text{otherwise.} 
\end{cases}
$$

Since there are polynomial time algorithms for finding matchings, one can obtain polynomial sized circuits that compute Match. However, we do not know of any logarithmic depth circuits that compute Match. Here we show that there are no monotone circuits of depth $o(n)$ computing Match$^5$.

By Lemma 8.1, it is enough to prove a lower bound on the communication complexity of the corresponding monotone Karchmer-Wigderson game. In the monotone matching game, Alice gets a graph $G$ with $\text{Match}(G) = 1$ and Bob gets a graph $H$ with $\text{Match}(H) = 0$. Their goal is to find an edge which is in $G$, but not in $H$. We shall prove:

Theorem 8.3. Any randomized protocol solving the matching game must communicate $\Omega(n)$ bits.

As a corollary, we get:

Corollary 8.4. Every monotone circuit computing Match has depth $\Omega(n)$.

Proof of Theorem 8.3. The theorem is proved by reduction to the disjointness lower bound proved in Theorem 6.13. We shall show that if there is a protocol for the monotone matching game with...
communication complexity $c$, then there is a randomized protocol of complexity $O(c)$ solving the disjointness problem on a universe of size $\Omega(n)$. By Theorem 6.13, this implies that $c \geq \Omega(n)$.

Assume $n = 3m + 2$ for a non-negative integer $m$. Suppose Alice and Bob get inputs $X \subseteq [m]$ and $Y \subseteq [m]$. They will encode $X$ and $Y$ as two graphs $G_X$ and $H_Y$ on the vertex set $[3m + 2]$, and then feed the graphs into the protocol for the monotone matching game. The construction of $G_X$ and $H_Y$ is shown in Figure 8.2.

**Building $G_X$** Alice constructs the graph $G_X$ as follows. For each $i \in [m]$, the graph $G_X$ contains the edge $\{3i, 3i - 1\}$ if $i \in X$, and has the edge $\{3i, 3i - 2\}$ if $i \notin X$. In addition, $G_X$ contains the edge $\{3m + 1, 3m + 2\}$. The construction ensures that $G_X$ consists of $m + 1$ disjoint edges, and so $G_X$ contains a matching of size $m + 1$.

**Building $H_Y$** Bob uses $Y$ to build a graph $H_Y$ as follows. For each $i \in [m]$, if $i \in Y$ then Bob connects $3i - 2$ to all the other $3m + 1$ vertices of the graph, and if $i \notin Y$ then Bob connects $3i$ to all the other vertices. Every edge in $H_Y$ must touch one of the gray vertices in the figure, but there are only $m$ such gray vertices. So, $H_Y$ does not contain a matching of size $m + 1$.

Using public randomness, Alice and Bob permute the vertex set $[3m + 2]$ uniformly at random, and run the promised protocol for the monotone matching game on the permuted graphs.

If $X$ and $Y$ are disjoint, the outcome of the protocol must be the edge corresponding to $\{3m + 1, 3m + 2\}$. On the other hand, if $X$ and $Y$ intersect in $k > 0$ elements, then there are exactly $k + 1$ edges in $G_X$ that are not in $H_Y$. Since the graph is permuted randomly before the protocol is executed, the outcome of the protocol is equally likely to be one of these $k + 1$ edges. So, the probability that the protocol outputs the edge corresponding to $\{3m + 1, 3m + 2\}$ is at most $1/2$. If the output of the protocol is not the edge corresponding to $\{3m + 1, 3m + 2\}$, the players know that the sets are not disjoint.
Repeating this experiment a constant number of times, the players are able to solve disjointness with probability of error at most 1/3. □

**Monotone Circuit Depth Hierarchy**

We can use the connection to communication to show that monotone circuits of large depth are strictly more powerful than circuits of small depth. Throughout this section, we work with circuits of arbitrarily large fan-in.

Let $F_{n,k}$ be the formula that is computed by the full AND-OR tree with gates of fan-in $n$ and depth $k$. This is the formula where all non-input gates have fan-in exactly $n$. The gates of odd depth are OR gates, and the gates of even depth are AND gates. Every input gate is labeled by a distinct unnegated variable. The size of $F$ is $O(n^k)$.

We shall prove that any formula of smaller depth computing $F$ must have exponential size:

**Theorem 8.5.** Any monotone circuit of depth $k - 1$ that computes $F$ must have size at least $2^{n/16 - k}$.

**Proof.** It suffices to show that any protocol computing the associated monotone Karchmer-Wigderson game has communication at least $n/16 - k$. The lower bound follows, since if the size of the circuit is at most $s$, the communication of a $k - 1$ round protocol for the Karchmer-Wigderson game can be at most $(k - 1)[\log s]$.

We prove that the Karchmer-Wigderson game has large communication by reducing the problem to the pointer-chasing problem that we studied in Chapter 6. Here Alice and Bob are given $x, y \in [n]^n$ and want to compute $z = z(k)$, where $1 = z(0), z(1), z(2), \ldots$ are inductively defined using the rule

$$
z(i) = \begin{cases} 
xz(i-1) & \text{if } i \text{ is odd,} 

\end{cases}
\begin{cases} 
yz(i-1) & \text{if } i \text{ is even.} 
\end{cases}
$$

Given inputs $x, y$ to the point-chasing problem, the inputs $x', y'$ in $\{0, 1\}^{|n|^k}$ to $F$ are constructed as follows. Note that every variable in the formula can be described by a string in $v \in [n]^k$. We say that $v$ is consistent with $x$ if

$$v_i = \begin{cases} 
x_1 & \text{when } i = 1, 

x_{v_{i-1}} & \text{when } i \text{ is odd and not 1.} 
\end{cases}
$$

We say that $v$ is consistent with $y$ if $v_i = y_{v_{i-1}}$ when $i$ is even. Alice sets all the coordinates of $x'$ that are consistent with her input to be
0, and all other coordinates to be 1. Bob sets all the coordinates of $y'$ that are consistent with his input to be 1, and all other coordinates to be 0.

We now prove that $F(x') = 0$ and $F(y') = 1$. We focus on $F(x')$; a similar argument works for $F(y')$. Every a gate of depth $d$ in the formula corresponds to a vector in $[n]^d$. We claim that every gate that corresponds to a vector that is consistent with Alice's input evaluates to 0. This clearly true for the gates at depth $k$, since that is how we set the variables in $x'$. For the gates at depth $d < k$, if the gate is an AND gate then one of its children is consistent with $x$ and so evaluate to 0, and if the gate is an OR gate then all of its children are consistent with Alice's input and so evaluate to 0.

For all $x, y$, there is a unique $v$ that is consistent with both $x$ and $y$, and that's when $v = z(k)$, the output of the pointer-chasing problem. The only coordinate where $x'$ is 0 and $y'$ is 1 is the coordinate that is consistent with both $x, y$.

Thus, any protocol for the monotone Karchmer-Wigderson game gives a protocol solving the pointer-chasing problem, and so by Theorem 6.17, we get that the communication of the game must be at least $n/16 - k$, as required.

Boolean Formulas

A formula is a circuit whose underlying graph is a tree. Although we do not know how to prove super-linear circuit lower bounds for arbitrary circuits, we do know how to prove super-linear lower bounds for formulas. When it comes to formulas, the choice of basis can affect the formula size by more than a constant factor. Nevertheless, one can prove super-linear lower bounds even when allowing each gate to compute an arbitrary function of two bits.

Consider the function $\text{Distinct}(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } x_1, \ldots, x_n \text{ are distinct}, \\ 0 & \text{else.} \end{cases}$

Distinct is a boolean function that depends on $O(n \log(n))$ bits. We shall prove:

**Theorem 8.6.** Any formula computing $\text{Distinct}$ must have at least $n^2 - O(n \log n)$ input gates.

To prove the theorem, we start by proving a simple communication lower bound. Suppose Alice is given $n$ numbers $y_1, \ldots, y_n \in [2n]$, and Bob is given $z \in [2n]$. They want to compute $\text{Distinct}(y_1, \ldots, y_n, z)$.

For example, one can compute the parity $x_1 \oplus \cdots \oplus x_n$ of $n$ bits with a linear sized tree of parity gates, but one can show that $O(n^2)$ gates are required if the formula is restricted to using AND and OR gates.

*Neciporuk, 1966*
Lemma 8.7. If there is a 1-round protocol where Alice sends Bob \( t \) bits and Bob outputs \( \text{Distinct}(y_1, \ldots, y_n, z) \), then \( t \geq \log \left( \frac{2^n}{n} \right) \geq 2n - O(\log n) \).

**Proof.** To prove the lower bound, it is enough to consider the case when \( y_1, \ldots, y_n \) are distinct elements. In this case, Alice’s message must determine \( S = \{y_1, \ldots, y_n\} \), or else Bob will not be able to compute \( \text{Distinct}(y_1, \ldots, y_n, z) \). This is because if \( S \neq S' \) are two sets of size \( n \) that are consistent with Alice’s message, then there must be an element \( z \in S \) such that \( z \notin S' \). Then \( z \) is distinct from \( S' \), but not from \( S \). Thus, the number of bits transmitted by Alice must be at least \( \log \left( \frac{2^n}{n} \right) \), as required.

We are ready to prove the formula lower bound:

**Proof of Theorem 8.6.** Suppose there is a formula \( F \) computing \( \text{Distinct} \) using \( s \) gates. Each input gate in the formula reads a bit of one of the numbers \( x_i \) in the input to \( \text{Distinct}(x_1, \ldots, x_{n+1}) \). For each \( i \in [n+1] \) we define the tree \( T_i \) as follows. Every vertex of \( T_i \) corresponds to a gate in \( F \). Start by discarding all the gates in \( F \) that do not depend on \( x_i \). In the graph that remains, iteratively replace every gate that has only one input feeding into it with an edge connecting its input to its output, as in Figure 8.3.

Suppose Alice knows all of the input numbers except \( x_i \), Bob knows \( x_i \), and Alice and Bob want to compute \( \text{Distinct}(x_1, \ldots, x_n) \). They can use the tree \( T_i \) to carry out the computation efficiently as follows. Bob already knows the values at the leaves of the \( T_i \). Every gate \( v \) in \( T_i \) a boolean function \( f_v \) which depends on gates in \( T_i \) and some number of Alice’s inputs. There are \( 2^{2^2} = 2^4 \) boolean
functions that depend on two variables, so Alice can send 4 bits to Bob to indicate which of these functions he should use to compute $f_2(x_1, \ldots, x_{n+1})$ using the 2 inputs that correspond to gates of $T_i$.

Using this information, Bob can compute $\text{Distinct}(x_1, \ldots, x_{n+1})$. The overall communication is at most 4 times the number of vertices in $T_i$.

Since $F$ has only $s$ gates, there must be some $i$ for which $T_i$ has at most $\ell = s/n$ leaves. If $m$ denotes the number of vertices of $T_i$, and $e$ the number of edges in $T_i$, then we must have $e = m - 1$, since $T_i$ is a tree. On the other hand, counting the number of edges by adding up the degrees of the vertices, we have

$$2(m - 1) = 2e \geq 3(m - \ell - 1) + \ell,$$

which implies that $m \leq 2\ell + 1 \leq 2s/n + 1$.

By Lemma 8.7, we get $2s/n + 1 \geq 2n - O(\log n)$, proving the theorem. 

Similar ideas can be used to show non-trivial lower bounds even when the gates are allowed to compute arbitrary functions of $2n/3$ variables. Suppose we want to express a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ as

$$f = g(g_1, \ldots, g_k),$$

where each of the functions $g_1, \ldots, g_k$ depends on $2n/3$ input bits.

What is the minimum $k$ required? This is equivalent to asking about the minimum fan-in required to compute $f$ using a circuit of depth 2 using gates that compute arbitrary functions of $2n/3$ variables.

We can $\text{Distinct}$ with $k = O(\log n)$. Nevertheless, there is a closely related explicit function that requires $k \geq n^{\Omega(1)}$. It remains an open problem to find an explicit function for which $k = n$.

For $S \in \binom{[n]}{\lfloor n \log(2n) \rfloor}$, a subset of $[n \log(2n)]$ of size $\log(2n)$, and $b \in \{0, 1\}^{n \log(2n)}$, define the scrambled distinctness function $\text{SDistinct}(S, b)$ as follows. Use the coordinates of $S$ in $b$ to define a number $z \in [2n]$. Use the remaining bits of $b$ to define $y_1, \ldots, y_{n-1} \subseteq [2n]$, and finally output $\text{Distinct}(y_1, \ldots, y_{n-1}, z)$. We can prove:

**Theorem 8.8.** $\text{SDistinct}(S, b)$ requires $k \geq n^{\Omega(1)}$.

**Proof.** As in the formula lower bound, we shall appeal to Lemma 8.7. Suppose we are given a circuit with gates of fan-in $k$ that computes $\text{SDistinct}$ as $g(g_1, \ldots, g_k)$. Each of the gates $g_i$ depends on at most $2/3$’rds of the input variables.

We claim that if $k$ is small, there must be some $S$ for which every gate $g_i$ reads at most $4 \log(2n)/5$ of inputs that correspond to $S$. Indeed, suppose we pick the elements of $S$ independently and uniformly at random. For any $i$, the expected number of coordinates of $S$ read by $g_i$ is at most $2 \log(2n)/3$, so by the Chernoff-Hoeffding

---

7 Hrubes and Rao, 2015

The parameter $2n/3$ can be varied, but we choose $2n/3$ for simplicity of the discussion.

Counting arguments show that most functions $f$ require $k = n$. Try to verify this by counting the number of such circuits.

To see this, let $S_1, \ldots, S_k \subset [n]$ be sets of size $n/2$. If this sequence has the property that for every $i, j \in [n]$, there is some set of the sequence that contains both $i, j$, then we can use the sequence to construct a formula. Let $g_i$ be the function that reads the numbers $x_j$ for $j \in S_i$, and outputs 1 if and only if these numbers are distinct, and let $g$ be the function that takes the OR of its inputs.

The input to $\text{SDistinct}$ can be encoded using $n \log(2n) + \log^* n$ bits.
bound, the probability that more than $4 \log(2n)/5$ of the coordinates are read is at most $e^{-\Omega(\log(2n))} = n^{-\gamma}$, for some constant $\gamma$. The probability that the $\log(2n)$ coordinates sampled are not all distinct is at most $\log^2(2n)/n$. So, if $k < n^{\gamma}/2$, then $k \cdot n^{-\gamma} + \log^2(2n)/n < 1$, and there is a set $S$ satisfying the properties we want.

Given such a set $S$, Alice and Bob can use the circuit to obtain a protocol solving the distinctness problem. Bob sets the coordinates of $b$ in $S$ according to his input, and Alice sets the remaining coordinates according to her input. Each gate $g_i$ depends on at most $4 \log(2n)/5$ of Bob’s bits. There are $2^{\log(2n)/5} = 2^{O(n^{4/5})}$ boolean functions that depend on $4 \log(2n)/5$ bits, so Alice can send Bob $k \cdot O(n^{4/5})$ bits to describe the function Bob should evaluate to compute each $g_i$. By Lemma 8.7, we must have that $k \cdot O(n^{4/5}) \geq \Omega(n)$. Thus, $k \geq n^{\Omega(1)}$, as required.

Boolean Depth Conjecture

Can every function computable by a polynomial sized circuit also be computed by a circuit of depth $O(\log n)$? In a sense, this is the same as asking whether every circuit can be balanced—a balanced tree with $s$ nodes has depth $O(\log s)$. If a function cannot be computed by a balanced circuit, this suggests that the computation of the function cannot be parallelized.

This simple problem remains open, despite much effort to resolve it. Here we discuss an approach\textsuperscript{8} based on direct-sums in communication complexity to proving that there is a function that can be computed using polynomial sized circuits but cannot be computed by a circuit of depth $O(\log n)$.

Given a function $f : \{0,1\}^t \rightarrow \{0,1\}$, and a function $g : \{0,1\}^k \rightarrow \{0,1\}$, define the composition of the functions as $f \circ g : \{0,1\}^{tk} \rightarrow \{0,1\}$ by $f \circ g(x_1, \ldots, x_t) = f(g(x_1), \ldots, g(x_k))$. Define $f^{\circ k}$ to be $f \circ f$, and $f^{\circ k}$ to be $f \circ f^{k-1}$, the composition of $f$ with itself $k$ times.

Now, if $f : \{0,1\}^t \rightarrow \{0,1\}$ is any function with $t > 2$, the function $f^{\circ t} : \{0,1\}^{t^2} \rightarrow \{0,1\}$ can be computed using a boolean circuit of size $2 \cdot 2^t \cdot t^2 < (t^2)^2$. Indeed, $f$ itself can be computed using a circuit of size $2^t$, and the tree of evaluations of $f$ has at most $2 \cdot t^2$ nodes, so one can compute $f^{\circ t}$ using $2 \cdot 2^t \cdot t^2 < (t^2)^2$ gates.

Almost all functions $f$ as above require circuit depth $t$. If we could show that there is a function $f$ as above for which the circuit depth of $f \circ f^{k-1}$ must be at least $ct$ more than the circuit depth of $f^{\circ k-1}$, then that would imply that the circuit depth of $f^{\circ t}$ is at least $ct^2 \gg t \log t = \log t^t$. This would prove that there is a boolean function depending on $n = t^t$ variables that can be computed using

\textsuperscript{8}Karchmer et al., 1995
Figure 8.4: A refutation of $F$. In each step, two clauses are combined to give a new clause that must be true. The final step produces an empty clause, which represents a contradiction.

Proof Systems

Proof systems provide a framework for proving theorems and for studying the complexity of proofs. A proof system is a specific language for expressing proofs. It consists of a set of rules that allow one to logically derive a theorem from axioms. The study of proof systems has lead to many interesting results, including Gödel’s famous incompleteness theorem.9

Resolution Refutations

Perhaps the simplest example of a proof system is resolution.10 Resolution can be used to refute a boolean formula $F$ expressed in conjuctive normal form, that is, to show that $F$ cannot possibly be satisfied.

For example, consider the formula

$$F = (x_2 \lor x_1) \land (\neg x_2 \lor x_1) \land (\neg x_1 \lor x_3 \lor \neg x_4) \land (\neg x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor \neg x_3).$$

$F$ cannot be satisfied by any boolean assignment. To prove that the formula is unsatisfiable, we repeatedly use the resolution rule. The

$O(n^2)$ gates, but cannot be computed with a circuit of depth $O(\log n)$.

In terms of communication complexity, all that is needed is an example of a function $f$ for which the communication complexity of the Karchmer-Wigderson game of $f$ is at least $ct$ larger than the communication complexity of the game of $f \circ f^{\circ k - 1}$. This looks quite similar to understanding the direct-sum question in communication that we studied in Chapter 1 and Chapter 7. However, the ideas we discussed there do not seem to apply in this situation.

9Gödel, 1931

10Robinson, 1965
rule derives a clause that must be true if two other clauses are both true:

\[(a \lor B) \land (\neg a \lor C) \implies B \lor C,\]

where \(a\) is a variable, \(B, C\) are clauses and \(B \lor C\) is the derived clause obtained by including all the literals in \(B\) and \(C\). The resolution refutation for \(F\) shown in Figure 8.4 uses this rule to give a proof that \(F\) cannot be satisfied.

In general, a resolution refutation is a sequence of clauses where each clause is derived by combining two previously derived clauses using the resolution rule. The proof ends when the empty clause, which implies a contradiction, is derived. The proof is said to be tree-like if every derived clause is used only once. A tree-like proof corresponds to the concept of a formula in circuit complexity.

The problem of understanding whether a formula in conjunctive normal form is satisfiable (SAT) is a central problem because of its connection\(^{11}\) to the complexity classes \(NP\) and \(coNP\). The best SAT-solvers known today try to find a satisfying solution while simultaneously trying to prove that formulas obtained after partial assignments cannot be satisfied using resolution refutations. Thus, it is important to understand what kinds of formulas can be efficiently refuted.

To study the power of a given proof system, like resolution, we need to study sequences of formulas of growing complexity. A basic example of such a sequence is the well-known pigeonhole principle.

**The Pigeonhole Principle**

The pigeonhole principle states that if \(n\) pigeons are placed into \(n-1\) holes, then some hole must contain at least 2 pigeons. One can use the principle to construct a sequence of unsatisfiable boolean formulas. For \(i \in [n]\) and \(j \in [n-1]\), we have the variable \(x_{i,j}\) which indicates that the \(i\)'th pigeon is in the \(j\)'th hole. Define the following \(n+1\) formulas:

\[
\forall i \in [n], \quad P_i = \bigvee_{j \in [n-1]} x_{i,j},
\]

\[
H = \bigwedge_{i < i' \in [n]} \big( \neg x_{i,j} \lor \neg x_{i',j} \big).
\]

The pigeonhole principle implies that

\[
P = H \land \bigwedge_{i} P_i
\]
cannot be satisfied by any assignment to the variables $x_{i,j}$.

How hard is it to prove that $P$ is unsatisfiable? If one uses resolution, we can prove that it is very hard\textsuperscript{12}:

**Theorem 8.9.** Any resolution refutation of the pigeonhole principle must involve $2^{\Omega(n)}$ derivation steps.

We give the proof of this theorem, even though it is not directly related to communication complexity. It will help us get a feel for the basic notions in proof complexity. Later, we explain the relationship between proof complexity to communication complexity in the context of a different lower bound.

A key idea in the proof is to give the proof system even more power. We allow the proof to assume the following axiom for free:

**Axiom 8.10.** Each hole contains exactly one pigeon, and the $n-1$ pigeons that are in the holes are distinct.

This can only make it easier to derive a contradiction. Indeed, Axiom 8.10 implies that for each $i,j$,

$$
\neg x_{i,j} \Leftrightarrow \bigvee_{i' \neq i} x_{i',j}.
$$

This allows us to replace every negated variable in the proof with a disjunction of unnegated variables.

Consider any refutation of $P$ that derives $s$ clauses. Let $C$ be one of the clauses derived in the proof. We say that $C$ is **big** if there is a set $S \subset [n]$ of size $|S| \geq n/4$ such that for each $i \in S$ the number of $j$'s so that $C$ contains $x_{i,j}$ is at least $n/4$.

Let us see how a random assignment affects the refutation. Pick $n/4$ of the pigeons uniformly at random, and randomly assign them to $n/4$ different holes. If pigeon $i$ is assigned to hole $j$ in this process, then we set $x_{i,j} = 1$, we set $x_{i',j} = 0$ for all $i' \neq i$, and $x_{i,j'} = 0$ for all $j' \neq j$. In words, this amounts to making sure that the relevant pigeons and holes are not involved with any of the remaining holes and pigeons. This assignment is enforced by Axiom 8.10.

After this assignment to the variables, $n/4$ of the pigeon clauses become true. Moreover, several variables disappear, and the formula becomes equivalent to the corresponding formula for $3n/4$ pigeons and $3n/4 - 1$ holes. The resolution refutation must still derive a contradiction. We claim:

**Claim 8.11.** One of the big clauses must survive the assignment.

**Proof.** Consider the refutation of $P$ after the random assignment. Say that a clause has pigeon complexity $w$ if there is a set $S \subset [n]$ of size $|S| \geq n/4$ such that

$$
\neg x_{i,j} \Leftrightarrow \bigvee_{i' \neq i} x_{i',j}.
$$

It is no loss of generality to assume that 4 divides $n$. If this is not the case, replace $n$ with a nearby multiple of 4.

\textsuperscript{12} Haken, 1985; and Beame and Pitassi, 1996

In fact, the proof will show that an exponential number of steps are required in any proof system where each step derives a clause using any derivation rule from two clauses.
w such that

$$\bigwedge_{i \in S} P_i \Rightarrow C,$$

yet no smaller set $S$ has this property.

The contradiction can only be derived from all $3n/4$ pigeon clauses that remain, since one can satisfy any strict subset of those clauses with some assignment to the variables. So, the empty clause in the proof has pigeon complexity at least $3n/4$. Since the contradiction is derived from two clauses, one of the clauses used to derive the contradiction must have pigeon complexity at least $3n/8$. Continuing in this way, we obtain a sequence of clauses in the proof, where each clause requires at least half as many pigeon clauses as the previous one. Since the clauses of $P$ have pigeon complexity at most $1 < n/4$, there must be a clause $C$ in this sequence that has pigeon complexity between $n/4$ and $n/2$.

Let $S \subset [n]$ be the minimal set of pigeon clauses that imply $C$. Suppose $i \in S$ and $j$ is a hole that did not yet receive a pigeon during the random assignment. Since $S$ is minimal, there must be an assignment to all the variables where $\bigwedge_{i' \in S \setminus \{i\}} P_{i'}$ is true, yet $C$ is false. This assignment places all of the pigeons of $S$ into holes, except for the $i$'th pigeon. Suppose $i' \notin S$ and $x_{i',j} = 1$ in this assignment. Consider what happens when we set $x_{i',j} = 0$ and $x_{i,j} = 1$, and leave the rest of the variables as they are. Doing so must make $C$ true, since $\bigwedge_{i \in S} P_i = 1$ in the assignment. Since $C$ is a disjunction of unnegated variables, this can only happen if $C$ contains $x_{i,j}$.

Thus for each $i \in S$, there must be at least $3n/4 - n/2 = n/4$ values of $j$ for which $x_{i,j}$ is in the clause $C$. So, $C$ is big even in the proof after the random assignment.

Claim 8.12. If a clause $C$ is big, then the probability that $C$ survives the random assignment is at most $(\frac{63}{64})^{n/8}$.

Proof. Consider what happens when the first pigeon is assigned to a hole. The probability that the pigeon is one of the $n/4$ pigeons relevant to $C$ is at least $1/4$. The probability that it is assigned to one of the $n/4$ holes that would imply $C$ is at least $1/4$. So the probability that $C$ becomes true after the first pigeon is assigned to a hole is at least $1/16$. Continuing in this way, we see that for each of the first $n/8$ pigeons that we assign to a hole in the random assignment, there are at least $n/4 - n/8 = n/8$ pigeons which if assigned to $n/4 - n/8 = n/8$ holes would lead to the clause becoming true. Thus, the probability that $C$ survives the first $n/8$ assignments of pigeons to
holes is at most
\[
(1 - \frac{(n/8) \cdot (n/8)}{n^2})^{n/8} = \left(\frac{63}{64}\right)^{n/8},
\]
as required.

We are ready to prove the theorem:

**Proof of Theorem 8.9.** Suppose towards a contradiction that the refutation of $P$ has less than $(64/63)^{n/8}$ clauses. By Claim 8.12, there is an assignment of the pigeons to holes such that every big clause does not survive. On the other hand, by Claim 8.11, at least one big clause must survive.

---

**Cutting Planes**

A stronger proof system than resolution can be obtained by reasoning about linear inequalities instead of clauses. For example, the clause $a \lor \neg b \lor c$ can be viewed as asserting that the boolean variables $a, b, c$ satisfy the linear inequality

\[
a + 1 - b + c \geq 1,
\]
or equivalently,

\[
-a + b - c \leq 0.
\]

In the cutting planes proof system, we convert the clauses into inequalities and then argue about the inequalities. Suppose the variables in the proof are $x = x_1, \ldots, x_n$. Since the variables are boolean, we allow the proof to use the inequalities $x_i \leq 1$ and $-x_i \leq 0$ for free. We are allowed to take positive linear combinations of two inequalities in the natural way, and round the right hand side. If we have already derived $\langle c, x \rangle \leq t$ and $\langle c', x \rangle \leq t'$, for any non-negative $a, a'$ for which $ac + a'c'$ is an integer valued vector, we can use the derivation rule to derive

\[
\begin{align*}
\langle c, x \rangle &\leq t \\
\langle c', x \rangle &\leq t'
\end{align*}
\implies \langle ac + a'c', x \rangle \leq at + a't'.
\]

We also allow the rounding rule:

\[
\langle c, x \rangle \leq t \implies \langle c, x \rangle \leq \lfloor t \rfloor,
\]

namely, we can replace any number on the right hand side with the closest integer that is smaller than it. This makes sense because $x$ is boolean and $c$ is always integer valued. One proves that the clauses are unsatisfiable by arriving at the contradiction $1 \leq 0$. 

---

CIRCUITS AND PROOFS

159
The cutting planes system can efficiently simulate resolution, so it is at least as powerful as resolution. For example, consider the resolution derivation
\[
\begin{align*}
- x \lor y \lor z \\
- x \lor y \lor -w
\end{align*}
\Rightarrow y \lor z \lor -w.
\]

Viewing the clauses as inequalities, this corresponds to
\[
\begin{align*}
x - y - z &\leq 0 \\
w &\leq 1 \\
-x - y + w &\leq 0 \\
z &\leq 0 \\
\frac{1}{2} \cdot (x - y - z + w &\leq 1) \\
\frac{1}{2} \cdot (-x - y - z + w &\leq 0)
\end{align*}
\Rightarrow -y - z + w \leq 0.
\]

This derivation does not directly follow by taking linear combinations—if we add the first two inequalities, we get \(-2y - z + w \leq 0\), which is not quite what we want. However, we can derive the inequality we seek using the rounding rule. We have:
\[
\begin{align*}
x - y - z &\leq 0 \\
w &\leq 1 \\
-x - y + w &\leq 0 \\
z &\leq 0 \\
\frac{1}{2} \cdot (x - y - z + w &\leq 1) \\
\frac{1}{2} \cdot (-x - y - z + w &\leq 0)
\end{align*}
\Rightarrow -y - z + w \leq \lfloor 1/2 \rfloor = 0.
\]

In this way, one can show:

**Lemma 8.13.** If a formula can be refuted in \(s\) steps using resolution, then it can be refuted in \(O(ns)\) steps using cutting planes.

In fact, cutting planes gives a strictly stronger proof system. For example, one can give a cutting planes refutation of the formula for the pigeonhole principle using just \(O(n^2)\) steps\(^\text{13}\). Rewriting the clauses of the pigeonhole principle as linear inequalities, we get:
\[
P_i \equiv - \sum_{j=1}^{n-1} x_{i,j} \leq -1,
\]
\[
H_{i,j'} \equiv x_{i,j} + x_{i',j} \leq 1.
\]

We claim that for each \(j\), we can derive the inequality
\[
L_{k,j} \equiv \sum_{i=1}^{k} x_{i,j} \leq 1
\]
in \(O(k)\) steps. The inequality \(L_{2,j}\) is \(H_{1,2,j}\). To derive \(L_{k,j}\) from previously derived inequalities, use the derivation rule \(k\) times to get
\[
(k - 1) \cdot L_{k-1,j} + \sum_{i=1}^{k-1} H_{i,k,j}
\]
\[
\equiv k(x_{1,j} + x_{2,j} + \ldots + x_{k,j}) \leq 2k - 1.
\]

\(^{13}\)Cook et al., 1987; and Jukna, 2012
Now, divide by $k$ and round to get $L_{k,j}$. To complete the proof, observe

$$\sum_{j=1}^{n-1} L_{n,j} = \sum_{j=1}^{n-1} \sum_{i=1}^{n} x_{ij} \leq n - 1,$$

while

$$\sum_{i=1}^{n} P_i = -\sum_{i=1}^{n-1} \sum_{j=1}^{n} x_{ij} \leq -n.$$

Adding these last two inequalities gives $1 \leq 0$.

To summarize, cutting planes can efficiently prove the pigeonhole principle, although resolution can not. Are there simple formulas that are difficult to refute using cutting planes? We shall use communication complexity to analyze such an example.

**Lower Bounds on Cutting Planes**

Here we give an example of an unsatisfiable formula that requires an exponential number of steps to refute in the cutting planes proof system. The formula is based on the combinatorial properties of graphs.

Given a graph, a **vertex cover** is a set of vertices $U$ such that every edge of the graph contains at least one vertex from $U$. A **matching** is a set of disjoint edges. We shall design the formula to encode the fact that given any vertex cover in a graph and any matching, the vertex cover must have more vertices than the matching has edges. Indeed, every edge in the matching must be covered by one vertex from the vertex cover, and the edges are disjoint.

We shall construct a formula that asserts that the input graph has a vertex cover of size $k - 1$, as well as a matching of size $k$—this ensures that the formula is unsatisfiable. For each possible edge $e = \{v, u\} \subset [n]$, we have the variable $x_e$ which is 1 if and only if the edge $e$ is present in the graph. For $i \in [k]$ and $e$, the variable $y_{i,e}$ encodes whether $\{u, v\}$ is the $i$'th edges in the matching. For $j \in [k - 1]$ and a vertex $v \in [n]$, the variable $z_{j,v}$ encodes whether $v$ is the $j$'th vertex in a cover. Now, define the following formulas:

$$C = \bigwedge_{e \in \binom{[n]}{2}} \left( \neg x_e \lor \bigvee_{v \in e, j \in [k-1]} z_{j,v} \right),$$

every edge is covered

$$\forall j \in [k-1], \quad C_j = \left( \bigvee_{v} z_{j,v} \right) \land \bigwedge_{v \neq v' \in [n]} \left( \neg z_{j,v} \lor \neg z_{j,v'} \right),$$

the $j$'th vertex in the cover is unique
Finally, define the formula:

$$F = C \land \left( \bigwedge_{j=1}^{k-1} C_j \right) \land \left( \bigwedge_{i=1}^{k} M_i \right) \land M \land K.$$  

The formula $F$ has at most $O(n^4)$ clauses. However, an exponential number of inequalities are needed to refute it, at least with a tree-like proof. Here we prove\(^{14}\):

**Theorem 8.14.** Any tree-like cutting planes refutation of $F$ must derive $2^{\Omega(n/\log n)}$ inequalities.

To prove the theorem, we shall reduce the problem to the communication complexity of the matching game, for which we proved a lower bound in Theorem 8.3. In the matching game, Alice gets a graph $G$ so that that has a matching of size $k \approx n/3$, and Bob gets a graph $H$ that does not have a matching of size $k - 1$. Their goal is to find an edge which is in $G$, but not in $H$. We prove:

**Lemma 8.15.** If there is a tree-like cutting plane proof of size $s$ showing that $F$ is not satisfiable, then there is a randomized protocol for the matching game with communication $O(\log(s)(\log(n) + \log \log(s)))$.

By Theorem 8.3, we must have $\log(s)(\log(n) + \log \log(s)) \geq \Omega(n)$, which proves Theorem 8.14. The proof of the lemma shows how to efficiently convert a cutting planes refutation to a communication protocol for the relevant game. In fact, the proof extends to many other formulas that have similar structure, but simplicity, we limit the discussion to this particular formula. A key step in the proof is the observation that we can efficiently check if linear inequalities that arise in the proof are true or not.

**Proof of Lemma 8.15.** Alice sets the variables $y_{i,e}$ to be consistent with her matching, and Bob sets the variables $z_{j,v}$ and $x_e$ to be consistent with the graph $H$. Under this setting of variables, all of the clauses in

\[ M = \bigwedge_{e,e' \in \binom{[n]}{2}, |e \cap e'| = 1, i \neq i' \in [k]} (-y_{i,e} \lor -y_{i',e'}) , \]  

matching edges are disjoint

\[ \forall i \in [k], \quad M_i = \left( \bigvee_{e \in \binom{[n]}{2}} y_{i,e} \right) \land \left( \bigwedge_{e \neq e' \in \binom{[n]}{2}} (-y_{i,e} \lor -y_{i,e'}) \right) , \]  

the $i$'th edge of the matching is a unique edge in the graph

\[ K = \bigwedge_{e \in \binom{[n]}{2}, j \in [k]} (x_e \lor -y_{j,e}) . \]  

decides the matching are are in the graph

\[ C = \bigwedge_{e \in \binom{[n]}{2}} (x_e \lor -y_{j,e}) . \]  

decides the matching are are in the graph

\[ F = C \land \left( \bigwedge_{j=1}^{k-1} C_j \right) \land \left( \bigwedge_{i=1}^{k} M_i \right) \land M \land K. \]  

\(^{14}\) Krajíček, 1997; Pudlák, 1997; Impagliazzo et al., 1994; and Hrubeš, 2013
$M_i, M, C_j, C$ are true, but one of the clauses in $K$ must be false. This false clause specifies an edge that is in Alice’s graph $G$ but not in Bob’s graph $H$. Our goal is to find this clause using the refutation of $F$.

By Lemma 1.8, there must be some inequality $L$ in the proof that depends on at most $2s/3$ of the clauses, and on at least $s/3$ of the clauses. Let $L$ be such an inequality. Our aim is to check whether $L$ is satisfied or not under the assignment to the variables held by Alice and Bob. $L$ can be written as

$$\kappa + \sum_{i, x} \alpha_i \cdot y_i \cdot x \leq \sum_{j, y} \beta_{j, y} \cdot z_j \cdot y + \sum_{e} \gamma_e \cdot x_e.$$ 

All of the variables on the left hand side are known to Alice, and all the variables on the right hand side are known to Bob. Since the variables are boolean, there are at most $2^n$ possible values for the left hand side, and at most $2^n$ possible values for the right hand side.

Alice and Bob can thus use the randomized protocol for solving the greater-than problem to compute whether or not this inequality is satisfied by their variables (see Exercise 8.1). They expend $O(\log(n) + \log(1/\epsilon))$ bits of communication in order to make sure that output of their computation is correct with error $\epsilon$.

If the inequality $L$ is not satisfied, Alice and Bob can safely discard the clauses that are not used to derive $L$, and continue to find a false clause. Otherwise, all of the clauses used to derive $L$ can safely be discarded, and Alice and Bob can start their search again after discarding all the inequalities used to derive $L$.

In either case, they discard at least $s/3$ clauses. This process can repeat at most $O(\log s)$ times, so the probability that they make an error is at most $O(\epsilon \log s)$ by the union bound. Setting $\epsilon$ to be small enough so that this number is at most $1/3$, we obtain a protocol whose communication is at most $O((\log n + \log \log s)(\log s))$, as promised.

Exercise 8.1

Show that the formula that asserts that there cannot be a graph which both has a $k$-matching and a set of size $k - 1$ that covers every edge requires an exponential number of inequalities to prove in the cutting planes proof system.

Exercise 8.2

Show that the formula that asserts that there cannot be a graph on $[n]$ which both has a path from 1 to $n$ and a set $S \subset [n]$ with $1 \in S$, $n \notin S$ $k$-matching and a set of size $k - 1$ that covers every edge
requires an exponential number of inequalities to prove in the cutting planes proof system.
Chapter 9

Streaming Algorithms and Branching Programs

The memory used by an algorithm is an important resource. In this chapter, we explore two related models that measure the amount of memory used, and prove lower bounds on the best possible algorithms optimizing this resource, using communication complexity.

A branching program of length \( \ell \) and width \( w \) is a layered directed graph whose set of vertices is a subset of \([\ell + 1] \times [w] \). Each layer from 1 to \( \ell \) is associated with a variable in \( x_1, \ldots, x_n \in [d] \). Every vertex in layers 1 to \( \ell \) has \( d \) out-going edges, each labeled by a distinct symbol from \([d] \). Edges go from layer \( u \) to layer \( u + 1 \) for \( u \leq \ell \). The vertices on layer \( \ell + 1 \) are labelled with an output of the program.

Computing a function \( f(x) \) using the branching program is straightforward. On input \( x \in [d]^n \), the program is executed by starting at the vertex \((1, 1) \) and reading the variables associated with each layer in turn. These variables define a path through the program. The program outputs the label of the last vertex on this path.

Intuitively, if an algorithm uses only \( s \) bits of memory, then it can be modeled as a branching program of width at most \( d^s \). Every function \( f : \{0, 1\}^n \to \{0, 1\} \) can be computed by a branching program of width \( 2^n \) and length \( n \), and counting arguments show that most such functions require exponential width. Another motivation for understanding branching programs comes from a powerful theorem due to Barrington:

**Theorem 9.1.** If \( f : \{0, 1\}^n \to \{0, 1\} \) can be computed by a boolean circuit of depth \( O(\log n) \), then it can be computed by a branching program of width 5 and length \( n^{O(1)} \).

Barrington’s theorem implies that if a function requires a super-polynomial length when the width is restricted to being 5, then it must require super-logarithmic depth. As we discussed in Chapter 8, finding a function that requires super-logarithmic depth is a major open problem.
A streaming algorithm is a specific type of branching program—one where the inputs (often called the data stream) are read exactly once, and in order: $x_1, x_2, \ldots, x_n$. Streaming algorithms are motivated by applications where massive amounts of data need to be processed quickly. In these applications, we cannot afford to store all of the data that is coming in, so we need to process it on the fly, and yet be able to compute some function that depends on all the inputs. We also want to compute the result after making just one pass on the data.

The parameter $\log w$ is called the space of the streaming algorithm.

We start by describing a couple of clever streaming algorithms.

**Maximum Matching** Suppose the data stream consists of a sequence of edges $e_1, \ldots, e_m \in \binom{[n]}{2}$. The goal of the algorithm is to find a matching of largest size in the graph formed by the union of the edges.

There is a very simple algorithm for finding a matching that is within a factor of 2 of the largest one, using space at most $2n \log n$. We store each new edge as long as it does not intersect any of the previously stored edges. At most $n/2$ edges are stored at the end, and these edges must form a matching. The space of the algorithm is at most $(n/2) \cdot \log n^2 \leq n \log n$.

We claim that this algorithm must compute a matching that is at least half as big as the largest matching in the graph. Indeed, by construction, every edge output by the algorithm must intersect an edge of the largest matching, and it can only intersect two such edges. Thus, if the algorithm outputs $t$ edges, the largest matching must have size at most $2t$.

**Frequency moments** Suppose the data stream consists of a sequence of elements $x_1, \ldots, x_m \in [n]$. For $i \in [n]$, let $f(i)$ denote the number of times that $i$ occurs in the input sequence. The $t$'th moment of the sequence is defined to be $\sum_{i=1}^{n} f(i)^t$.

We can efficiently count the 1'st moment $\sum_{i=1}^{m} f(i)^1$—this is just $m$, which can be computed using space $\log m$. The 0'th moment is just the number of distinct elements in the sequence. Although computing the 0'th moment requires space $n$ in general, one can estimate it with less space using a randomized algorithm. Let $S \subset [n]$ be a random subset obtained by sampling $k$ uniformly random independent elements of $[n]$. In Chapter 3, we gave a protocol for the gap-hamming problem, where we showed that counting the number of distinct elements in $S$ is enough to approximate the number of distinct elements in the sequence, up to an additive error of $m = O(n/\sqrt{k})$. The number of distinct elements within the set $S$ can be counted using space $S$. Better algorithms are

The algorithms we give here are randomized. Although they are not as strong as deterministic algorithms, they do imply that for every distribution on inputs, there is an efficient deterministic algorithm with small error.
known\(^1\), if we wish to estimate the number of distinct elements up to a small multiplicative factor.

Here we show that one can estimate the 2’nd moment \(\sum_{i=1}^{n} f(i)^2\) of the stream\(^2\) efficiently:

**Theorem 9.2.** For any constant \(\delta\), we can estimate the 2’nd moment up to a factor of \(1 - \epsilon\), with probability of error \(\delta\), using memory \(O\left(\frac{\log m}{\epsilon^2}\right)\).

For a parameter \(k\), and \(i \in [n], j \in [k]\), let \(e_{i,j} \in \{+1, -1\}\) be uniformly random, and independent of all other variables. Consider the parameter \(X_j = \sum_{l \in [n]} e_{i,j} f(i)\). \(X_j\) can be computed from the data stream using at most \(\log n\) bits of memory. We shall argue that the average of \(X_j^2, \ldots, X_k^2\) is a good estimate for the 2’nd frequency moment. To do this, we shall use Chebyshev’s inequality. We have:

\[
\mathbb{E} \left[ X_j^2 \right] = \mathbb{E} \left[ \left( \sum_{i=1}^{n} e_{i,j} f(i) \right)^2 \right] = \mathbb{E} \left[ \sum_{i=1}^{n} e_{i,j}^2 \cdot f(i)^2 \right] + \mathbb{E} \left[ \sum_{i \neq i'} e_{i,j} \cdot e_{i',j} \cdot f(i)f(i') \right] = \sum_{i=1}^{n} f(i)^2.
\]

So, \(X_j^2\) has the right expected value, for each \(j\). We have

\[
\mathbb{E} \left[ X_j^4 \right] = \mathbb{E} \left[ \left( \sum_{i=1}^{n} e_{i,j} f(i) \right)^4 \right] = \mathbb{E} \left[ \sum_{i=1}^{n} e_{i,j}^4 \cdot f(i)^4 \right] + \mathbb{E} \left[ \sum_{i \neq i'} \frac{4!}{2} \cdot e_{i,j}^2 \cdot e_{i',j}^2 \cdot f(i)^2 f(i')^2 \right] = \sum_{i=1}^{n} f(i)^4 + 6 \sum_{i \neq i'} f(i)^2 f(i')^2.
\]

So, the variance of \(X_j^2\) can be bounded by

\[
\mathbb{E} \left[ X_j^4 \right] - \mathbb{E} \left[ X_j^2 \right]^2 = 4 \sum_{i \neq i'} f(i)^2 f(i')^2 \leq 2 \left( \sum_{i=1}^{n} f(i)^2 \right)^2.
\]

The variance of \(Z\) can be bounded:

\[
\mathbb{E} \left[ Z^2 \right] - \mathbb{E} \left[ Z \right]^2 = \mathbb{E} \left[ \left( \frac{1}{k} \cdot \sum_{j=1}^{k} X_j^2 \right)^2 \right] - \mathbb{E} \left[ \frac{1}{k} \cdot \sum_{j=1}^{k} X_j^2 \right]^2 = \frac{1}{k^2} \cdot \sum_{j=1}^{k} \mathbb{E} \left[ X_j^4 \right] - \mathbb{E} \left[ X_j^2 \right]^2 \leq \frac{2}{k} \left( \sum_{i=1}^{n} f(i)^2 \right)^2.
\]

In general, the variance of the average of \(k\) independent identically distributed random variables is always smaller by a factor of \(k\). Since \(X_1, \ldots, X_k\) are independent.

\(^1\) Alon et al., 1999

The second moment is an important parameter, because it can be used to estimate the number of heavy hitters—elements that occur much more often than other.

\(^2\) Alon et al., 1999

\(k\) will be set to \(O(1/\epsilon^2)\).
By Chebyshev’s inequality, we get
\[
\Pr \left[ |Z - \sum_{i=1}^{n} f(i)^2| \geq \epsilon \sum_{i=1}^{n} f(i)^2 \right] \leq \frac{2}{\epsilon^2 k^2}.
\]

Setting \( k = \Omega(1/\epsilon) \) gives a good approximation to the 2\(^{nd}\) moment.

**Lower bounds on streaming algorithms**

The dominant method for proving lower bounds on the memory requirements of streaming algorithms is by appealing to lower bounds in communication complexity. Our approach will be to break the stream into two parts\(^3\). Alice will simulate the execution of the algorithm on the first part of the data stream. She will then send Bob the contents of the memory, allowing him to continue the simulation over the second half of the data stream. Perhaps surprisingly, this approach often gives tight lowerbounds.

**Lower bounds for estimating the frequencies of a stream**

To illustrate this basic idea, let us start with computing the frequency moments. Recall that the inputs is a stream of elements \( x_1, \ldots, x_m \in [n] \), and \( f(i) \) denotes the number of times that \( i \in [n] \) occurs in the stream.

Consider the problem of computing the 0’th moment, which is the number of distinct elements in the stream. We prove\(^4\):

**Theorem 9.3.** Any randomized streaming algorithm estimating the number of distinct elements in the stream up to an error of \( \sqrt{n} \), requires memory at least \( \Omega(n) \).

**Proof.** The theorem follows by reduction to the lower bound on the communication complexity of the gap-hamming problem. Indeed, suppose Alice has a string \( x \in \{0,1\}^n \) and Bob has a string \( y \in \{0,1\}^n \), and they wish to estimate the hamming distance between \( x, y \). Then, viewing \( x \) as the indicator vector for a set \( S \subseteq [n] \), Alice simulates the execution of the given algorithm on \( S \), and computes the contents of the memory after the algorithm has seen the elements of \( S \). Alice sends Bob the contents of the memory, as well as \( |S| \). Bob continues executing the algorithm on the elements of the set \( T \), obtained by viewing \( y \) as the indicator vector of \( T \). After the algorithm has finished executing, Bob recovers \( |S \cup T| \). The hamming
distance between $x$ and $y$ is exactly
\[
|S \cup T| - |S \cap T| = |S \cup T| - (|S| + |T| - |S \cup T|)
= 2 \cdot |S \cup T| - |S| - |T|,
\]
a quantity that Bob can compute. By Theorem 5.17, the memory must take $\Omega(n)$ bits to encode.

Next, suppose we are interested in computing the maximum frequency: $\max_i f(i)$. A simple reduction to the communication complexity of disjointness\(^5\) proves:

**Theorem 9.4.** Any randomized algorithm that can estimate $\max_i f(i)$ within a multiplicative factor of 2 must use memory $\Omega(n)$.

**Proof.** Suppose Alice and Bob have sets $X, Y \subseteq [n]$ and want to know whether the sets are disjoint or not. Alice simulates an execution of the streaming algorithm whose input stream consists of the elements of $X$, and then sends the contents of the memory to Bob. Bob continues the simulation using the elements of $Y$. If $X$ and $Y$ are disjoint, the maximum frequency is at most 1. If they are not disjoint, then the maximum frequency is at least 2. So, the output of the algorithm allows Alice and Bob to distinguish the two cases. By Theorem 6.13, the memory must contain $\Omega(n)$ bits. □

**Lower bounds for computing the maximum matching**

If the algorithm is restricted to using space $n \text{polylog}(n)$, the best known single-pass algorithm for computing the maximum matching is the algorithm we presented at the beginning of this chapter. Although we do not know whether this algorithm is optimal, we can show\(^6\) that no algorithm can approximate the size of the maximum matching up to a factor of $2/3$. In fact, it is known that there can be no algorithm\(^7\) with an approximation factor better than $1 - 1/e$.

A key combinatorial construction that is useful for proving the lower bound is a dense graph that can be covered by many induced matchings. A matching in a graph is induced if there is a subset of vertices $A$ such that an edge belongs to the matching if and only if the edge is contained in $A$.

**Theorem 4.2** asserts that there is a subset $T \subseteq [n]$ of size at least $n/2 - \Omega(\sqrt{\log n})$ that does not contain any arithmetic progressions. One can use the set $T$ to construct\(^8\) such a graph. Let $A, B$ be two disjoint sets of vertices of size $3n$. Identify each of these sets with $[3n]$. Now put an edge between two vertices $x \in A, y \in B$, if $y = v - t, x = v + t$, for some $v \in [n/3, 2n/3], t \in T$. In this way, we get $n/3$ matchings, one for every fixed choice of $v$. Moreover, these matchings

---

\(^5\) Alon et al., 1999

\(^6\) Goel et al., 2012

\(^7\) Kapralov, 2013

\(^8\) Ruzsa and Szemerédi, 1978
are induced: for example if \((v - t, v + t), (v - t', v + t'), (v - t', v + t)\) are all edges, then \((v + t) - (v - t') = t + t'\) must be equal to \(2t''\) for some \(t'' \in T\). Then we see that \(t, t', t''\) form an arithmetic progression in \(T\).

The graph we obtain in this way does not quite have the parameters we need to prove the strongest lower bound, but a better construction is known\(^9\):

**Theorem 9.5.** For every \(\delta > 0\), there is a constant \(c\) such that for every \(m\) there is a bipartite graph with \(m\) vertices on each side that is the disjoint union of \(m^{1-c/\log \log m}\) induced matchings, and each matching has at least \((1/2 - \delta)m\) edges.

When \(\delta\) is small, the graph promised by the theorem has to have at least \(m^{2-c/\log \log m}/3\) edges. Given Theorem 9.5, and ideas similar to those used to prove the communication complexity lower bound on indexing, we can prove:

**Theorem 9.6.** For any constant \(\delta > 0\), there is a constant \(c\) such that any randomized streaming algorithm that computes a matching whose size is at least \(2/3 + \gamma\) fraction of the size of the maximum matching, with probability \(2/3\), and memory \(\ell\) must have \(\gamma \leq 4\delta + O\left(\ell/n^{2-c/\log \log n}\right)\).

**Proof.** Let \(G\) be the graph promised by Theorem 9.5, which consists of \(k \geq m^{1-c/\log \log m}\) induced matchings on \(2m\) vertices, each of size at least \(t = (1/2 - \delta)m\). We consider the following communication game. Alice gets a set of edges \(H\), and Bob gets a set of edges \(J\). Their goal is to output a large matching that is contained in the union of their edges. In the game, Alice must send a message to Bob, and Bob must output the final matching. Alice and Bob can always use the streaming algorithm to get a protocol for the game—Alice simulates the execution of the algorithm on her edges, and then sends the contents of the memory to Bob, who completes the execution on his edges and outputs the edges found by the algorithm.

Consider the following distribution on edges. For \(\delta\) as in Theorem 9.5, the graph \(H_i\) is obtained by picking exactly \((1 - \delta)t\) of edges \(H_i\) from the \(i\)’th matching of \(G\) uniformly and independently. Let \(i \in [k]\) be uniformly random. The graph \(J\) is obtained by matching all of the edges that do not touch the \(i\)’th matching in \(G\) to distinct vertices from a new set of vertices \(S\). See Figure 9.2.

The largest matching in the graph is obtained by taking all of the edges that touch \(S\), together with the edges of \(H\) that belong to the \(i\)’th matching of \(G\). Indeed, if any matching in \(H \cup J\) includes an edge of \(G\) that is not in the \(i\)’th matching, then we can always remove that edge and replace it with two edges touching \(S\) to obtain a larger matching. Since Bob knows the edges touching \(S\), it is no loss of
generality to assume that Bob always outputs the edges touching \( S \) in his matching. When Bob does not make an error, he must output an additional \( r \) edges from \( H_i \). We shall prove that \( r \) must be small.

Let \( M \) denote the \( \ell \)-bit message that Alice sends to Bob, and let \( H_1, \ldots, H_k \) denote the edges of \( H \) that come from each of the \( k \) matchings. By Theorem 6.9, we have

\[
\sum_{i=1}^{k} I(H_i : M) \leq I(H_1, \ldots, H_k : M) \leq \ell.
\]

So, there must be an \( i \) for which \( I(H_i : M) \leq 2\ell/k \) and the probability of making an error given \( i \) is at most \( 2/3 \). Fix such an index \( i \). Let \( E \) be the random variable that is 1 if the algorithm makes an error or outputs a matching of size at most, and 0 otherwise. We compute:

\[
I(H_i : ME) \leq 2\ell/k + 1
\]

and yet

\[
I(H_i : ME) = H(H_i) - H(H_i|ME).
\]
H(H_i) is exactly log \((\frac{t}{1-\delta})^t\). We have the identity
\[
\binom{a-1}{b-1} = \frac{a! \cdot (b-1)! \cdot (a-b)!}{b! \cdot (a-b)! \cdot (a-1)!} = \frac{a}{b}
\]
which implies
\[
\frac{\binom{a}{b}}{\binom{a-j}{b-j}} = \prod_{j=0}^{r-1} \frac{a-j}{b-j} = \frac{a}{b}^r.
\]

Whenever \(E = 0\), the protocol outputs \(r\) edges of \(H_i\) that are correct. So, we get:
\[
I(H_i : ME) \geq \log \left( \frac{t}{(1-\delta)t} \right) - \frac{2}{3} \cdot \log \left( \frac{t-r}{(1-\delta)t-r} \right)
\]
\[
= \frac{1}{3} \cdot \log \left( \frac{(1-\delta)t}{t-r} \right) \geq \frac{r}{3} \cdot \log \left( \frac{t}{(1-\delta)t} \right)
\]
This gives
\[
\frac{2\ell}{k} + 1 \geq \frac{r}{3} \cdot \log \left( \frac{1}{1-\delta} \right).
\]
and so
\[
r \leq O \left( \frac{\ell}{k} \right).
\]
So, we get that the approximation ratio of the algorithm can be bounded
\[
\frac{r + 2(n-t)}{(1-\delta)t + 2(n-t)} = \frac{2(n-t)}{(1-\delta)t + 2(n-t)} + \frac{r}{(1-\delta)t + 2(n-t)}
\]
\[
= \frac{n + 2\delta n}{3n/2 + \delta n/2 + \delta^2 n} + O \left( \frac{\ell}{kn} \right)
\]
\[
\leq \frac{2}{3} + 4\delta + O \left( \frac{\ell}{kn} \right) \leq \frac{2}{3} + 4\delta + \frac{\ell}{n^{2-1/\log^2 n}},
\]
since \(k \geq n^{1-c/\log \log n}\),
as required.

\[\square\]

Lower bounds on branching programs

Communication complexity can be used to prove lower bounds on general branching programs as well, though we need to use a different reduction, since a branching program may read the variables in arbitrary order. Here we define explicit functions that cannot be
computed by branching programs that are simultaneously short and narrow.

To prove the lower bound, we first show that any branching program can be efficiently simulated, at least in some sense, by a communication protocol in the number-on-forehead model (Chapter 4).

Let \( g : \{0, 1\}^k \to \{0, 1\} \) be an arbitrary function that \( k \)-parties wish to compute in the number-on-forehead model. Define the function \( g' \) by

\[
g'(x, S_1, \ldots, S_k) = g(x'|S_1, \ldots, x'|S_k),
\]

where \( x \in \{0, 1\}^n, S_1, \ldots, S_k \) are subsets of \( [n] \) of size \( r \), and \( x|S_i \in \{0, 1\}^r \) is the projection of \( x \) to the coordinates in \( S_i \). The input to \( g' \) can be described using at most \( n + O(tr \log n) \) bits. The key claim\(^{10} \) is that any branching program computing \( g' \) can be used to obtain an efficient protocol computing \( g \) in the number-on-forehead model.

**Theorem 9.7.** There is a constant \( \gamma > 0 \) such that for any \( g, g' \) as above with \( t \leq 8\sqrt{\gamma} \log n, r \leq \sqrt{n} \), the following holds. If \( g' \) can be computed by a length \( \gamma n \log^2 n \), width \( w \) branching program, then \( g \) can be computed by \( t \) players in the number-on-forehead model with communication at most \( \frac{\log w \cdot \log^2 n}{\gamma} \) in the number-on-forehead model.

Setting \( g \) to be the generalized inner-product function, Theorem 9.7 and Theorem 5.8 imply that any program with length \( \ell < O(n \log^2 n) \) that computes \( g' \) must have width that is at least \( 2^{\Omega(1)} \).

**Proof of Theorem.** Let \( \gamma \) be a constant that we will set to be small enough in the proof. Given a branching program of length \( \gamma n \log^2 n \) and width \( w \), partition the layers of the program into consecutive parts, in such a way that each part reads at most \( n/3 \) variables, and there at most \( 3\gamma \log^2 n \) parts.

Consider the bipartite graph where every vertex on the left corresponds to a part of the partition, and every vertex on the right corresponds to one of the \( n \) variables \( x_1, \ldots, x_n \) of \( g' \). Connect two vertices if the variable does not occur in the corresponding part in the partition. The degree of each vertex on the left is at least \( 2n/3 \), so the edge density is at least \( 2/3 \geq 1/2 \).

We shall prove:

**Claim 9.8.** One can find \( t \) disjoint sets \( Q_1, \ldots, Q_t \) on the left, and \( t \) disjoint sets of vertices \( R_1, \ldots, R_t \) on the right, such that

- Every vertex on the left is in some set \( Q_i \).
- For all \( i \), \( |Q_i| \leq \gamma \log n \), \( |R_i| \geq \sqrt{n} \).
- For all \( i \), every vertex of \( Q_i \) is connected to every vertex of \( R_i \) by an edge.

Figure 9.3: The bipartite graph defined by the branching program can be partitioned into cliques.

\(^{10}\) Babai et al., 1989; and Hrubes and Rao, 2015
Before proving the claim, let us see how to use it. Given an input \((x_1, \ldots, x_t)\) to \(g\), use the branching program for \(g'\) with inputs \((x, R_1, \ldots, R_k)\) where \(x|_{R_i} = x_i\) and \(x\) is zero outside \(\bigcup R_i\). Thus, \(g(x_1, \ldots, x_t) = g'(x, R_1, \ldots, R_k)\). The protocol for computing \(g\) proceeds as follows. Recall that player \(i\) sees \(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_t\). The player that corresponds to the first \(n/3\) steps of the branching program can simulate those steps. He uses \(\lceil \log w \rceil\) bits to announce the result of this simulation. Now the player that corresponds to the next \(n/3\) steps of the program can compute the output of the next part of the program and announce it, and so on. The overall communication is \(L \cdot \lceil \log w \rceil\), and the protocol indeed computes \(g\).

It only remains to prove the claim. We do this by repeatedly using Lemma 5.4. Take the subgraph of this bipartite graph induced by the first \(m = 8\sqrt{\gamma} \log n\) vertices on the left and the \(n\) vertices on the right, and apply Lemma 5.4 to find a set \(Q_1\) on the left and \(R_1\) on the right such that \(|Q_1| = 2\sqrt{\gamma} \log n, |R_1| \geq \sqrt{n}\) such that every vertex of \(Q_1\) is connected to every vertex of \(R_1\). We can do this because the edge density satisfies \(1/2 \geq 2 \cdot 2^{\sqrt{\gamma} \log n} / 8 \sqrt{\gamma} \log n\), and we shall set \(\sqrt{\gamma} \leq 1/(16 \log(4e))\).

Deleting the edges of \(Q_1 \times R_1\) from the graph, we can repeat the process to find pairs of disjoint sets \((Q_2, R_2), (Q_3, R_3), \ldots\) with the same property. In each step, we take \(m\) more vertices from the left and apply Lemma 5.4 to the resulting graph. Since this process can continue at most \(\frac{\gamma \log^2 n}{8 \sqrt{\gamma} \log n} \leq (\sqrt{\gamma}/8) \log n\) times, the number of edges we delete is at most \((\sqrt{\gamma}/8) \log n \cdot \sqrt{n}\), and so the graphs we are working with will always have edge density \(\geq 1/2\). In this way we end up with at most \((\sqrt{\gamma}/8) \cdot \log n\) parts in the partition. If there are fewer than \(m\) vertices available on the left, we put each remaining vertex in a separate set \(Q_i\). This could give us an additional \(8\sqrt{\gamma} \log n\) parts. \(\square\)

**Exercise 9.1**

Show that the formula that asserts that there cannot be a graph which both has a \(k\)-matching and a set of size \(k - 1\) that covers every edge requires an exponential number of inequalities to prove in the cutting planes proof system.

**Exercise 9.2**

Show that the formula that asserts that there cannot be a graph on \([n]\) which both has a path from 1 to \(n\) and a set \(S \subset [n]\) with \(1 \in S, n \notin S\) \(k\)-matching and a set of size \(k - 1\) that covers every edge requires an exponential number of inequalities to prove in the cutting planes proof system.
A data structure is a way to efficiently maintain access to data. Many well-known algorithms\(^1\) rely on efficient data structures to achieve their performance guarantees. Lower bounds on the performance of data structures are often proved by appealing to arguments about communication complexity. In this chapter, we explore some methods for proving lower bounds.

### Dictionaries

A dictionary is a data structure that maintains a subset \(S \subseteq [n]\), allowing us to add and delete elements from the set. We would also like to ask membership queries of the form—is \(i \in S\)?

The most straightforward implementation is to maintain a string \(x \in \{0, 1\}^n\) that is the indicator vector of \(S\). This allows us to add and delete elements, as well as support membership queries in time 1, but requires \(n\) memory cells.

A more efficient randomized alternative is to use hashing. Assume that we only perform \(m\) operations, and we permit the data structure to make an error with small probability. Then, for a parameter \(\epsilon\), we can pick a random function \(h : [n] \to [n']\) with \(n' = \lceil m^2 / \epsilon \rceil\). Now, we encode \(S\) using a string \(x \in \{0, 1\}^{n'}\), by setting \(x_j = 1\) if and only if there is an \(i \in S\) such that \(h(i) = j\). To add \(i\), set \(x_{h(i)} = 1\), and to delete \(i\), set \(x_{h(i)} = 0\). This data structure uses only \(n'\) cells of memory, and if at most \(m\) operations are involved, the probability that this data structure makes an error is at most \(\epsilon\).

It is a tantalizing open problem to prove that there is no data structure beating the indicator vector:

**Open Problem 10.1.** Find a data structure that can maintain a dictionary for \(S \subseteq [n]\) where each memory cell has \(O(\log n)\) bits, and all operations
Maintaining a Set of Numbers

Efficient algorithms for sorting numbers are key primitives in algorithm design, with countless applications. In many of these applications, we do not actually need to sort the numbers. It is enough to be able to query some information about the sorted list that can be computed very quickly.

Sort Statistics

Suppose we want to maintain a set \( S \subseteq [n] \) of \( k \) numbers, so that one can quickly add and delete numbers from the set, as well as compute the minimum of the set.

A trivial solution is to store the \( k \) numbers in a list. Then adding a number is fast, but finding the minimum might take \( k \) steps in the worst case. A better solution is to maintain the numbers in a heap, as in Figure 10.1. The numbers are stored on the nodes of a balanced binary tree, with the property that every node is at most the value of its children. One can add a number to the heap by adding it at a leaf, and bubbling it up the tree. One can delete the minimum by deleting the number at the root, inserting one of the numbers at a leaf into the root, and bubbling down the number. This takes only \( O(\log k) \) time for each operation.

These are operations need to be carried out efficiently in the execution the fastest algorithms for computing the shortest path connecting two vertices of a graph with edge weights.
Another solution is to maintain the numbers in a binary search tree, as in Figure 10.2. Each memory location corresponds to a node in the binary tree. Each leaf corresponds to an element of \( x \in [n] \) and stores a binary value indicating if \( x \in S \) or not. Each node above it maintains three numbers: the number of elements of \( S \) in the corresponding subtree, the minimum of \( S \) in that subtree, and the maximum of \( S \) in that subtree. If the subtree is empty, the minimum and maximum are set to 0. An element can be added to \( S \) or deleted from \( S \) in \( O(\log n) \) steps, by visiting all the memory cells that correspond to the ancestors of the element in the tree. One can also compute the \( i \)'th smallest element of \( S \) in \( O(\log n) \) steps, by starting at the root and moving to the appropriate subtree.

**Predecessor Search**

Suppose we want to maintain a set of numbers \( S \subseteq [n] \) and given \( x \in [n] \) we want to quickly determine the predecessor of \( x \), defined as

\[
P(x) = P_S(x) = \max\{y \in S : y \leq x\}.
\]

If we maintain the numbers using a binary search tree as in Figure 10.2, we can handle updates in \( O(\log n) \) time, and answer queries in time \( O(\log n) \). In fact the queries can be computed in time \( O(\log \log n) \).

One can improve the update time using van Emde Boas trees\(^2\). We sketch the solution. For simplicity of the description, assume \( \sqrt{n} \) is an integer. Let \( I_1, I_2, \ldots, I_{\sqrt{n}} \) be \( \sqrt{n} \) consecutive intervals of integers, each of size \( \sqrt{n} \). We store the maximum and the minimum elements of the set \( S \) in two sets. We recursively apply the whole scheme to store the set \( T_S = \{i : S \cap I_i \neq \emptyset\} \) using a van Emde Boas tree on a universe of size \( \sqrt{n} \). Finally, for each \( i \), we recursively store the set \( S \cap I_i \) using a van Emde Boas tree on a universe of size \( \sqrt{n} \). See Figure 10.3 for an illustration.

To compute \( P(x) \) from the van Emde Boas tree, let \( i \) be such that \( x \in I_i \). If \( S \cap I_i \) is empty, or \( x \) is less than the minimum of \( S \cap I_i \), then \( P(x) \) is the maximum of \( S \cap I_j \), where \( j < i \) is the largest index

\[^2\] van Emde Boas, 1975

![Figure 10.2: Maintaining numbers in a binary search tree with \( S = \{2, 3, 7, 8, 9, 10, 11, 12, 16\} \).](image)
for which $S \cap I_j$ is non-empty. To find $P(x)$, find the predecessor of the relevant interval in $T_S$, and output its maximum element. Otherwise, $P(x)$ is in $S \cap I_i$, and we can compute it recursively using the recursive structure that stores $S \cap I_i$. In either case, we only need to make one recursive call to a smaller data structure. Since the universe goes from $n$ to $\sqrt{n}$ after each recursive call, there can be at most $O(\log \log n)$ recursive calls before $P(x)$ is found. Similarly, one can add and delete numbers to the set $S$ using at most $O(\log \log n)$ operations.

Later in this chapter, we shall prove that van Emde Boas trees are essentially optimal when it comes to the predecessor search problem. The trees can also be used to query the minimum, median and maximum of a set in time $O(\log \log n)$. Surprisingly, we still do not know whether not this the optimal way to store a set $S$ to compute these parameters, though we can prove some lower bounds on restricted types of data structures.\(^3\)

Open Problem 10.2. Find a data structure that can maintain a set $S \subseteq [n]$, support the insertion and deletion of elements, and querying the minimum or median of the set in time $\ll \frac{\log \log n}{\log \log \log n}$. Alternatively, prove that there is no such data structure that can carry out all operations in time $O(1)$.

Union-find

The union-find data structure allows us\(^4\) to efficiently keep track of a partition $S_1, \ldots, S_k$ of $[n]$. The initial partition is the partition to $n$ sets of size 1. The data structure supports the union operation, which forms a new partition by replacing two sets $S_i, S_j$ in the partition by their union $S_i \cup S_j$. It also supports find queries of the form: $f(x)$, which returns a unique identifier for the set $S_i$ containing
This data structure has numerous applications; for example, it plays a key role in the fastest algorithms for computing the minimum spanning tree of a graph.

Here is the high-level scheme for an implementation of a union-find data structure, with operations that take time $O(\log n)$. See Figure 10.4 for an illustration. The data structure stores a table with $n$ elements, each consisting of $O(\log n)$ bits. The idea is to represent each set $S_i$ in the partition by a rooted tree $T_i$ with $|S_i|$ nodes. The nodes of $T_i$ are labelled by all elements of $S_i$. The edges of the tree $T_i$ are directed towards the root. Each element $x \in [n]$ has an entry in the table that stores a pointer to its parent, as well as the height of the subtree rooted at $x$. If $x$ is the root of the whole tree, then $x$ points to itself. The find operation with input $x \in [n]$ follows the pointer from $x$ to the root of the tree it belongs to, and outputs the name of the root. The union operation for $S_i, S_j$ merges the corresponding trees $T_i, T_j$ by making the root of the shorter tree (breaking ties arbitrarily) $T_i$ a child of the root of taller tree $T_j$, and adjusting the heights appropriately. One can show that this ensures that no tree has height more than $5O(\log n)$, and so all operations take time at most $O(\log n)$.

**Approximate Nearest Neighbor Search**

In the nearest neighbor search problem, we are given a set $S \subseteq \{0,1\}^d$ of size $n$, and wish to store it, so that one can quickly compute the nearest neighbor $N(x)$ of a query $x \in \{0,1\}^d$. The nearest neighbor is the element $y \in S$ minimizing the hamming distance $\Delta(x,y)$.

\[ \Delta(x,y) = \sum_{i=1}^{d} \mathbb{1}_{x_i \neq y_i} \]

Figure 10.4: Maintaining a partition of the universe into sets using the union-find data structure. Each cell is associated with an element of the universe, and stores a pointer to the cell corresponding to its parent, as well as the height of the subtree rooted at the element. The result of merging two sets is shown.

Later, we shall show that this solution is essentially optimal.
Typically \( n \gg d \).

One can always store the set using \( nd \) bits, and answer the query by querying all \( nd \) bits. We could also store a table with \( 2^d \) entries, each with \( n \) bits, recording the response for every possible query. Here we sketch the idea for a scheme to approximate the nearest neighbor. The key idea is to use locality sensitive hashing—hashing that is useful to identify the Hamming weight of a string.

**Theorem 10.3.** For every constant \( \epsilon > 0 \), there is a data structure that can solve the nearest neighbor search problem by storing the set using a table of size \( s = \log d \cdot (n \log \log d)^{O(1/\epsilon^2)} \), where each cell contains \( w = d + 1 \) bits, and the algorithm makes \( t = O(\log \log d) \) queries to answer queries.

A key step in the design of the data structure involves a method to hash strings so that their hamming distance can be verified. For a binary vector \( x \), let \( |x| \) denote the number 1’s in \( x \). For a parameter \( 0 \leq \delta \leq 1 \), and a number \( r \), let \( Z \) be a random \( r \times d \) matrix, where each entry of \( Z \) is independent and

\[
Z_{ij} = \begin{cases} 
1 & \text{with probability } \delta, \\
0 & \text{with probability } 1 - \delta.
\end{cases}
\]

We shall show that for \( \delta \) chosen appropriately, \( Zx \pmod{2} \) and \( Z_y \pmod{2} \) can be used to certify the Hamming distance \( \Delta(x, y) \). We start by proving the following lemma:

**Lemma 10.4.** For every \( k, \gamma, \epsilon > 0 \), there is an \( r = O(\log(1/\gamma)/\epsilon^2) \), bias \( \delta > 0 \) for the entries of \( Z \), and an interval \( T \), so that for every \( d \times 1 \) column vector \( w \) with \( 0/1 \) entries:

- If \( k \leq |w| \leq k(1 + \epsilon) \), then \( \Pr [|Zw| \in T] \geq 1 - \gamma \).
- If \( k/(1 + \epsilon) > |w| \) or \( |w| > k(1 + \epsilon)^2 \), then \( \Pr [|Zw| \in T] \leq \gamma \).

The lemma allows us to efficiently check whether the Hamming distance of \( x, y \) is close to \( k \). Indeed, if \( Z \) is sampled as above, then \(|Zx - Z_y \pmod{2}| = |Z(x + y) \pmod{2}|\) is enough to verify that the hamming distance \( \Delta(x, y) \) is close to \( k \).

Before proving the lemma, let us see how to use it to obtain an efficient data structure for approximate nearest neighbor search. For a given constant \( \epsilon > 0 \), we set \( \gamma \) to be a small multiple of \( \frac{1}{n \log \log d} \). Apply Lemma 10.4 \( \log_{1+\epsilon} d \leq O(\log d) \) times, with

\[
k = 1, (1 + \epsilon), (1 + \epsilon)^2, \ldots
\]

up to the point that \( k \leq d \). For each value of \( k \), sample a matrix \( Z^k \) as promised by the lemma. For each \( r \times 1 \) binary vector \( q \), the
data structures stores a cell that contains \( y \in S \), if and only if \( |Z_k(y + q) \pmod{2}| \) \( \in T \). The space of the data structure is thus
\[
s = O(\log d \cdot 2^r) = \log d \cdot (n \log \log d)^{O(1/\epsilon^2)}.
\]

To find the nearest neighbor of a given vector \( x \), the algorithm uses binary search. It first sets \( k \) to be the middle value and checks if there is an element \( y \in S \) at distance \( k \). If there is, it looks at the middle of the left half, and so on. This involves at most \( O(\log \log d) \) queries. Since there are only \( n \) strings in \( S \), the probability of making an error is at most \( n \gamma \) in each step. The final probability of making an error is at most \( n \cdot \gamma \cdot \log \log d \leq 1/3 \), if \( \gamma \) is chosen to be a small constant multiple of \( 1/n \log \log d \).

It only remains to prove the hashing lemma:

**Proof of Lemma 10.4.** Let \( z \) be a single row of \( Z \). Then we claim:
\[
\Pr_z [(\langle z, w \rangle = 0 \pmod{2})] = \frac{1 + (1 - 2\delta)|w|}{2}.
\]

Indeed, if we set \( \gamma = \Pr_z [(\langle z, w \rangle = 0 \pmod{2})] \), we have:
\[
(2\gamma - 1) = \mathbb{E}_z [(-1)^{\langle w, z \rangle}] = \mathbb{E}_z \left[ (-1)^{\sum_{i=1}^d w_i z_i} \right] = \mathbb{E}_z \left[ \prod_{i=1}^d (-1)^{w_i z_i} \right] = \prod_{i=1}^d \mathbb{E}_z [(-1)^{w_i z_i}] = (1 - 2\delta)|w|,
\]
so we get \( \gamma = \frac{1 + (1 - 2\delta)|w|}{2} \). Now let \( \delta \) be set so that
\[
b(\delta, k) = \frac{1 + (1 - 2\delta)^k}{2} = 2/3,
\]
so \( (1 - 2\delta)^k = 1/3 \). We want to show that \( Zw \) is enough to verify that \( |w| \) is close to \( k \). Observe that
\[
|b(\delta, k) - b(\delta, k(1 + \epsilon))| = |(1 - 2\delta)^k - (1 - 2\delta)^{k(1 + \epsilon)}|
= \frac{1}{3} - \left( \frac{1}{3} \right)^{1+\epsilon}
\geq \frac{1}{3} \cdot (1 - (1 - \epsilon \cdot (\log 3)/2))
\geq \epsilon \cdot \frac{\log 3}{6} \geq \Omega(\epsilon).
\]

Similarly, one can show that
\[
|b(\delta, k(1 + \epsilon)^d) - b(\delta, k(1 + \epsilon)^{d+1})| = (\frac{1}{3})^{(1+\epsilon)^d} - (\frac{1}{3})^{(1+\epsilon)^{d+1}} \geq \Omega(\epsilon),
\]
for any $a \in \{-2, -1, \ldots, 2, 3\}$. Let

$$T = \left[ b(\delta, k(1 + \epsilon)^{-1}) \cdot \frac{r}{d}, b(\delta, k(1 + \epsilon)^{2}) \cdot \frac{r}{d} \right].$$

Then, by the Chernoff-Hoeffding bound, if

$$k \leq |w| \leq k(1 + \epsilon),$$

we have

$$\Pr [ |Zw \pmod{2}| \in T ] \geq 1 - e^{-\Omega(\epsilon^2 r)}.$$

Similarly, if $|w| \geq k(1 + \epsilon)^3$ or $|w| \leq k(1 + \epsilon)^{-2}$, the probability that the number of non-zero inner products is in the same interval $I$ is at most $e^{-\Omega(\epsilon^2 r)}$. Choosing $r = O(\log(1/\gamma)/\epsilon^2)$ ensures that the probability of making a mistake for any $y \in S$ is at most $\gamma$. \hfill \square

**Lower Bounds on Static Data Structures**

A static data structure specifies a way to store data in memory, and to answer queries about the data, without the ability to update the data. There are three main parameters that we seek to optimize:

- **Number of cells $s$**: This is the total number of memory cells used to store the data.
- **Word size $w$**: This is the number of bits in each memory cell.
- **Query time $t$**: This is the number of cells that need to be accessed to answer a query on the data.

Ideally, we would like to minimize all three parameters.

The primary method for proving lower bounds on the parameters of static data structures is via communication complexity. In a nutshell, efficient data structures lead to efficient communication protocols. Say we are given a data structure for a particular problem. We define the corresponding data structure game as follows: Alice is given a query to the data structure, and Bob is given the data that is stored in the data structure. Using the data structure, the communication problem can be solved by a $2t$-round protocol. Bob encodes the data using the data. Alice and Bob then simulate the execution of the data structure algorithm—Alice sends $\log s$ bits to indicate the name of the memory cell she wishes to read, and Bob responds with $w$ bits encoding the contents of the appropriate cell. After $t$ repetitions steps, Alice and Bob know the result of the computation.
Lemma 10.5. If there is a data structure of size $s$, word size $w$, and query time $t$ for solving a particular problem, then there is a deterministic protocol solving the related communication game with $2t$-rounds. In each round Alice sends $\log s$ bits and Bob responds with $w$ bits.

Appealing to lower bounds in communication complexity gives us lower bounds on the parameters of the data structure. Let us explore some examples.

Set Intersection

Suppose we wish to store an arbitrary subset $Y \subseteq [n]$ so that on input $X \subseteq [n]$ one can quickly compute whether or not $X \cap Y$ is empty. There are several solutions one could come up with:

- We could store $Y$ as string of $n$ bits broken up into words of size $w$. This would give the parameters $s = t = \lceil n/w \rceil$.

- We could store whether or not $Y$ intersects every potential set $X$. This would give $s = 2^n$ and $w = t = 1$.

- For every subset $V \subseteq [n]$ of size at most $p$, we could store whether or not $Y$ intersects $V$. Since $X$ is always the union of at most $\lceil n/p \rceil$ sets of size at most $p$, this gives $s = \sum_{i=0}^{p} \binom{n}{i}$, $w = 1$, and $t = \lceil n/p \rceil$.

On the other hand, since every data structure leads to a communication protocol for computing set disjointness, for which the communication must be at least $n + 1$, we have:

Theorem 10.6. Any data structure that solves the set intersection problem must have $t \cdot (\lceil \log s \rceil + w) \geq n + 1$.

Lopsided Set Intersection

In practice, the bit complexity of the queries is often much smaller than the amount of data being stored. In the $k$-lopsided set intersection problem, the data structure is required to store a set $Y \subseteq [n]$. A query to the problem is a set $X \subseteq [n]$ of size $k \ll n$. The data structure must compute whether or not $X$ intersects $Y$.

When $k = 1$, we can get $s = \lceil n/w \rceil$ and $t = 1$, and no better parameters are possible. The problem becomes more interesting when $k > 1$. We can get a solution with $s = \lceil (\binom{n}{k})/w \rceil$ and $t = 1$ by storing whether or not $Y$ intersects each set of size $k$.

Theorem 1.22 yields the following lower bound:

\[ t \cdot \left( \log s + \log w \right) \geq n + 1. \]
Theorem 10.7. In any data structure solving the lopsided set intersection problem,
\[ t(\log s + w) \geq \frac{n}{2(t \log s/k + 1)} = \frac{n}{s^{1/k} + 1}. \]

For example, if \( s \leq (\frac{n}{k})^{k/(2t)} \) then
\[ t \geq \frac{n}{s^{1/k} + 1} \geq \frac{n}{\sqrt{n/k} + 1} \geq \frac{\sqrt{n}}{2}. \]

The Span Problem

In the span problem\(^7\), the goal is to store \( n/2 \) vectors \( y_1, \ldots, y_{n/2} \in \mathbb{F}_2^n \), with \( n \) even. A query is a vector \( x \in \mathbb{F}_2^n \). The data structure must quickly compute whether or not \( x \) is a linear combination of \( y_1, \ldots, y_{n/2} \).

Theorem 10.8. In any static data structure solving the span problem,
\[ tw \geq n^2/4 - t \log s \cdot (n + 1) - n \log n. \]

For example, if \( s < 2^{n/8t} \), then \( tw = \Omega(n^2) \).

Predecessor Search

In the predecessor search problem, the data structure is required to encode a subset \( S \subseteq \mathbb{Z}_n \) of size at most \( k \). The data structure should also be able to compute the predecessor \( P(x) \) of any element \( x \in \mathbb{Z}_n \).

We have seen that there is a data structure that can achieve this in time \( t \leq O(\log \log n) \). Here we show that this bound is essentially tight\(^8\):

Theorem 10.9. Any static data structure solving the predecessor search problem with \( s \leq \text{poly}(k) \) must have \( t \geq \Omega\left(\frac{\log k}{\log(w \log k)}\right) \) or \( w \geq 2^{\left(\frac{\log n}{\log \log x}\right)} \).

Proof. We prove the theorem by appealing to a lower bound for the tree pointer chasing problem (Theorem 7.13). For a large constant \( c \), say we have an input to the tree pointer chasing problem on a tree \( T_{a,b} \) of depth \( d \) with
\[ a = \lceil 9w \cdot \log^2 k \rceil, \quad b = \lceil 9 \log s \cdot \log^2 k \rceil, \quad d = \left\lfloor \frac{\log k}{\log a} \right\rfloor. \]

We show how Alice and Bob can transform their inputs into \( x \in \{0, 1, \ldots, n - 1\} \) and \( S \subseteq \{0, 1, \ldots, n - 1\} \) for the predecessor search problem without communicating. The transformation guarantees that the predecessor of \( x \) in \( S \) determines the correct output of the tree.
pointer chasing problem. We further ensure that \( n \leq (a + b)^{bd} \) and \(|S| \leq a^d \leq k\).

Next we describe how to carry out the reduction. If \( d = 1 \) and Alice knows the edge coming out of the root, we set \( S = \{0, 1, \ldots, a - 1\} \) and \( x \in \{0, 1, \ldots, a - 1\} \) to be the name of the child that is connected to the root in Alice’s input. If Bob knows the edge coming out of the root, we set \( S = \{i\} \), where \( i \in \{0, \ldots, b - 1\} \) corresponds to the leaf of the tree that is connected to the root, and we set \( x = b - 1\).

In either case, \( S \) is a set of size at most \( a \), defined on a universe of size at most \( a + b \), and the predecessor of \( x \) in \( S \) determines the output of the tree pointer chasing problem.

If the tree is of depth \( d > 1 \), with Alice knowing the edge coming out of the root, we first inductively compute \( x_0, \ldots, x_{a-1} \) and \( S_0, \ldots, S_{a-1} \), which are the inputs to predecessor search determined by the \( a \) subtrees of depth \( d - 1 \) that lie just below the root. If \( i \in \{0, 1, \ldots, a - 1\} \) corresponds to the edge coming out of the root, we set

\[
x = i \cdot m + x_i,
\]

and

\[
S = \bigcup_{i=1}^{a} \{i \cdot m + y : y \in S_i\},
\]

where \( m = (a + b)^{bd-1} \). The new universe is of size at most \( a \cdot (a + b)^{bd-1} \leq (a + b)^{bd} \), and \(|S|\) is at most \( a \cdot a^{d-1} = a^d \).

Otherwise, when the edge touching the root belongs to Bob, inductively compute \( x_0, \ldots, x_{b-1} \) and \( S_0, \ldots, S_{b-1} \) using the \( b \) subtrees of depth \( d - 1 \). If \( i \in \{0, 1, \ldots, b\} \) corresponds to the edge touching the root, we set

\[
x = \sum_{j=0}^{b-1} x_j \cdot m^{b-j-1},
\]

and

\[
S = \left\{ \sum_{j=0}^{i-1} x_j \cdot m^{b-j-1} + y \cdot m^{b-i-1} : y \in S_i \right\}.
\]

This corresponds to writing \( x \) in base \( m \) with the digits \( x_0, x_1, x_2, \ldots, x_{b-1} \), and setting the \( y \)’th element of \( S \) to be \( x_0, x_1, \ldots, x_{i-1}, y, 0, \ldots, 0 \), where \( y \in S_i \).
By the definition of the tree pointer chasing problem, Bob knows \( x_0, \ldots, x_{i-1} \), so he can compute the set \( S \). The size of the universe in this case is at most \( (a + b)^{bd-1} b \leq (a + b)^{bd} \), and \( |S| \leq |S_i| \).

In both cases, the predecessor of \( x_i \) in \( S \) determines the relevant predecessor of \( x_i \) in \( S_i \), and hence determines the output of the tree pointer chasing input.

The reduction above implies that any data structure with \( t < d/2 \) queries for the predecessor search problem gives a communication protocol for the tree pointer chasing problem where Alice sends \( \log s \) bits in each round and Bob responds with \( w \) bits in each round, and the total number of rounds is less than \( d \).

By Theorem 7.13, the protocol can succeed with probability at most

\[
\frac{1}{2} + 2(d - 1) \left( \sqrt{\frac{\log s}{b}} + \frac{\sqrt{w}}{a} \right) \\
\leq \frac{1}{2} + 2 \log k \left( \frac{1}{3 \log k} + \frac{1}{3 \log k} \right) < 1,
\]

by the choice of \( a, b \) and \( d \). So, the protocol for tree pointer chasing must make an error, as long as \( n \geq (a + b)^{bd} \). This implies that \( n < (a + b)^{bd} \) or

\[
\log \log n \leq d \log b + \log \log(a + b) \leq O(\log k \log \log k + \log \log w),
\]

as required.

**Approximate Nearest Neighbor Search**

Here we prove a lower bound for the nearest neighbor search problem. The lower bound is proved by appealing to the communication complexity of the lopsided disjointness problem. In Chapter 1, we proved such a lower bound for deterministic protocols. To a prove a lower bound matching the upper bound for approximate nearest neighbor search that we discussed earlier in this chapter, we need to appeal to the following randomized lower bound for lopsided disjointness. Suppose Alice is given a set \( X \subseteq [r] \) of size \( \ell \), and Bob is given a set \( Y \subseteq [r] \), then we have:\

\footnote{Andoni et al., 2006}

**Theorem 10.10.** For every \( \beta > 0 \), if there is a protocol solving lopsided disjointness as above for \( n \) large enough, with error \( \delta \), then either Alice sends \( \beta \ell \log r \) bits, or Bob must send \( r^{1-2\beta} \) bits.

We can use the above theorem to prove:\

**Theorem 10.11.** For any \( \beta > 0 \) the following holds. If there is a data structure solving the approximate nearest neighbor search problem with
Unlike for static data structures, not all of methods for proving lower bounds on dynamic data structures involve reductions to communication complexity. Nevertheless, intuitions from understanding the role of information play a role here as well.

A dynamic data structure is one that allows to both update and query on the data. The union-find data structure, the van Emde Boas tree, heaps and binary search trees are all examples of dynamic data structures. In this section, we develop methods for proving lower bounds on such data structures.

Dynamic data structures have four main parameters:

Number of cells $s$: This is the total number of memory cells used to store the data.
Word size $w$: This is the number of bits in each memory cell.

Query time $t_q$: This is the number of cells that need to be accessed to answer a query on the data.

Update time $t_u$: This is the number of cells that need to be accessed to update the data.

We allow data structures to be randomized, making decisions using random coin tosses, and this may induce errors. We say that the error of the data structure is $\epsilon$ if for every sequence of updates followed by a single query, the probability that the query is computed correctly is at least $1 - \epsilon$.

Prefix Sum and Maintaining a Sorted List of Numbers

Our first lower bound for dynamic data structures concerns a very basic problem. Although the proof does not involve communication complexity, it does use the concepts of entropy that we developed in Chapter 6. We include it here, because it is of fundamental importance.

Suppose we want to maintain a set $S \subseteq [n]$ so that we can add and delete elements from the set, as well as compute the $i$’th element of the set in the sorted order. We prove:

Theorem 10.12. Any data structure maintaining a sorted list of numbers with error at most $1/3$ for $m \geq 2n^2$ operations must satisfy:

$$t_q \cdot \log (t_u w) \geq \Omega(\log n).$$

In particular, if $t_u, w$ are polylogarithmic in $n$, then Lemma ?? asserts that $s$ can be assumed to be polynomial in $m$. This gives $t_q \geq \Omega(\log n / \log \log n)$.

We first prove a lower bound for a different task called prefix sum. Suppose we want to maintain a binary string $x \in \{0, 1\}^n$, which is initially 0, allow to update the value of $x_i$ with $1 - x_i$ for any $i \in [n]$, and query $q(i) = \sum_{j=1}^{i} x_j$. We prove:

Theorem 10.13. Any data structure correctly computing prefix sum of an $n$ bit string with error at most $1/3$ for $m \geq n \log n$ operations satisfies

$$t_q \cdot \log (t_u w)) \geq \Omega(\log n).$$

Before proving Theorem 10.13, we show how to use it to prove a lower bound for maintaining a sorted list of numbers. To do this, we assume we are given a data structure for maintaining a sorted list of numbers, and show how to use it to solve the prefix sum problem.
We initialize our set of numbers to be
\[ S = \{ j \in [n^2] : j \neq 1 \pmod{n} \}. \]

We also maintain the indicator vector of \( S \). This allows us to add and
delete elements to \( S \), and check whether \( i \in S \) in constant time, and
word size, and space \( n^2 \). Whenever we wish to flip the value of \( x_i \),
we check if \( (i-1)n+1 \in S \). If \( (i-1)n+1 \in S \), we delete it from \( S \).
If \( (i-1)n+1 \notin S \), we add it to \( S \). Let \( y_i \) denote the \((i-1)n+1\)'st
element of \( S \) in sorted order. Then observe that
\[ y_i = (i-1)n+1 + (i - q(i)). \]

So, if we know \( y_i \), we know \( q(i) \). We can find \( y_i \) using the given data
structure. This simulates \( n \log n \) operations of the prefix sum data
structure using at most \( n^2 + n \log n \leq 2n^2 \) operations of the given
data structure. The space of the resulting data structure has been
increased by \( n \), but since the data structure must have space at least
\( n/w \), this does not affect the bounds in the theorem.

Proof of Theorem 10.13. To prove the lower bound, we use a particular
distribution on operations. By fixing the randomness used by the
data structure, we can assume that it is deterministic, and makes an
error on at most \( \epsilon \) fraction of the random sequences of updates and
query.

We set \( n = k^r \) with \( k \leq O(t_n w) \) a parameter that we shall set in the
proof. For \( j = 0, 1, 2, \ldots, r \), define
\[ S_j = \{ a \cdot k^j + 1 : a \in \{0, 1, \ldots, k^{r-j} - 1\} \}. \]

\( S_j \) consists of \( k^{r-j} \) uniformly spaced numbers in \([n]\).

Consider the following sequence of random updates that consists
of \( r + 1 \) rounds. In the \( j \)'th round we pick a uniformly random subset
\( T_j \subseteq S_j \), independently of other choices. For each \( i \in T_j \), we flip the
value of \( x_i \) using the data structure. At the end of these \( r + 1 \) rounds of
updates, we pick a uniformly random coordinate \( L \in [n] \) and query
\( q(L) \) using the data structure.
We shall compute the expected number of queries the data structure needs to make to correctly compute $q(L)$. Say that a cell of the data structure belongs to round $j$ if it was last touched during the updates of round $j$. We shall prove that for every round $j$, the probability that a cell belonging to round $j$ is queried during the final query operation is at least $\Omega(1)$. By linearity of expectation, this implies that the expected number of cells queried is

$$
E[q] \geq \Omega \left( \frac{\log n}{\log(tuw)} \right),
$$

as required.

The key step in the proof involves the subadditivity of entropy. Intuitively, all the information about $T_j$ can only be encoded by cells belonging to rounds $j'$ with $j' \geq j$. However, the number of cells belonging to rounds $> j$ is much smaller than the entropy of $T_j$, because the total number of operations performed by the data structure in these rounds is much smaller than $|S_j|$. So, even accounting for the cells belonging to rounds $> j$, the algorithm must read a cell belonging to round $j$ if it wishes to learn access information about $T_j$. We show that it must learn information about $T_j$ if it is to answer the query.

Now, let us make this intuition more formal. For the rest of the proof, fix a specific round $j$. Let $A$ be the indicator vector of $T_j$. Let $I$ be the maximum element in $S_j$ that is at most $L$. $I$ is uniformly distributed in $S_j$, and $I$ can be interpreted as the name of a coordinate of $A$. For ease of notation, let $R = A_I$, and let $D$ be a random variable encoding $L, A_{<I}, T_1, \ldots, T_{j-1}, T_{j+1}, \ldots, T_r$ and the contents and locations of all the cells accessed by updates in rounds after the $j$'th round.

The key claim of the proof shows that if $k$ is large enough, the data structure does not learn much information about $R$ from the cells belonging to the rounds $> j$:

**Claim 10.14.** $H(R|D) \geq 1 - \frac{2tuw}{k}$.

Before proving Claim 10.14, let us use it to prove that a cell belonging to round $j$ must be accessed with probability $\Omega(1)$. We set $k = O(tuw)$ to be large enough so that the claim implies that $H(R|D)$ is very close to 1. Define

$$Q = \begin{cases} 
1 & \text{if the data structure queries a cell that belongs to round } j, \\
0 & \text{otherwise.}
\end{cases}$$

By Claim 10.14, we have:

\[
1 - \frac{2tuw}{k} \leq H(R|D) \leq H(QR|D) = H(Q|D) + H(R|DQ) \leq H(Q) + H(R|DQ).
\]

by the chain rule

by subadditivity

Suppose \( Q = 1 \) with probability \( \gamma \). When \( Q = 0 \), the output of the algorithm is determined by \( D \), since all the cells that are read in order to compute \( q(L) \) are determined by \( D \). Define

\[
E = \begin{cases} 
1 & \text{if the data structure makes an error}, \\
0 & \text{otherwise}. 
\end{cases}
\]

When \( Q = 0 \), \( R \) is determined by \( E \) and \( D \). Thus,

\[
1 - \frac{2tuw}{k} \leq H(Q) + H(R|DQ) \leq h(\gamma) + \gamma \cdot H(E|D, Q = 0) \leq h(\gamma) + \gamma \cdot H(E|Q = 0)
\]

\[
\leq h(\gamma) + \gamma + h(\epsilon/(1 - \gamma)).
\]

If \( k \) is set to be a large multiple of \( tuw \), the left hand side is close to 1. If \( \epsilon, \gamma \) are small, the right hand side is close to 0. Thus we must have \( \gamma = \Omega(1) \).

It only remains to prove Claim 10.14.

**Proof of Claim 10.14.** Let \( A \in \{0, 1\}^{kr-j} \) be the indicator vector for the set \( T_j \subseteq S_j \); namely \( A_i = 1 \) if and only if the \( i \)'th element of \( S_j \) is included in \( T_j \). \( A \) is a uniformly random binary string. Let

\[
B = (T_1, T_2, \ldots, T_{j-1}, T_{j+1}, \ldots, T_r).
\]

Let \( C \) denote the locations and contents of all the cells that belong to rounds larger than \( j \). The number of cells belonging to rounds larger than \( j \) is at most

\[
t_u \cdot \sum_{j'=j+1}^{r} k^{r-j'} = t_u \cdot k^{r-j} - 1 \leq 2tu k^{r-j-1}.
\]

Given \( B \), contents of \( C \) can be described by specifying the contents of all cells that are touched during the updates \( T_{j+1}, \ldots, T_r \). Thus, we have

\[
H(C|B) \leq 2k^{r-j} - 1 \cdot t_u \cdot w,
\]
which implies

\[
H(A|BC) \geq H(A|B) - H(C|B) \\
\geq k^{r-j} - 2k^{r-j-1}t_uw \\
\geq k^{r-j} \left( 1 - \frac{2t_uw}{k} \right).
\]

by subadditivity of entropy

Now, by the chain rule,

\[
\sum_{i=1}^{k^{r-j}} H(A_i|A_{<i}BC) \geq k^{r-j} \left( 1 - \frac{2t_uw}{k} \right),
\]


Graph Connectivity

Efficient algorithms that maintain graphs are widely used in computer science. These provide another source of basic data structure questions. The union-find data structure that we discussed earlier in this chapter is a very useful tool in this context.

Suppose we want to implement a static data structure that can store a graph on \( n \) vertices using a small amount of space such that we can later quickly answer whether two given vertices are connected or not. A trivial solution is to store the adjacency matrix of the graph, and then perform a breadth first search using this matrix. This would take \( n^2 \) bits to encode the graph, but might involve making \( n^2 \) probes to answer the query.

A better solution is to store a vector in \( |n|^n \) which stores the name of the connected component that each vertex belongs to. This can be stored with \( n \) words, each of size \( \log n \), and now connectivity for two vertices can be computed by two queries to the data structure.

If we want to maintain the graph using a dynamic data structure allowing for the addition of edges and querying whether or not two vertices are connected, we can use the union-find data structure to maintain the connected components of the graph while allowing edges to be added, we can do this using the union-find data structure as in Figure 10.4. At each point in time, the partition of the vertices represents the current connected components. When a new edge is added, if the two vertices of the edge are contained in the same connected component, nothing needs to be done. Otherwise, the two connected components are merged with the union operation.
Static Connectivity in Sparse Directed Graphs

We have seen that one can solve the static graph connectivity problem on \( n \) vertices with \( t = 2, s = n, w = \log n \). Here we consider the same problem in directed graphs.

A trivial solution is to store whether or not \( u \) is connected to \( v \) for every pair \((u, v)\). This gives \( t = 1, s = n(n - 1), w = 1 \). In general, one cannot do much better than this.

Indeed, let \( A \) and \( B \) be two disjoint sets of vertices, each of size \( n/2 \), and consider the graphs where all edges go from \( A \) to \( B \). There are \( 2^{n^2/4} \) such graphs, and the data structure must distinguish all of them, since the queries to the data structure can reconstruct all edges in such graphs. Thus we must have \( sw \geq n^2/4 \).

The problem becomes more interesting when we consider sparse graphs. What if we are guaranteed that every vertex only has \( O(\log n) \) edges coming out of it? Is there a data structure solving graph connectivity with \( s = O(n \log n) \) and \( t, w = O(1) \)? Here we use communication complexity to show that such a data structure does not exist:

**Theorem 10.15.** In any static data structure solving the directed graph connectivity problem on graphs with at most \( nw \) edges with word-size \( w \), we must have

\[
t \geq \Omega \left( \frac{\log n}{\log \log n \cdot \log \frac{sw \log n}{n}} \right).
\]

In particular, Theorem 10.15 implies that if \( t \) is a constant, then \( s = n^{1+\Omega(1)} \).

**Proof.** To prove Theorem 10.15, we consider a special family of sparse directed graphs: subgraphs of the butterfly graph. A \((d, w)\) butterfly graph is a layered directed graph where each vertex corresponds to a tuple \((i, u) \in [d + 1] \times [w]^d\). Here \( w \) is set to be the word size of the data structure. Each layer has a vertex for each string of length \( d \) from the alphabet \([w]\). Each vertex in the \( i \)th layer is connected to exactly \( w \) vertices from the \( i + 1 \)'st layer that agree in all but the \( i \)'th coordinate. See Figure 10.7 for an example. This graph has the feature that there is a unique path from every vertex in the first layer to a vertex in the last layer. To go from \((w_1, \ldots, w_d)\) to \((w'_1, \ldots, w'_d)\), the only path is

\[
(w_1, \ldots, w_d) \rightarrow (w'_1, w_2, \ldots, w_d) \\
\quad \rightarrow (w'_1, w'_2, w_3, \ldots, w_d) \\
\quad \rightarrow \ldots \rightarrow (w'_1, w'_2, \ldots, w'_d)
\]

To describe the graph more rigorously, for any \( u \in [w]^d \), let \( u_{-i} \in [w]^{d-1} \) denote \( u \) after deleting the \( i \)'th coordinate: \( u_{-i} = \ldots \)
(u_1, u_2, \ldots, u_{i-1}, u_{i+1}, \ldots, u_d). There is an edge from (i, u) to (j, v) if and only if j = i + 1 and u_{-i} = v_{-i}.

Our proof of the lower bound from Chapter 1 implies:

**Theorem 10.16.** Suppose Alice is given a string x ∈ [w]^s, and Bob is given a sequence of sets Y_1, \ldots, Y_k ⊆ [w]. If there a protocol that determines whether or not there is an i such that x_i ∈ Y_i, with Alice sending a bits and Bob sending b bits, then a + b ≥ \frac{\log n}{2^{\log w}}.

We shall show how Alice and Bob can use the data structure to solve the lopsided disjointness problem with inputs x, Y as in Theorem 10.16. We set k = dw^{d-1}, and w to be the same parameter as in the butterfly graph. The size of the graph is thus n = wk. This gives d = \Omega(\log n / \log w).

Alice uses x ∈ [w]^k to construct a subgraph G of the butterfly graph. Each coordinate of x is associated with a tuple (i, u_{-i}). The edge from (i, u) to (i + 1, v) is included in G if and only if v_i + u_i = x_{(i,u_{-i})} (mod w). Observe that G consists of w^d vertex disjoint paths from the first layer of the graph to the last layer. Similarly, Bob uses Y_1, \ldots, Y_k to construct a subgraph H of the butterfly graph. The edge from (i, u) to (i + 1, v) is included in H if and only if v_i + u_i ∉ Y_{(i,u_{-i})} (mod w).

Now, if there is an i for which x_i ∈ Y_i, this corresponds to w edges that are present in G, but not in H. Thus, Alice and Bob can determine if such an i exists by answering w^d connectivity queries. Actually, it is enough to answer just w^{d-1} connectivity queries to compute disjointness— for every vertex u, Alice only needs to know whether the paths that start at (1, u_{-1}) in H are included in G or not.

Alice and Bob can simulate the execution of these w^{d-1} queries on the data structure in parallel. In each round, Alice sends Bob \lceil \log (\frac{d}{w^{d-1}}) \rceil bits to indicate the cells that she needs to look up for each of her queries. Bob responds with w \cdot w^{d-1} = w^d bits to describe the contents of those cells. This simulation gives a protocol where Alice sends a = t \lceil \log (\frac{d}{w^{d-1}}) \rceil bits, and Bob responds with b = tw^d bits. Theorem 10.16 implies that

\[2t \log \left( \frac{s}{w^{d-1}} \right) + tw^d \geq \frac{d w^d}{2^t \log (\frac{s}{w^{d-1}}) / (d w^{d-1}) + 1}.
\]

Since (\frac{d}{w^{d-1}}) \geq \left( \frac{es}{w^{d-1}} \right)^{w^{d-1}}, we have \log (\frac{d}{w^{d-1}}) \geq w^{d-1} \cdot \log \frac{es}{w^{d-1}}.

This simplifies the inequality to:

\[2tw^{d-1} \cdot \log \frac{es}{w^{d-1}} + tw^d \geq \frac{d w^d}{\left( \frac{es}{w^{d-1}} \right)^{t/d} + 1}.
\]

\[\Rightarrow t \left( \log \frac{es}{w^{d-1}} + 1 \right) \geq \frac{d}{\left( \frac{es}{w^{d-1}} \right)^{t/d} + 1}.
\]
As we saw at the beginning of the chapter, the union-find data structure can be used to solve this problem with parameters $s = \Theta(n)$ and $w = t_u = t_q = \Theta(\log n)$.

Fredman and Saks, 1989

Setting $w = \log n$, $\epsilon < 1/2$ to be constant, and $t_u = \text{polylog}(n)$, we get that $t_q \geq \Omega(\log n / \log \log n)$.

Lower bound for Dynamic Graph Connectivity

In the dynamic graph connectivity problem, the data structure is required to maintain a graph on the vertex set $[n]$, supporting addition of edges, as well as queries that compute whether or not two vertices are connected in the graph.

Here we prove\textsuperscript{14}:

**Theorem 10.17.** Any data structure solving the graph connectivity problem with error at most $1/3$ for $m \geq n + 1$ operations must satisfy:

\[ t_q \cdot \log (n \cdot w) \geq \Omega(\log n). \]

The proof is similar to the proof of the lower bound for the prefix-sum problem given in Theorem 10.13. Assume we have a data structure for solving the graph connectivity problem. We shall find a random sequence of edge additions, and then ask one random connectivity query, and prove that the data structure must probe many locations. As usual, we can assume that the data structure is deterministic, and prove that the expected number of memory cells accessed is large.

**Proof.** For a parameter set $k \leq O(t_uw)$ that we shall in the proof, we assume without loss of generality that $n = 2(k^{r+1} - 1)/(k - 1)$, for some $r$. We sample a uniformly random graph that consists of two disconnected $k$-ary trees $T_0, T_1$, each of depth $r$. The number of vertices in such a graph is $n$. One can sample such a graph by randomly relabeling the vertices of 2 such trees.

We add the edges of this graph to the data structure in $r$ rounds, labelled $j = 1, 2, \ldots, r$. In the first round, we add all $2k^r$ edges to the leaves of the trees. In the $j$'th round, we add the edges from depth $r - j$ to $r - j + 1$. After all the edges of the trees have been added, we pick two random leaves and query whether or not they are connected in the graph. The leaves will be connected when they belong to the same tree, and disconnected if they belong to distinct trees.

Say that a cell of the data structure belongs to round $j$ if it was last touched in round $j$ of the updates. We prove that for each $j$, the probability that a cell that belongs to round $j$ was accessed to answer
the query is at least \( \Omega(1) \). Thus the expected number of queries
made must be at least
\[
E[t_q] \geq \Omega(r) \geq \Omega \left( \frac{\log n}{\log(t_uw)} \right),
\]
which completes the proof.

It remains to prove that the probability that a cell that belongs to
round \( j \) was queried is \( \Omega(1) \). For the rest of the proof, fix a particular
round \( j \). Let \( A \in \{0, 1\}^{2k^{r-j+1}} \) be the random variable which has a bit
for vertex \( v \) at depth \( r-j+1 \) in the graph, describing which tree \( v \)
belongs to. Formally,
\[
A_v = \begin{cases} 
1 & \text{if } v \in T_1, \\
0 & \text{if } v \in T_0.
\end{cases}
\]

Let \( U, V \) be two uniformly random and independent vertices at
depth \( r-j+1 \) in the graph. Set \( R = A_U + A_V \mod 2 \). Note that \( R \)
encodes whether \( U, V \) are connected in the graph. Let \( D \) encode \( U, V \),
the edges of the graph not added in the \( j \)'th round, and the locations
and contents of all cells that belong to rounds \( j' > j \). The key claim of
the proof is:

**Claim 10.18.** \( H(R|D) \geq 1 - \frac{7t_uw}{k} \).

Before we prove the claim, let us see how to use it. We shall set \( k \)
large enough so that the claim shows \( H(R|D) \) is close to 1. Define
\[
Q = \begin{cases} 
1 & \text{if the data structure queries a cell that belongs to round } j, \\
0 & \text{otherwise.}
\end{cases}
\]

By Claim 10.18, we have:
\[
1 - \frac{7t_uw}{k} \leq H(R|D) \leq H(QR|D)
\]
\[
= H(Q|D) + H(R|Q)
\]
\[
\leq H(Q) + H(R|Q). 
\]

Let \( U, V \) denote the ancestors of the two leaves queried by the algo-
rithm at depth \( r-j+1 \). Suppose \( Q = 1 \) with probability \( \gamma \). When
\( Q = 0 \), the output of the algorithm is determined by \( D \), since all the
cells that are read in order to compute whether \( U, V \) are connected
are determined by \( D \). Define
\[
E = \begin{cases} 
1 & \text{if the data structure makes an error,} \\
0 & \text{otherwise.}
\end{cases}
\]
When $Q = 0$, $R$ is determined by $E$ and $D$. Thus,

$$1 - \frac{7t_u \bar{w}}{k} \leq H(Q) + H(R|DQ)$$

$$\leq h(\gamma) + \gamma \cdot 1 + (1 - \gamma) \cdot H(R|D, Q = 0)$$

$$\leq h(\gamma) + \gamma + H(E|D, Q = 0)$$

$$\leq h(\gamma) + \gamma + h(\epsilon/(1 - \gamma)).$$

If $k$ is set to be a large multiple of $t_u \bar{w}$, the left hand side is close to 1. If $\epsilon, \gamma$ are small, the right hand side is close to 0. Thus we must have $\gamma = \Omega(1)$.

**Proof of Claim 10.18.** Let $B$ be the random variable encoding all the edges not added to the graph in the $j$'th round. After fixing the value of $B$, the roots of the two trees have been fixed, the identities of the leaves have also been determined, but the graph still consists of many disjoint and full $k$-ary trees, as in Figure 10.8.

Let $C$ denote the locations and contents of all cells that belong to rounds larger than $j$. Then we see that

$$H(A|B) = \log \left( \frac{2k^{r-j+1}}{k^{r-j+1}} \right)$$

$$\geq 2k^{r-j+1} - 1 - \log k^{(r-j+1)/2}$$

$$\geq 2k^{r-j+1} - 2k^{r-j},$$

since after fixing $B$, there are $2k^{r-j+1}$ vertices at depth $r - j + 1$, and exactly half of these components will get $A_v = 0$. The number of edges added after the $j$'th round is

$$\sum_{i=j+1}^{r} 2k^{r-i+1} = 2 \cdot \sum_{i=1}^{r-j} k^i = 2k \cdot \frac{k^{r-j} - 1}{k - 1} \leq 4k^{r-j}.$$  

Since $k - 1 \geq k/2$. 

Figure 10.8: The edges of $B$ when $k = 3, r = 5, j = 2$.  

here $h(\gamma)$ is the binary entropy function

$$h(\gamma) = \gamma \cdot \log(1/\gamma) + (1 - \gamma) \cdot \log(1/(1 - \gamma)).$$

by subadditivity

by Markov’s inequality, the probability of error is at most $\epsilon/(1 - \gamma)$ conditioned on $Q = 0$. 


Given $B$, each of these edges can contribute at most $t_u w$ to the entropy of $C$. This is because $C$ can be described by specifying the contents of each of the cells accessed when the algorithm adds these edges. Hence,

$$H(A|BC) \geq H(A|B) - H(C|B) \geq 2k^{r-j+1} - 2k^{r-j} - 1 - 4k^{r-j} t_u w \geq 2k^{r-j+1} - 6k^{r-j} t_u w.$$ 

Let $U, V$ be two uniformly random and independent vertices at depth $r-j+1$ in the graph. For any fixed vertex $w$, we have

$$\Pr[w \in \{U, V\}] = 1 - \left(1 - \frac{1}{2k^{r-j+1}}\right)^2 = \frac{1}{k^{r-j+1}} - \left(\frac{1}{2k^{r-j+1}}\right)^2.$$

Applying Shearer’s lemma (Lemma 6.5), we conclude that

$$H(A_{U}, A_{V}|UVBC) \geq \left(\frac{1}{k^{r-j+1}} - \left(\frac{1}{2k^{r-j+1}}\right)^2\right) \cdot \left(2k^{r-j+1} - 6k^{r-j} t_u w\right) \geq 2 - \frac{1}{2k^{r-j+1}} - \frac{6t_u w}{k} \geq 2 - \frac{7t_u w}{k}.$$

Now, $A_{U}, A_{V}$ are determined by $R, A_{U}$. So, we have

$$H(R|D) \geq H(A_{U}, A_{V}|D) - H(A_{U}|D) \geq 2 - \frac{7t_u w}{k} - 1 \geq 1 - \frac{7t_u w}{k},$$

as required.

Exercise 10.1

Modify the Van Emde Boas tree data structure so that it can maintain the median of $n$ numbers, with time $O(\log \log n)$ for adding, deleting and querying the median.
Polytope are subsets of Euclidean space that can be defined by a finite number of linear inequalities. They are fundamental geometric constructs that have been studied by mathematicians for centuries. Any $n \times d$ matrix $A$ and $n \times 1$ vector $b$ defines a polytope $P$:

$$P = \{ x \in \mathbb{R}^d : Ax \leq b \}.$$ 

In this chapter, we explore some questions about the complexity of representing polytopes—when can a complex polytope be expressed as the shadow of a simple polytope? Loosely speaking, a complex polytope is one that requires many inequalities to describe, and a simple polytope is one that requires very few inequalities. Besides being mathematically interesting, it turns out that this question is relevant to understanding the complexity of algorithms based on linear programming.

Basic properties and features of polytopes

Polytopes have many nice properties that makes them easy to manipulate and understand. For example, a polytope $P$ is always convex—whenever $x$ and $y$ are in $P$, then all the points of the line segment between $x$ and $y$ are also in $P$. Indeed, if $\gamma \in [0,1]$, then

$$A(\gamma x + (1 - \gamma)y) \leq \gamma Ax + (1 - \gamma)Ay \leq \gamma b + (1 - \gamma)b = b.$$ 

Although the definition of the polytope seems to involve only inequalities, sets defined using equalities are also polytopes. For example, the set given by solutions $(x,y,z) \in \mathbb{R}^3$ with

$$x = y + z + 1,$$
$$z \geq 0,$$

Here $a \leq b$ means $a_i \leq b_i$ for all $i$. 

In other texts, polytopes are sometimes assumed to be bounded—it is assumed that there is a finite ball that contains the polytope. Throughout this textbook, polytopes may have infinite volume.
is a polytope, because it can be expressed as:
\[
\begin{align*}
  x - y - z &\leq 1, \\
  -x + y + z &\leq -1, \\
  -z &\leq 0.
\end{align*}
\]

A halfspace \( H \) is a particular type of polytope—one that is defined by a single inequality, \( H = \{ x \in \mathbb{R}^d : h \cdot x \leq c \} \), where \( h \) is a \( 1 \times d \) matrix, and \( c \) is a real number. A polytope \( P \) can therefore be viewed as the intersection of the \( n \) halfspaces given by the \( n \) inequalities that define the polytope. Moreover, every linear inequality that the points of the polytope satisfy can be derived from the inequalities that define the polytope:

**Fact 11.1.** If a polytope \( P = \{ x \in \mathbb{R}^d : Ax \leq b \} \) is contained in a halfspace \( H = \{ x \in \mathbb{R}^d : hx \leq c \} \), then there is a \( 1 \times n \) row vector \( v \geq 0 \) such that \( vA = h \) and \( vb \leq c \).

The vector \( v \) promised by Fact 11.1 shows how to derive the inequality of the halfspace from the inequalities defining the polytope. It gives a proof that the points of the polytope belong to the halfspace—for every \( x \in P \), we have \( hx = vAx \leq vb \leq c \).

The intersection of two polytopes is also a polytope. For example, if \( P = \{ x \in \mathbb{R}^d : Ax \leq b \} \) and \( P' = \{ x \in \mathbb{R}^d : A'x \leq b' \} \), then \( P \cap P' = \{ x \in \mathbb{R}^d : Zx \leq d \} \), with

\[
Z = \begin{bmatrix} A \\ A' \end{bmatrix}, \quad d = \begin{bmatrix} b \\ b' \end{bmatrix}.
\]

The **dimension** of a polytope \( P \) is the dimension of the minimal affine subspace \( A \) such that \( P \subseteq A \). A point \( v \) is on the **boundary** of \( P \) if \( v \in P \), and for every \( \epsilon > 0 \), there is a point \( u \in A - P \) at distance at most \( \epsilon \) from \( v \). A **face** of a polytope \( P \) is a set of the form \( F = P \cap H \), where the dimension of \( F \) is strictly less than the dimension of \( P \), and \( H \) is a halfspace that intersects \( P \) only on its boundary. There may be multiple halfspaces \( H \) that generate the same face in this way. Since the intersection of two polytopes is always a polytope, every face of a polytope is also a polytope.

When the dimension of the face is exactly one less than the dimension of the polytope itself, we call the face a **facet**. Every point on the boundary of the polytope must belong to some facet of the polytope, so the boundary of the polytope is the union of all of its facets. The inequalities defining the polytope may not correspond to the facets of the polytope. In fact, they can be redundant, since some of the inequalities may be implied by the others. Nevertheless, by Fact 11.1, a halfspace defining a facet can be derived by combining the inequalities defining the polytope.
We always need at most as many inequalities to express a polytope as it has facets:

**Fact 11.2.** If a polytope $P \subseteq \mathbb{R}^d$ has $r$ facets, it can be expressed with $r$ inequalities as $P = \{ x : Ax \leq b, Cx = 0 \}$, where $A$ is an $r \times d$ matrix, $b$ is an $r \times 1$ column vector, and $C$ is a $k \times d$ matrix for some $k$.

The facets can be used to generate all the other faces of the polytope:

**Fact 11.3.** Every face of $P$ can be expressed as the intersection of some subset of the facets of $P$. If the polytope is defined by $n$ inequalities, it can have at most $n$ facets.

One important consequence of Fact 11.3 is that a polytope with $f$ facets can have at most $2^f$ faces, since there are at most $2^f$ subsets of the facets.

A vertex of the polytope is a face of dimension 0—it consists of a single point.

**Transformations of polytopes**

Polytopes behave nicely under some natural transformations. Rotating or translating a polytope gives another polytope.
For example, if \( P = \{ x \in \mathbb{R}^d : Ax \leq b \} \) is a polytope, and \( z \in \mathbb{R}^d \) is any vector, then the set
\[
P + z = \{ x + z : Ax \leq b \}
\]
is also polytope. This is because \( y \in P + z \) exactly when \( A(y - z) \leq b \), which is equivalent to \( Ay \leq b + Az \). So,
\[
P + z = \{ y \in \mathbb{R}^d : Ay \leq b + Az \}.\]

We shall see that applying an arbitrary linear transformation to a polytope gives another polytope:

**Theorem 11.4.** If \( L \) is a \( k \times d \) real valued matrix, and \( P \subseteq \mathbb{R}^d \) is a polytope, then
\[
L(P) = \{ Lx : x \in P \} \subseteq \mathbb{R}^k
\]
is also a polytope. Moreover, every face of \( L(P) \) is equal to \( L(F) \), for some face \( F \subseteq P \).

It is often much easier to describe a polytope by applying a linear transformation to another polytope. To illustrate this, let us explore a generic way to generate a polytope from a finite set of points. Given a set of points \( V = \{ v_1, \ldots, v_k \} \in \mathbb{R}^d \), the **convex hull** of these points is the minimum convex set containing \( V \). Equivalently, it the set of points \( x \in \mathbb{R}^d \) satisfying
\[
x_i = \sum_{j=1}^{k} \mu_j \cdot v_i \quad \text{for } i = 1, 2, \ldots, d
\]
\[
\mu_j \geq 0 \quad \text{for } j = 1, 2, \ldots, k
\]
\[
\sum_{j=1}^{k} \mu_j = 1
\]
for some \( \mu_1, \ldots, \mu_k \in \mathbb{R} \). The equations above describe a polytope in \( (x_1, \ldots, x_d, \mu_1, \ldots, \mu_k) \). Projecting this polytope onto the variables \( x_1, \ldots, x_d \) is a linear transformation, so the convex hull is also a polytope.

When the polytope is bounded—when it is contained in some finite sized ball, we have the fact:

**Fact 11.5.** A bounded polytope can always be expressed as the convex hull of all its vertices.

Similarly, the **conical hull** of a finite set \( V \subseteq \mathbb{R}^d \) is the set of points that can be obtained by taking non-negative linear combinations of the points in \( V \). It is the set of points \( x \in \mathbb{R}^d \) satisfying:
\[
x_i = \sum_{j=1}^{k} \mu_j \cdot v_i \quad \text{for } i = 1, 2, \ldots, d
\]
\[
\mu_j \geq 0 \quad \text{for } j = 1, 2, \ldots, k,
\]
Not every polytope is the convex hull of a finite set of points. However, every bounded polytope is the convex hull of its vertices.

Figure 11.1 shows the convex hull of 8 points, and Figure 11.2 shows the convex hull of 6 points.
for some $\mu_1, \ldots, \mu_k \in \mathbb{R}$. Again, Theorem 11.4 implies that the conical hull of a finite set of points is a polytope.

If $k = d$ and $L$ is invertible, then the proof of Theorem 11.4 is straightforward—if $P = \{x : Ax \leq b\}$, then

$$L(P) = \{y : AL^{-1}y \leq b\},$$

so $L(P)$ is a polytope by definition. In this case, the structure of the polytope is also preserved: there is a one to one correspondence between the faces of $P$ and the faces of $L(P)$. Every face of $P$ given by $F = P \cap H$ corresponds to the face $L(F)$ of $L(P)$, and the linear dimension of $L(F)$ is exactly the same as the linear dimension of $F$, so every facet or vertex of $P$ becomes a facet or vertex of $L(P)$ under the linear transformation.

When $L$ is not invertible, the theorem becomes a little more involved to prove. Recall that every $k \times d$ matrix $L$ has a singular value decomposition $L = UV^T\Lambda$, where $U$ is an invertible $k \times k$ matrix with $UU^T = I$, $V$ is an invertible $d \times d$ matrix with $VV^T = I$, and $\Lambda$ is a $k \times d$ diagonal matrix, with $\Lambda_{ij} = 0$ whenever $i \neq j$. So, $Lx$ can be computed by first applying an invertible linear transformation $V$, then projecting and scaling the polytope on to a subset of the coordinates using $\Lambda$, and then applying another invertible linear transformation $U$. Since $\Lambda$ is diagonal, we can express it as $\Lambda = DS$, where $D$ is a $k \times d$ diagonal matrix where all non-zero entries are 1, and $S$ is a $d \times d$ diagonal matrix where all entries on the diagonal are non-zero. Then we have $L = UDSV$, where $U$, $V$ and $S$ are invertible, so they all preserve the polytope structure.

Given a matrix $D$ as above, we say that $D(P)$ is a projection of $P$. The projection of a polytope can actually have a different number of faces and facets than the original polytope. The following Lemma is enough to prove Theorem 11.4.

**Lemma 11.6.** If $D$ is a $k \times d$ diagonal matrix, such that every non-zero entry of $D$ is 1, and $P \subseteq \mathbb{R}^d$ is a polytope, then $D(P)$ is also a polytope. Moreover, every face of $D(P)$ is equal to $D(F)$, for some face $F \subseteq P$.

**Proof.** The lemma follows from repeating a process called Fourier-Motzkin elimination. It is enough to prove that the Lemma holds when $d = k + 1$, since any projection onto fewer coordinates can be obtained by repeatedly projecting on to one smaller dimension.

Suppose $P \subseteq \mathbb{R}^{k+1}$ is a polytope, and every point of $P$ can be written $(x_1, \ldots, x_k, z)$. Suppose $T$ is the projection operation that maps such a point to $(x_1, \ldots, x_k)$. Let $x = (x_1, \ldots, x_k)$. The inequalities

We used the singular value decomposition of matrices when proving the linear lower bound for the gap-hamming problem in Chapter 5.

Quantifying how the facets of a polytope can increase under linear transformations is the main goal of this chapter.
We call these inequalities of type $A$ which $H$.

Then, $(A_i - A_j) \cdot x \leq b_i + b_j$.

If $P$ was defined by $n$ inequalities, we obtain at most $n^2$ inequalities in this way. We claim that these inequalities define $P$.

Clearly every inequality we have derived is satisfied by the elements of $P$. On the other hand, if $x \in \mathbb{R}^d$ satisfies all of these inequalities, we shall prove that there is a choice of $z$ such that $(x,z) \in P$, so $x \in D(P)$. Let $\ell$ be the index maximizing $A_\ell \cdot x - b_\ell$ over all $\ell$ for which $A_\ell, b_\ell$ define an inequality of type $-1$. Set $z = A_\ell \cdot x - b_\ell$. This choice of $z$ ensures that $(x,z)$ satisfies all the inequalities of type $-1$:

$$A_j \cdot x - b_j \leq z \Rightarrow A_j \cdot x - z \leq b_j.$$ 

$(x,z)$ also satisfies all the inequalities of type $0$—these do not involve $z$. Every inequality $A_i \cdot x + z \leq b_i$ of type $1$ is also satisfied, because $x$ satisfies

$$(A_i + A_\ell) \cdot x \leq b_i + b_\ell$$

$$\Rightarrow A_\ell \cdot x - b_\ell + A_i \cdot x \leq b_i$$

$$\Rightarrow z + A_i \cdot x \leq b_i.$$

Suppose $H \subseteq \mathbb{R}^k$ is halfspace such that $H \cap D(P)$ is a face of $D(P)$. Then, $H$ can be expressed as the set of points satisfying $h \cdot x \leq c$ for some $h,c$. Let $H' \subseteq \mathbb{R}^{k+1}$ be the set of points $(x,z)$ such that $h \cdot x \leq c$. Then $H'$ is a halfspace, and $D(H' \cap P) = H \cap P$. Indeed, if $(x,z) \in H' \cap P$, then $x \in H \cap D(P)$. If $x \in D(P)$, then there must be a $z$ such that $(x,z) \in P$, and we see that $(x,z) \in H'$ as well. Moreover, $H' \cap P$ is a face, because for every $\epsilon$, if $(x,z) \in H' \cap P$, then there is a point $x'$ such that $||x' - x|| \leq \epsilon/2$ and $x'$ is in the smallest affine subspace containing $D(P)$, but not in $D(P)$. Then $(x',z)$ is within $\epsilon$ of $(x,z)$, is in the smallest affine space containing $P$, but not in $P$ itself.

\[\square\]
Algorithms from Polytopes

Besides being fundamental geometric objects, polytopes are useful from the perspective of algorithm design, because many interesting computational problems can be reduced to the problem of optimizing a linear function over some polytope. We illustrate this with some examples of how to reduce combinatorial problems to optimizing linear functions over polytopes.

Shortest Paths  Say we want to compute the distance between the two vertices 1 and n in an undirected graph with vertex set \([n]\). We show how to encode this algorithmic problem as a question about optimizing a linear function over a polytope. For every pair of distinct vertices \(u, v \in [n]\), define the variable \(x_{u,v}\), and define the graph polytope \(P\) by

\[
\begin{align*}
  x_{u,v} &\geq 0 & \text{for every } u \neq v \\
  \sum_w x_{1,w} &= 1 \\
  \sum_w x_{w,u} &= 1 \\
  \sum_{w \neq u} x_{u,w} &= \sum_{w \neq u} x_{w,u} & \text{for every } u \notin \{1, n\}
\end{align*}
\]

These equations define a polytope \(P \subseteq \mathbb{R}^d\), with \(d = n(n-1)\), which has at most \(d\) facets, since it is defined by \(d\) inequalities.

Now, given a connected graph \(G\) with the edge set \(E\), consider the problem of finding the point in the graph polytope that minimizes the linear function

\[
L(x) = \sum_{\{u,v\} \notin E} n \cdot (x_{u,v} + x_{v,u}) + \sum_{\{u,v\} \in E} (x_{u,v} + x_{v,u}).
\]

**Claim 11.7.** \(\min_{x \in P} L(x)\) is the distance from 1 to n in \(G\).

**Proof.** Without loss of generality, suppose \(e_1, e_2, \ldots, e_\ell\) are the edges of a shortest path in the graph. If we give the edges of this path weight 1 and all other edges weight 0, that gives a point \(x\) in the polytope with \(L(x) = \ell\).

To prove the claim, we show that for any other \(x \in P\), \(L(x) \geq \ell\). Suppose \(x \in P\) is such that \(x_v > 0\) for some edge \(e = (u, v)\) that is not on the shortest path. Since the flow into intermediate vertices must be equal to the flow out of intermediate vertices, there must be an edge \((v, w)\) that has positive flow, and then an edge \((w, z)\) with positive flow and so on. In this way, we see that
either there is a path $e'_1, \ldots, e'_k$ from 1 to $n$ with $k \geq \ell$, where all edges get positive flow, or there is a directed cycle $e'_1, \ldots, e'_k$ with positive flow. In the first case, we can reduce the flow on $e'_1, \ldots, e'_k$ a small amount, and increase the weight of the shortest path $e_1, \ldots, e_\ell$ by the same amount. This gives a new point in the polytope, and reduces the weight placed on edges that are not on the shortest path so the value of $L(x)$ can only decrease. In the case that $e'_1, \ldots, e'_k$ form a directed cycle, we can reduce the weight of all the edges by a small amount to get a new point in the polytope. Again, this reduces the weight on the edges outside the shortest path. See Figure 11.4.

Repeating these operations eventually gives a solution $x \in P$ that only places weight on the edges $e_1, \ldots, e_\ell$. Such a solution must have $L(x) = \ell$.

\[ \square \]

**Matchings** Suppose we wish to find the size of the largest matching in a given graph. We show that this problem can also be encoded as the problem of optimizing a linear function over some polytope. For every pair $u, v \in [n]$ of distinct vertices we have the variable $x_{\{u,v\}}$. Each matching corresponds to the points where $x_{\{u,v\}} = 1$ if $u, v$ are matched and $x_{\{u,v\}} = 0$ if they are not. The convex hull $M$ of these matching is called the matching polytope. One can prove that the facets of the polytope correspond to the inequalities\(^2\):

\[ \text{Figure 11.4: Given a point } x \text{ in the polytope that puts weight on a directed cycle or a path that is not the chosen shortest path of the graph, one can always modify the point to obtain a point that only puts weight on the shortest path, and yet does not have a larger value of } L(x). \]
We shall see in Exercise 11.5 that one can come up with \( O(n^2) \) equations to work with the bipartite matching polytope.

Given that the complexity of solving optimization problems on polytopes is related to the number of facets of the polytope, it is important to find polytopes that have a small number of facets that encode computational problems.
A generic way to do this is via extensions of the polytope. A polytope \( Q \subseteq \mathbb{R}^k \) is an extension of a polytope \( P \subseteq \mathbb{R}^d \) if there is a linear map \( L : \mathbb{R}^k \to \mathbb{R}^d \) such that \( L(Q) = P \). The extension complexity of \( P \) is the minimum number of facets achieved by any extension of \( P \). In fact, there are polytopes that admit non-trivial extensions, namely extensions with fewer facets. See Figure 11.5 for an example.

The following lower bound always holds:

**Claim 11.9.** If a polytope \( P \) has \( n \) faces, its extension complexity is at least \( \log n \).

**Proof.** Suppose \( P \) is the projection of a polytope \( Q \). Every face of \( P \) is to the projection of a face of \( Q \). Every face of \( Q \) is the intersection of a subset of the facets of \( Q \). So if \( Q \) has \( k \) facets, \( P \) can have at most \( 2^k \) faces. \( \square \)

Next, we explore some examples of natural polytopes that have small extension complexity.

**Regular Polygons** A polygon is a polytope in \( \mathbb{R}^2 \). Consider the polygons \( P \subseteq \mathbb{R}^2 \) with \( n \) facets, where \( n = 2^k \) is a power of 2. One can show\(^3\) that there are such \( n \)-gons whose extension complexity is at least \( \Omega(\sqrt{n}) \). Here we show that if the polygon has sufficient symmetries, then it has low extension complexity\(^4\)—when the polygon is regular, we prove that its extension complexity is \( O(k) \).

The key idea is that one can reflect a polygon without increasing its extension complexity by much. Consider any polygon \( P \subseteq \mathbb{R}^2 \) which is defined by \( P = \{ x : Ax \leq b \} \). By applying a rotation and a translation, we can assume without loss of generality that \( x_1 \leq 0 \) is an inequality defining a facet of \( P \). Define a new polytope \( Q \subseteq \mathbb{R}^3 \) by replacing each inequality

\[
A_{i,1} \cdot x_1 + A_{i,2} \cdot x_2 \leq b_i
\]

of \( P \) with the inequality

\[
A_{i,1} \cdot x_3 + A_{i,2} \cdot x_2 \leq b_i,
\]

for \( Q \), and add in the inequalities \( -x_3 \leq x_1 \leq x_3 \) to \( Q \).

Let \( \pi \) be the projection map defined by

\[
\pi(x_1, x_2, x_3) = (x_1, x_2),
\]

and set \( P' = \pi(Q) \). The polytope \( P' \) is the union of \( P \) and its reflection with respect to \( x_1 = 0 \). The number of inequalities defining \( Q \) is only 2 more than the number of inequalities defining \( P \), but the number of facets of \( P' \) may be a factor of 2 larger than the number of facets of \( P \)!
Given this description of $P'$, we can repeat the process again. We first rotate and translate $P'$ so that $x_1 \leq 0$ corresponds to a facet of $P'$. Then, if $P' = \{ x : A'x \leq b' \}$, we define a new polytope $Q'$ by replacing each inequality

$$A'_{i,1} \cdot x_1 + A'_{i,2} \cdot x_2 + A'_{i,3} \cdot x_3 \leq b'_i$$

of $P'$ by the inequality

$$A'_{i,1} \cdot x_4 + A'_{i,2} \cdot x_2 + A'_{i,3} \cdot x_3 \leq b'_i$$

of $Q'$, and add in the inequalities $-x_4 \leq x_1 \leq x_4$. If $\pi'(x_1, x_2, x_3, x_4) = (x_1, x_2)$ is the projection on to the first two coordinates, then the polytope $P'' = \pi'(Q')$ is the union of the reflection of $P'$ about the facet defined by $x_1 \leq 0$ with itself.

Using this method, we can construct a regular polygon with $n = 2^k$ facets using $k = \log n$ reflections, starting from a triangle, which only requires 3 inequalities to define. So, every such polygon has extension complexity at most $O(\log n)$. By Fact 11.9, the extension complexity of such a polygon cannot be less than $\log n$, so up to constant factors, this is the best we can hope for.

**Permutahedron** For every permutation $\sigma : [n] \to [n]$, define the point $p^\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n)) \in \mathbb{R}^n$. The permutahedron is the convex hull of these $n!$ points.

The dimension of this polytope is $n - 1$. To see this, first observe that the permutahedron lies in the hyperplane $\sum_{i=1}^n x_i = \binom{n}{2}$, so its dimension is at most $n - 1$. To see that the dimension is at least $n - 1$, let $p^\mathbb{1} = (1, 2, 3, \ldots, n)$ be the point corresponding to the identity permutation, and let $\sigma_2, \ldots, \sigma_n$ be the $n - 1$ permutations obtained by swapping 1 with 2, \ldots, $n$:

$$\sigma_i(k) = \begin{cases} 1 & \text{if } k = i, \\ i & \text{if } k = 1, \\ \sigma(k) & \text{otherwise.} \end{cases}$$

The $n - 1$ points of the form $p^\mathbb{1} - p^{\sigma_i}$ are linearly independent, since $p^\mathbb{1} - p^{\sigma_i}$ is the only such vector with a non-zero entry in the $i$'th coordinate.

The permutahedron is the convex hull of the points given by the permutations, so we can express it using $n!$ inequalities. Thus, it has at most $n!$ facets. Next, we compute each of the facets explicitly.

**Lemma 11.10.** The permutahedron is the set of points satisfying the
conditions:
\[
\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} i = \binom{n}{2}, \\
\sum_{i \in S} x_i \geq \sum_{i=1}^{\lfloor |S| \rfloor} i 
\]
for all sets \( S \subseteq [n] \) with \( 0 < |S| < n \).

The facets of the permutahedron correspond to these \( 2^n - 2 \) inequalities.

Proof. Let \( Q \) denote the polytope defined by the constraints. Every permutation satisfies the constraints, so the permutahedron is contained in \( Q \). Let us prove that every point of \( Q \) is in the permutahedron.

Suppose \( x \in Q \). We shall show that \( x \) is in the convex hull of the points corresponding to the permutations. The key claim is:

Claim 11.11. If \( x \) satisfies the constraints

\[
\sum_{i \in S} x_i \geq \sum_{i=1}^{\lfloor |S| \rfloor} i 
\]
for all sets \( S \subseteq [n] \) with \( 0 < |S| < n \),

then there is a permutation \( \sigma \), and a positive number \( \epsilon > 0 \), such that the point \( x - \epsilon \cdot p^\sigma \) also satisfies all of the constraints.

Repeatedly applying this claim, we see that \( x \) must lie in the conical hull of the points \( p^\sigma \) given by the permutation. Indeed, in each step, we reduce \( \sum_{i=1}^{n} x_i \), until \( x \) is expressed as a conical
combination of the permutations. But this implies that if \( x \in Q \), then \( x \) must lie in the convex hull of the permutations, since if

\[
x = \sum_{\sigma} \mu_\sigma \cdot p^\sigma,
\]

then

\[
\binom{n}{2} = \sum_{i=1}^{n} x_i = \sum_{\sigma} \mu_\sigma \sum_{i=1}^{n} p^\sigma_i = \binom{n}{2} \cdot \sum_{\sigma} \mu_\sigma,
\]

so \( \sum_{\sigma} \mu_\sigma = 1 \). To prove the claim, suppose \( x \) satisfies \( \sum_{i \in S} x_i = \sum_{i=1}^{\left|S\right|} i \) and \( \sum_{i \in T} x_i = \sum_{i=1}^{\left|T\right|} i \), for two distinct sets \( S, T \). We claim that we must have \( S \subseteq T \) or \( T \subseteq S \). Otherwise we would have

\[
\sum_{i \in S \cup T} x_i = \sum_{i \in S} x_i + \sum_{i \in T} x_i - \sum_{i \in T \cap S} x_i \\
\leq \sum_{i=1}^{\left|S\right|} i + \sum_{i=1}^{\left|T\right|} i - \sum_{i=1}^{\left|T \cap S\right|} i \\
= \sum_{i=1}^{\left|S\right|} i + \sum_{i=\left|T \cap S\right|+1}^{\left|T\right|} i < \sum_{i=1}^{\left|S \cup T\right|} i,
\]

using the inequality for \( T \cap S \).

contradicting the constraint for \( S \cup T \). Thus, the sets that give constraints that \( x \) satisfies with equality can be arranged into a chain: \( S_1 \subseteq S_2 \subseteq \ldots \subseteq S_k \). Let \( \sigma \) be a permutation with \( \sigma(S_i) = S_i \) for \( i = 1, 2, \ldots, k \). This permutation also satisfies the same equations with equality. For a small enough \( \epsilon > 0 \), we must have that \( x - \epsilon p^\sigma \in Q \), as required.

To see that each of the inequalities gives a facet, fix a set \( S \subset [n] \) with \( 0 < \left|S\right| < n \), and consider the points of the permutahedron satisfying the constraint corresponding to \( S \) with equality. These points form a face. For simplicity, suppose \( S = \{1, 2, \ldots, t\} \). Let \( p^\alpha = (1, 2, \ldots, n) \) denote the point corresponding to the identity permutation, and for \( i = 2, 3, \ldots, t, t+2, t+3, \ldots, n \), define \( \sigma_i \) to be the permutation that swaps the \( i \)'th element with 1 if \( i \leq t \), or the \( i \)'th element with \( t+1 \) if \( i > t \):

\[
\sigma_i(k) = \begin{cases} 
 1 & \text{if } i \leq t, k = i, \\
  i & \text{if } i \leq t, k = 1 \\
  t+1 & \text{if } i > t, k = i, \\
  i & \text{if } i > t, k = t+1, \\
  k & \text{otherwise.}
\end{cases}
\]

Then we see that the \( n-2 \) vectors \( p^{\sigma_i} - p^\alpha \) are linearly independent, and all of these points belongs to the face. Thus, the face has dimension \( n-2 \), and so must be a facet. Similarly, one can argue that the inequality corresponding to every set \( S \) defines a facet of the polytope.
It is known⁵ that the extension complexity of the permutahedron is $O(n \log n)$. This bound is tight: the polytope has $n! = 2^{\Omega(n \log n)}$ vertices, so its extension complexity must be at least $\Omega(n \log n)$ by Fact 11.9.

Here we prove⁶ that its extension complexity is at most $n^2$.

Define the polytope $Q$ using the inequalities:

$$Y_{i,j} \geq 0 \quad \text{for } i, j \in [n]$$

$$\sum_{i=1}^{n} Y_{i,j} = 1 \quad \text{for all } j \in [n]$$

$$\sum_{j=1}^{n} Y_{i,j} = 1 \quad \text{for all } i \in [n]$$

$Q$ is defined by $n^2$ inequalities, so it has at most $n^2$ facets. We shall prove that the permutahedron is a projection of $Q$, and so its extension complexity is at most $n^2$. Let $v$ be the column vector $(1, 2, \ldots, n)^T$.

**Claim 11.12.** $Yv$ is an element of the permutahedron if and only if $Y \in Q$.

**Proof.** Each permutation $\sigma$ corresponds to a boolean permutation matrix $Y^\sigma \in Q$ where $Y^\sigma_{i,j} = 1$ if and only if $\sigma(i) = j$. Since $p^\sigma = Y^\sigma v$, every element of the permutahedron can be realized as $Yv$ for some $Y \in Q$. This proves that every point of the permutahedron can be expressed as $Yv$ for some $Y \in Q$, since a convex combination of the permutations can be obtained by taking the appropriate convex combination of the permutation matrices.

For any $Y \in Q$,

$$\sum_{i=1}^{n} (Yv)_i = \sum_{i=1}^{n} \sum_{j=1}^{n} Y_{i,j} \cdot j = \sum_{j=1}^{n} j.$$ 

So, every point obtained as $Yv$ satisfies the first constraint of the permutahedron. It only remains to check that $Yv$ satisfies the inequalities of the permutahedron. For any set $S \subseteq [n]$ with $0 < |S| = k < n$, we want to prove

$$\sum_{i \in S} (Yv)_i \geq \sum_{i=1}^{k} i.$$ 

Write

$$\sum_{i \in S} (Yv)_i = \sum_{j=1}^{n} \alpha_j \cdot j,$$

with

$$\alpha_j = \sum_{i \in S} Y_{i,j}.$$
For all \( j \), we have

\[
0 \leq \alpha_j = \sum_{i \in S} Y_{i,j} \leq \sum_{i=1}^n Y_{i,j} \leq 1,
\]

and we have

\[
\sum_{j=1}^n \alpha_j = \sum_{i \in S} \sum_{j=1}^n Y_{i,j} = k.
\]

Under these constraints, the vector \((\alpha_1, \ldots, \alpha_n)\) that minimizes \(\sum_{j=1}^n \alpha_j \cdot j\) has \(\alpha_1 = \ldots = \alpha_k = 1\) and \(\alpha_{k+1} = \ldots = \alpha_n = 0\), as required.

**Polytopes from Boolean Circuits**  
The connection between polytopes and algorithms also goes the other way—efficient algorithms lead to efficient ways to represent polytopes. To explain this connection, we need the concept of a *separating* polytope. Given a boolean function \( f : \{0, 1\}^n \to \{0, 1\} \), we say that the polytope \( P \subseteq \mathbb{R}^n \) is separating for \( f \) if \( f(x) = 1 \) if and only if \( x \in P \).

**Claim 11.13.** If \( f \) can be computed by a circuit with \( s \) gates, there is a polytope separating \( f \) that has extension complexity at most \( O(s) \).

**Proof.** Consider the polytope \( P \) obtained by defining a variable \( v_g \) for every gate \( g \), and let \( v_1, \ldots, v_n \) be the variables corresponding to the \( n \) inputs to \( f \). For each variable, we have the constraints:

\[
0 \leq v_g \leq 1.
\]

If \( g = \neg h \), we add the constraint

\[
v_g = 1 - v_h.
\]

If \( g = h \lor r \), we add the constraints

\[
v_g \geq v_h,
\]
\[
v_g \geq v_r,
\]
\[
v_g \leq v_h + v_r.
\]

If \( g = h \land r \), we add the constraints

\[
v_g \leq v_h,
\]
\[
v_g \leq v_r,
\]
\[
v_g \geq v_h + v_r - 1.
\]

Finally, we add the constraint \( v_f = 1 \), where \( f \) denotes the gate computing the output of the circuit.

When this polytope is projected onto the inputs, we obtain a polytope \( P \) that we claim separates \( f \). Indeed, we see that if
214 Communication Complexity

If \( f(x) = 1 \), then \( x \in P \), since we can assign all of the gate variables the values computed in the circuit, and these values satisfy the inequalities we defined. On the other hand, if \( f(x) = 0 \), then we claim that the constraints cannot be satisfied by any assignment. This can be seen by assigning the value 0 to gate \( g \) if \( v_g = 0 \) and 1 if \( v_g \neq 0 \). Since \( f(x) = 0 \), some gate in the circuit must get a value that is not consistent with its inputs.

Let \( g \) be such a gate. If \( g = h \lor r \), and \( v_g = 0 \), then the inequalities for \( g \) are violated, since \( v_h > 0 \) or \( v_r > 0 \). If \( v_g > 0 \), then \( v_h = v_r = 0 \), and this violates \( v_g \leq v_h + v_r \). Similarly if \( g = h \land r \) and \( v_g = 0 \), then we must have \( v_h, v_r > 0 \) and this violates \( v_g \geq v_h + v_r - 1 \). If \( v_g > 0 \), then we must have either \( v_h = 0 \) or \( v_r = 0 \), which again causes a violation of the inequalities for \( g \).

\[ \square \]

Slack Matrices

A useful tool for understanding the extension complexity of a polytope is the concept of a slack matrix.

**Definition 11.14.** Suppose \( P \subseteq \{ x : Ax \leq b \} \) is a polytope, where \( A \) is an \( n \times d \) matrix and \( b \) is an \( n \times 1 \) column vector, and \( V = \{ v_1, \ldots, v_k \} \subseteq P \) is a finite set of points. The slack matrix of the polytope with respect to \( A, b, V \) is the \( n \times k \) matrix \( S \) with \( S_{i,j} = b_i - A_i \cdot v_j \), where \( A_i \) is the \( i \)th row of \( A \).

The slack matrix is a non-negative matrix—all of its entries are non-negative. If all the rows of \( A \) have length 1, then \( S_{iv} \) is the distance of \( v \) from the hyperplane defined by \( A_i \). Note that the same polytope can have many different slack matrices, because the facets of the polytope can be defined using many different halfspaces. The extension complexity any polytope \( P \) is determined by its slack matrix\(^7\): it is equal to maximum non-negative rank achievable by a slack matrix of \( P \), as we prove in the following theorem. The theorem gives a powerful way to prove both upper and lower bound on the extension complexity of polytopes. The lower bounds are usually proved using ideas inspired by communication complexity.

**Theorem 11.15.** If \( P \) has extension complexity \( r \), then every slack matrix of \( P \) has non-negative rank at most \( r + 1 \). Conversely, suppose \( P \) is bounded, and \( P = \{ x : Ax \leq b \} \). Suppose the slack matrix corresponding to these inequalities and the vertices of \( P \) has non-negative rank \( r \), then \( P \) has extension complexity at most \( r \).

**Proof.** First, suppose the extension complexity of the polytope \( P \subseteq \mathbb{R}^d \) is \( r \). Then there is a polytope \( Q \subseteq \mathbb{R}^f \) with \( r \) facets, and a linear

\[^7\] Yannakakis, 1991
transformation $L$, represented as a $d \times \ell$ matrix, such that $P = L(Q)$. Without loss of generality, possibly by modifying the linear transformation, we can assume that $Q$ has full dimension, so $Q$ can be expressed as

$$Q = \{x : Cx \leq e\},$$

where $C$ is an $r \times \ell$ matrix, and $e$ is a $r \times 1$ column vector. Then we have $P = \{Lx : Cx \leq e\}$.

Let $S$ be any $n \times k$ slack matrix of $P$, that corresponds to the representation $P \subseteq \{x : Ax \leq b\}$ and the set of points $V = \{v_1, \ldots, v_k\}$. For $i \in [n], v \in V$, we have $S_{i,j} = b_i - A_i v_j$. Let $w_1, \ldots, w_k \in Q$ be such that $Lw_j = v_j$, for $j = 1, 2, \ldots, k$. For each $i$, since $A_i, b_i$ give a valid inequality for $P$, they must also give a valid inequality $A_i Lx \leq b_i$ for the points of $Q$. By Fact 11.1, this inequality can be proved by taking linear combinations of the inequalities defining $Q$. Namely, there must be a non-negative $1 \times r$ row vector $u_i$ such that

$$u_i C = A_i L,$$

and

$$u_i e \leq b_i \Rightarrow b_i = u_i e + \alpha_i,$$

with $\alpha_i \geq 0$. Thus, we have

$$S_{i,j} = b_i - A_i v_j = u_i e + \alpha_i - u_i C w_j = u_i (e - C w_j) + \alpha_i.$$

Let $U$ be the $n \times r + 1$ non-negative matrix whose $i$'th row is given by

$$\begin{bmatrix} u_i & \alpha_i \end{bmatrix},$$

and set $W$ to be the $r + 1 \times k$ non-negative matrix whose $j$'th column is given by

$$\begin{bmatrix} e - C w_j \\ 1 \end{bmatrix},$$

then we see that $S = UW$, proving that the non-negative rank of $S$ is at most $r + 1$.

Conversely, suppose $P = \{x : Ax \leq b\}$, and let $V = \{v_1, \ldots, v_k\}$ be the vertices of $P$. Suppose the slack matrix can be expressed as $S = UW$, where $U$ has $r$ columns and $W$ has $r$ rows, both with non-negative entries. We claim that $P$ is the projection of the polytope $Q = \{(x,y) : Ax + U y = b, y \geq 0\}$, which has at most $r$ facets.

Every vertex $v \in P$ corresponds to a column $y$ of $W$, and the column of $S$ that corresponds to $v$ is $U y = b - A v$. So, $A v + U y = b$, which means that every vertex of $P$ is in the projection of $Q$, since $y \geq 0$. On the other hand, the condition $y \geq 0$ implies that the projection of $Q$ is contained in $P$, since $b = Ax + U y \geq Ax$ for all $(x,y) \in Q$. □
**Spanning Tree Polytope**

The spanning tree polytope is the convex hull of all trees of size $n$: There is a variable $x_e$ for every potential edge $e \subseteq [n]$ of size 2. Every tree gives a point $x$ by setting $x_e = 1$ if $e$ is an edge of the tree, and 0 otherwise. The spanning tree polytope is the convex hull of all of these points. Its vertices correspond to the spanning trees.

The facets of the spanning tree polytope correspond to the inequalities:

$$\sum_{e} x_e = n - 1$$

$$\sum_{e \subseteq S} x_e \leq |S| - 1 \quad \text{for every } S \subseteq [n]$$

Each facet of the polytope corresponds to a subset $S \subseteq [n]$, and each vertex corresponds to a spanning tree $T$. The slack of the pair $S, T$ is exactly $|S| - 1 - k$, where $k$ is the number of edges of $T$ that are contained in $S$. In the case that $T$ is rooted at a vertex $a \in S$, the slack is the number of children in $S$ whose parents are not in $S$.

Motivated by this observation, we show how to encode the slack matrix using a small non-negative factorization. For every tree $T$, define the vector $v$ in $\mathbb{R}^{(\binom{n}{3})}$ as follows. For distinct $a, b, c \in [n]$, set

$$v_{a, b, c} = \begin{cases} 
1 & b \text{ is the parent of } c \text{ when } T \text{ is rooted at } a \\
0 & \text{otherwise.}
\end{cases}$$

For every set $S$, define the vector

$$u_{a, b, c} = \begin{cases} 
1 & a = \min S, b \notin S \text{ and } c \in S \\
0 & \text{otherwise.}
\end{cases}$$

Thus $\sum_{a, b, c} u_{a, b, c} v_{a, b, c}$ is exactly the slack of $T$ from the facet of the set $S$. So, setting $U$ to be the matrix whose rows correspond to the vectors $u$ for each set $S$, and $V$ to be the matrix whose columns correspond to the vectors $v$ for each tree $T$, we can express the slack matrix as $UV$.

This proves that the non-negative rank of the slack matrix is at most $\binom{n}{3}$. By Theorem 11.15, the extension complexity of the spanning tree polytope is at most $\binom{n}{3}$.

**Lower bounds on Extension Complexity**

Our main tool for proving lower bounds on the extension complexity of polytopes are the entropy based techniques discussed in Chapter 6.
Separating Polytopes

Consider any separating polytope $P$ for the function $f : \{0,1\}^n \to \{0,1\}$. Let

$$\Delta(x,y) = \sum_{i=1}^{n} y_i(1-x_i) + (1-y_i)x_i$$

be the Hamming distance between $x, y$. Consider the $|f^{-1}(0)| \times |f^{-1}(1)|$ matrix $M^e$ whose entries are indexed by inputs $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$, and whose $(x,y)$'th entry is $\Delta(x,y) - \epsilon$.

We claim\footnote{Hrubeš, 2016}:

**Theorem 11.16.** Suppose for all $\epsilon > 0$, we have $\text{rank}_+ (M^e) \geq k$. Then the extension complexity of every separating polytope for $f$ is at least $k - 1$.

**Proof.** Let $P$ be a separating polytope for $f$. Thus, $f^{-1}(0) \subseteq P$. For each $y \in f^{-1}(1)$, consider the linear inequality

$$\Delta(x,y) \geq \epsilon.$$ 

Since $P$ is a closed set and $y \notin P$, for each $y$, there is a value $\epsilon > 0$ such that this inequality is satisfied by all the points in $P$. Thus, by taking the smallest $\epsilon$ valid for all $y$'s, we get that $\Delta(x,y) \geq \epsilon$ for every $y \in f^{-1}(1)$ and $x \in P$. Then the inequalities given by the points $y \in f^{-1}(1)$ and the points $x \in f^{-1}(0)$ together give the slack matrix $M^e$. The proof is then complete by Theorem 11.15.

Correlation Polytope

The correlation polytope $C_n$ is the convex hull of cliques. More precisely, there are $n + \binom{n}{2} = \binom{n+1}{2}$ variables of the form $x_T$ for $T \subseteq [n]$ of size 1 or 2. For every $A \subseteq [n]$, define the point

$$x_T^A = \begin{cases} 1 & \text{if } T \subseteq A, \\ 0 & \text{otherwise.} \end{cases}$$

The correlation polytope is the convex hull of all these $2^n$ points. Each point of the correlation polytope can be thought of as a lower triangular matrix.

**Claim 11.17.** $C_n$ has full dimension $\binom{n+1}{2}$.

**Proof.** For every set of size 1, $\{i\}$, $C$ contains the unit vector $x^{\{i\}}$ in the corresponding direction. For every set $\{i,j\}$ of size 2, $C$ contains the unit vector $x^{\{i,j\}} - x^{\{i\}} - x^{\{j\}}$ in that direction. So, $C$ contains $\binom{n+1}{2}$ linearly independent vectors.

We can use the lower bound we proved on non-negative rank, which was proved using ideas from communication complexity, to prove:

A vertex of the correlation polytope:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The facets of $C$ are hard to determine, and no explicit expression is known to capture them.
Theorem 11.18. The extension complexity of $C_n$ is $2^{\Omega(n)}$.

Consider the inequalities:

$$\sum_{i \in B} x(i) \leq 1 + \sum_{T \in \binom{B}{2}} x_T$$

for all non-empty $B \subseteq [n]$.

For each vertex $x^A$ of $C_n$, the left hand side is exactly $|A \cap B|$, and the right hand side is exactly $1 + \binom{|A \cap B|}{2}$, so the inequality always holds. Since the inequality is valid for all the vertices, it is valid for all the points of $C_n$.

Consider the slack matrix $S$ for $C_n$ that correspond to these inequalities, and the vertices of the polytope. We see that $S_{A,B} = 0$ when $A$ and $B$ intersect in one element, and $S_{A,B}$ is 1 when $A$ and $B$ are disjoint. For a parameter $0 \leq \delta \leq 1$, suppose we are given a $2^n \times 2^n$ non-negative matrix $S$ whose rows and columns are indexed by sets $x, y \subseteq [n]$, such that

$$S_{x,y} = \begin{cases} 1 & \text{if } x, y \text{ are disjoint,} \\ \leq 1 - \delta & \text{if } x, y \text{ are not disjoint.} \end{cases}$$

When $\delta = 0$, the matrix may have non-negative rank 1. Here we prove that the rank of the matrix is at least exponential in $\delta^4 n$. In fact, we shall prove the following stronger theorem, which implies Theorem 11.18.

Theorem 11.19. If $S_{x,y} = 1$ when $x, y$ are disjoint and $S_{x,y} \leq 1 - \delta$ when $|x \cap y| = 1$, then $\text{rank}(S) \geq 2^{\Omega(\delta^4 n)}$.

The proof of the theorem is an adaptation of the lower bound we proved for the randomized communication complexity of disjointness.

Proof. Consider the distribution on $x, y$ given by

$$q(xy) = \frac{S_{x,y}}{\sum_{a,b} S_{a,b}}.$$

If $S$ has non-negative rank $r$, then $S$ can be expressed as $S = \sum_{m=1}^{r} S(m)$, where $S(m)$ is a non-negative rank 1 matrix. In other words, $q(xy)$ can be expressed as a convex combination of $r$ product distributions, by setting

$$q(xy|m) = \frac{S(m)_{x,y}}{\sum_{a,b} S(m)_{a,b}},$$

and

$$q(m) = \frac{\sum_{a,b} S(m)_{a,b}}{\sum_{a,b} S_{a,b}}.$$
Let $D$ denote the event that the sets $X, Y$ sampled in this distribution are disjoint. The key step is to prove that for every $i \in [n]$,

$$I(X_i : M|X_{<i}, Y_{\geq i}, D) + I(Y_i : M|X_{<i}, Y_{>i}, D) \geq \Omega(\delta^4) \quad (11.1)$$

Before proving (11.1), we show how to use it. Since $S_{x,y} = 1$ for disjoint sets $x, y$, we know that $q(xy|D)$ is the uniform distribution on all pairs of disjoint sets. In particular, conditioned on $D$, the coordinates $(X_1, Y_1), \ldots, (X_n, Y_n)$ are independent. By Lemma 6.14, we get that

$$2 \log r \geq \sum_{i=1}^n I(X_i : M|X_{<i}, Y_{\geq i}, D) + I(Y_i : M|X_{<i}, Y_{>i}, D)$$

proving that $r \geq 2^{\Omega(\delta^4 n)}$ as required.

Let $Z = (M, X_{<i}, Y_{\geq i})$. Let $U$ denote the event that $X \cap Y \subseteq \{i\}$. Let $p(xym) = q(xym|U)$. Note that for every $z$, $p(xy|z)$ is a product distribution. For fixed $z$, let $\alpha_z$ denote the statistical distance of $p(x_i, y_i|z)$ from uniform. Set

$$I(X_i : M|X_{<i}, Y_{\geq i}, D) + I(Y_i : M|X_{<i}, Y_{>i}, D) = 2\gamma^4/3.$$

Let $G$ denote the set of $z$ for which $\alpha_z \leq 2\gamma$. Let $Q$ denote the event $i \notin X, i \notin Y$, and let $I$ denote the event that $i \in X, i \in Y$. We shall use Pinsker’s inequality to prove:

**Claim 11.20.** $p(z \in G|Q) \geq 1 - 4\gamma$.

Whenever $z \in G$,

$$\frac{p(I, z)}{p(Q, z)} \geq \frac{1/4 - 2\gamma}{1/4 + 2\gamma} = \frac{1 - 8\gamma}{1 + 8\gamma}.$$

So, we have

$$p(I) \geq \sum_{z \in G} p(I, z) \geq \frac{1 - 8\gamma}{1 + 8\gamma} \cdot \sum_{z \in G} p(Q, z) = \frac{1 - 8\gamma}{1 + 8\gamma} \cdot p(z \in G, Q)$$

$$\geq \frac{1 - 8\gamma}{1 + 8\gamma} \cdot (1 - 4\gamma) \cdot p(Q) \quad \text{by Claim 11.20}$$

$$= (1 - O(\gamma)) \cdot p(Q).$$

On the other hand, the assumption on $S$ implies that $p(I) \leq (1 - \delta) \cdot p(Q)$, so we must have $\gamma \geq \Omega(\delta)$.

To prove Claim 11.20, let $\beta_z$ denote the distance of $p(x_i|z)$ from uniform. We have

$$\frac{2}{3} \cdot I(X_i : M|X_{<i}, Y_{\geq i}, Y_i = 0, D) \leq I(X_i : M|X_{<i}, Y_{\geq i}, D) \leq \frac{2}{3} \cdot \gamma^4.$$
Since \( p(z|y_i = 0) \) is identical to \( q(z|y_i = 0, D) \), we can use convexity and Pinsker’s inequality (Lemma 6.10) to conclude
\[
\mathbb{E}_{p(z|y_i = 0)}[\beta_z] \leq \sqrt{\mathbb{E}_{p(z|y_i = 0)}[\beta_z^2]} \leq \sqrt{\gamma^4} = \gamma^2.
\]
In particular,
\[
\gamma \geq p(\alpha_m > \gamma|y_i = 0) \\
\geq p(x_i = 0|y_i = 0) \cdot p(\alpha_m > \gamma|x_i = 0 = y_i) \\
= \frac{p(\alpha_m > \gamma|x_i = 0 = y_i)}{2},
\]
and so
\[
p(\beta_z > \gamma|x_i = 0 = y_i) \leq 2\gamma.
\]
A symmetric argument proves that the probability that the distance of \( p(y_i|z) \) exceeds \( \gamma \) is at most \( 2\gamma \). By the union bound, the probability that \( x_i, y_i \) are \( 2\gamma \) close to uniform is at least \( 1 - 4\gamma \).

\[\square\]

**Matching Polytope**

We defined the matching polytope \( M_n \) as the convex hull of all matchings in a graph on \( n \) vertices. Here we prove\(^{10}\):

**Theorem 11.21.** The extension complexity of the matching polytope is at least \( 2^{\Omega(n)} \).

Like the lower bound for the correlation polytope, the proof will rely crucially on entropy based inequalities. To prove the theorem, we first identify a slack matrix of the matching polytope. Consider the set of inequalities:

\[
\sum_{u < v \in A} z_{(u,v)} \leq \frac{|A| - 1}{2} \quad \text{for all } A \subseteq [n], \text{ with } |A| \text{ odd}
\]

These inequalities hold for every matching \( z \), because the number of edges contained in a set \( A \) can be at most \( |A|/2 \), and since this number is an integer, it is at most \( (|A| - 1)/2 \) when \( |A| \) is odd. When \( n \) is even and \( z \) is a perfect matching—a matching that includes every vertex—the slack \( S_{A,z} \) is exactly \( \frac{k-1}{2} \), where \( k \) is the number of edges in \( z \) that go from \( A \) to its complement. To see this observe that
\[
k + 2 \cdot \sum_{u < v \in A} z_{(u,v)} = |A|
\]
\[
\implies \frac{|A| - 1}{2} - \sum_{u < v \in A} z_{(u,v)} = \frac{k - 1}{2}.
\]

\[\text{We make a slight change in notation here.}\]

\[\text{Rothvoß, 2014}\]
Let \( V \) denote the set of vertices of the polytope that correspond to matchings. Let \( S \) be the slack matrix that corresponds to the inequalities defined above, and \( V \). Given Theorem 11.15, it is enough to prove:

**Lemma 11.22.** \( \text{rank}_+(S) \geq 2^\Omega(n) \).

The proof of the lower bound closely follows ideas developed to prove lower bounds on the randomized communication complexity of disjointness. Consider the distribution on \((x, y)\) given by:

\[
q(xy) = \frac{S_{x,y}}{\sum_{i,j} S_{i,j}}
\]

If \( S \) has non-negative rank \( r \), then \( q(xy) \) can be expressed as a convex combination of \( r \) product distributions: there is a distribution \( q(m) \) on \( [r] \), and for each \( m \in [r] \), there is a product distribution \( q(xy|m) \) on \((x, y)\), so that

\[
q(xy) = \sum_m q(m) \cdot q(xy|m).
\]

For ease of notation, it will be convenient to work with \( 4n + 6 \) vertices. Let \( A \) be a perfect matching of all the vertices and let \( \mathcal{W} \) be a subset of the vertices that cuts all of the edges of \( A \). Let \( F = (C, B_1, \ldots, B_n) \) be a uniformly random partition of \( \mathcal{W} \) such that \( |C| = 3 \), and \( |B_i| = 2 \) for all \( i \). Let \( A_i \) denote the set of edges of \( A \) that touch \( B_i \).

We say that a set \( x \) is consistent with \( F \) if \( C \subseteq x \subseteq \mathcal{W} \), and \( x_i = x \cap B_i \) is not of size 1 for any \( i \). We say that a perfect matching \( y \) is consistent with \( F \) if \( y \) contains the 3 edges of \( A \) cut by \( C \), and for each \( i \), either \( A_i \not\subseteq y \), or \( y \) matches \( B_i \) to itself and matches the neighbors of \( B_i \) under \( A_i \) to themselves. We write \( y_i \) to denote the edges of \( y \) contained in \( A_i \).

Let \((X, Y)\) be sampled according to the distribution \( q \). Let \( D \) denote the event that \((X, Y)\) are consistent with \( F \), and for every \( i \), the set \( X_i \) does not cut the edges of \( Y_i \). We show that for every \( i \),

\[
I(X_i : M|X_{<i}Y_{\geq i}FD) + I(Y_i : M|X_{<i}Y_{>i}FD) \geq \Omega(1). \tag{11.2}
\]

Given the event \( D \), for each fixing of \( F \), the pairs

\[(X_1, Y_1), \ldots, (X_n, Y_n)\]

are mutually independent. Thus by Lemma 6.14, we get

\[
2 \log r \geq \sum_{i=1}^n I(X_i : M|X_{<i}Y_{\geq i}FD) + I(Y_i : M|X_{<i}Y_{>i}FD)
\]

\[
\geq \Omega(n), \quad M \text{ is supported on } r \text{ elements.}
\]

by (11.2)
proving that \( r \geq 2^{\Omega(n)} \) as required.

Fix \( i \) for the rest of the proof. Let \( \mathcal{U} \) denote the event that \((x, y)\) are consistent with the partition \( F \), and for each \( j \neq i \), the edges of \( y_j \) are not cut by \( x_j \). So \( D \) implies \( \mathcal{U} \), but under \( \mathcal{U} \) the edges of \( y_i \) may be cut by \( x_i \). Let \( Z \) denote the random variable \( Z = (X_{<i}, Y_{>i}, B_{<i}, B_{>i}) \).

In fact, we prove the stronger statement that for each fixing \( Z = z \), we have \( \gamma \geq \Omega(1) \) where

\[
\gamma^4 / 2 = \mathbb{I}(X_i : M | Y_i CzD) + \mathbb{I}(Y_i : M | X_i CzD).
\]

Fix \( z \) for the rest of the proof, and let

\[
p(xy | c, m) = q(xy | c, m | z) \quad \text{and} \quad q(x, y | c, m | z).
\]

Given \( c, m \), the distribution \( p(x, y | c, m) \) is supported on four possible values. Let \( a_{cm} \) denote the distance of this distribution from the uniform distribution on these four values. Call a pair \((c, m)\) good if \( a_{cm} \leq \gamma \). Denote by \( \mathcal{G} \) the set of good pairs. Let \( \mathcal{Q} \) denote the event that \( X_i = \emptyset, Y_i \neq A_i \), and \( \mathcal{I} \) denote the event that \( X_i \neq \emptyset, Y_i = A_i \). As in the lower bounds for disjointness and the correlation polytope, we shall use Pinsker’s inequality to prove:

**Claim 11.23.** \( p((c, m) \in \mathcal{G} | \mathcal{Q}) \geq 1 - 4\gamma. \)

We shall appeal to the combinatorial structure of matchings to argue that:

**Claim 11.24.** For any value of \( m \), we have \( |\{c : (c, m) \in \mathcal{G}\}| \leq 4. \)
Let us see how to use these claims to complete the proof of the lemma. First, we need to understand the distribution of \( p(x_i, y_i) \). Let \( L = (C, X_{< i}, X_{> i}, Y_{< i}, Y_{> i}) \). Then for every fixing of \( L \), we have that the distribution of \( X, Y \) under \( p \) is determined by a \( 2 \times 2 \) submatrix of \( S \) of the form:

\[
\begin{pmatrix}
Y_{i \neq A_i} & Y_{i = A_i} \\
X_{i = 0} & 1 & 1 \\
X_{i \neq 0} & 1 & 2
\end{pmatrix}.
\]

(11.3)

Namely, the slack of the entry is either \( 1 = (3 - 1)/2 \), when the 3 edges of \( C \) cross the cut, or \( 2 = (5 - 1)/2 \) when the edges of \( C \) and \( A_i \) cross the cut. Here lies a crucial difficulty in the proof. In the disjointness matrix, the corresponding submatrix is of the form

\[
\begin{pmatrix}
Y_{i} = 0 & Y_{i} = 1 \\
X_{i = 0} & 1 & 1 \\
X_{i = 1} & 1 & < 1
\end{pmatrix}.
\]

In that proof, we showed that there are many rectangles where the entry corresponding to \( X_{i} = 1 = Y_{i} \) gets as much weight as the entry corresponding to \( X_{i} = 0 = Y_{i} \), giving a contradiction. For the slack matrix of the matching polytope, the entry corresponding to intersections is larger than the entries corresponding to disjoint sets. The freedom in the choice of \( C \), and Claim 11.24 allow us to avoid this counterexample. Roughly speaking, we partition the weight of the \( 2 \) entry into many parts, thus getting a number smaller than 1.

Now we compute

\[
p(I) = \sum_m p(I, m) \geq \frac{1}{4} \cdot \sum_{(c, m) \in G} p(I, m)
\]

by Claim 11.24

\[
= \frac{5}{4} \cdot \sum_{(c, m) \in G} p(I, m) \cdot p(c | I, m)
\]

since given \( X_i \neq \emptyset, Y_i = A_i, t \), the conditional distribution on \( c \) is uniform

\[
= \frac{5}{2} \cdot \frac{1 - 8\gamma}{1 + 8\gamma} \cdot \sum_{(c, m) \in G} p(Q, c, m)
\]

by (11.4)

\[
\geq \frac{5}{2} \cdot \frac{1 - 8\gamma}{1 + 8\gamma} \cdot (1 - 4\gamma) \cdot p(Q).
\]

On the other hand, by (11.3), \( p(I) \leq 2 \cdot p(Q) \). Since \( 5/2 > 2 \), we must have \( \gamma \geq \Omega(1) \).

Now we turn to proving each of the two claims.
Proof of Claim 11.24. Fix any value of $m$. Under $p$, the set $C$ is a subset of size 3 in a universe of size 5. We claim that if $c$ and $c'$ are two sets with $|c \cap c'| \leq 1$, then we cannot have $(c, m) \in \mathcal{G}$ and $(c', m) \in \mathcal{G}$.

Indeed, assume towards a contradiction that $(c, m), (c', m) \in \mathcal{G}$ are as in Figure 11.13. Since $a_{cm} \leq \gamma < 1/2$, we know that $p(X_i = \emptyset | c t) > 0$, and the $X$ shown in Figure 11.13 has positive probability conditioned on $m$. Similarly, the edges shown in Figure 11.13 have positive probability conditioned on $m$. However, since $X, Y$ are independent conditioned on $m$, both have positive probability conditioned on $m$. But this cannot happen, since this configuration corresponds to an entry of $S$ that is 0, so it has 0 probability in $p$.

This means that all of the sets $c, c'$ that are in $\mathcal{G}$ must intersect in at least 2 elements. We claim that there can be at most 4 such sets. Indeed, take any two sets $c_1 \neq c_2$ in such a family. Assume without loss of generality that $c_1 = \{1, 2, 3\}$ and $c_2 = \{1, 2, 4\}$. If every other subset $c_3$ is contained in $[4]$, then indeed there are at most $\binom{4}{3} = 4$ sets. Otherwise, there is a set $c_3$ such that $5 \in c_3$. Then $\{1, 2\} \subset c_3$, otherwise $c_3$ cannot share two elements with each of $c_1, c_2$. Now, if there is a fourth set $c_4$ in the family then $c_4$ cannot include both 1, 2, since then it will be equal to $c_1, c_2$ or $c_3$. But if $c_4$ includes only one of the elements of $\{1, 2\}$, it will intersect one of $c_1, c_2, c_3$ in just one element, a contradiction. \hfill \Box

Proof of Claim 11.23. Recall
\[ I(X_i : M|Y_i, CzD) + I(Y_i : M|X_i, CzD) = \gamma^4/2. \]

Since both $q(Y_i \neq A_i | zD)$ and $q(X_i = \emptyset | zD)$ are at least $\frac{1}{2}$, we get
\[ I(X_i : M|Y_i, C, Y_i \neq A_i, zD) + I(Y_i : M|C, X_i = \emptyset, zD) \leq \gamma^4 \]

Since the events $X_i = \emptyset$ and $U$ together imply $D$,
\[ p(xycm | X_i = \emptyset) = q(xycm | z, X_i = \emptyset, D). \]

Let $\beta_{cm}$ denote the statistical distance of $p(x_i | cm)$ from uniform. By Pinsker’s inequality (Corollary 6.11):
\[ \mathbb{E}_{p(cm|X_i = \emptyset)} [\beta_{cm}] \leq \sqrt{\mathbb{E}_{p(cm|X_i = \emptyset)} [\beta_{cm}^2]} \leq \sqrt{\gamma^4} \leq \gamma^2. \]
This implies that $p(\beta_{cm} > \gamma | X_i = \emptyset) \leq \gamma$. So

$$p(\beta_{cm} > \gamma | X_i = \emptyset, Y_i \neq A_i) \leq \frac{p(\beta_{cm} > \gamma | X_i = \emptyset)}{p(Y_i \neq A_i | X_i = \emptyset)} \leq 2\gamma.$$ 

A symmetric argument proves that the statistical distance of $p(y_i | cm)$ is at most $\gamma$ except with probability $2\gamma$. Since $p(x_i, y_i | cm)$ is a product distribution, the claim then follows using the union bound. \hfill \Box

**Exercise 11.1**

Prove Fact 11.1.

**Exercise**

Prove that the intersection of two polytopes is also a polytope.

**Exercise 11.2**

Give a factorization of the slack matrix of the permutahedron $S = UV$, where $U$ is a non-negative $2^n - 2 \times r$ matrix, $V$ is a non-negative $r \times n!$ matrix, and $r = O(n^2)$.

**Exercise 11.3**

The cube in $n$ dimensions is the convex hull of the set $\{0, 1\}^n$. Identify the facets of the cube. Is it possible that the extension complexity of the cube is $O(\sqrt{n})$?

**Exercise 11.4**

Show that the non-negative rank of the slack matrix of a regular $2^k$ sided polygon in the plane is at most $O(k)$ by giving a factorization of the matrix into non-negative matrices.

**Exercise 11.5**

Given two disjoint sets $A, B$, each of size $n$, define the bipartite matching polytope to be the convex hull of all bipartite matchings: matchings where every edge goes from $A$ to $B$. Using what we know about the permutahedron, show that the extension complexity of the bipartite matching polytope is at most $O(n^2)$.

**Exercise 11.6**

Show that there is a $k$ for which the convex hull of cliques of size $k$ has extension complexity $2^{O(n)}/n$.

**Exercise 11.7**
The cut polytope $K_n$ is the convex hull of all cuts in a graph. Here the number of variables in $\binom{n}{2}$, one variable for each potential edge $e \subseteq [n]$ of size 2. For every set $A \subseteq [n + 1]$ define the vertex

$$y^A_e = \begin{cases} 1 & \text{if } |e \cap A| = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The cut polytope is the convex hull of all these points. Observe that $y^A = y^{A^c}$, so there are $2^n$ vertices. Prove that the extension complexity of the cut polytope is $2^{\Omega(n)}$. HINT: Find an invertible linear map that maps the cut polytope to the correlation polytope.
In general, networks may be asynchronous, and one can either charge or not charge for the length of each message. Moreover, one can also study the model in which some of the parties do not follow the protocol, or the protocol is disrupted by adversarial actions. In this chapter we stick to the model of synchronous networks, where we count both the number of rounds of communication and the total communication. We also assume that all parties execute the protocol correctly, there are no errors in the communication, and there are no participants who do not follow the protocol.

Cole and Vishkin, 1986

12

Distributed Computing

The study of distributed computing has become increasingly relevant with the rise of the internet, and in the use of smartphones, and similar devices. In a distributed computing environment, there are \( n \) parties that are connected together by a communication network, yet no single party knows exactly what the network is, and they wish to achieve some local or global goal.

More formally, the network is defined by an undirected graph on \( n \) vertices, where each of the vertices represents one of the parties. Each protocol begins with each party knowing its own name. In each round, each of the parties can send a message to all of their neighbors. In this way, the parties seek to achieve some common goal.

Coloring the Network

Suppose the parties in a distributed environment want to properly color themselves so that no two neighboring parties have the same color. Let us start with the example that \( n \) parties are connected together so that every party has at most 2 neighbors.

Here is a simple protocol that finds a coloring with 3 colors and takes \( O(n) \) rounds and communication. The party whose name is 1 colors itself 1 and sends its color to its neighbors. In the next round, each party that receives a message colors itself with one of the 3 colors, given the color of 1, and then sends its color to its neighbors. In this way, in each round some new vertex colors itself. Thus, every vertex obtains a color after \( n \) rounds.

Here is another protocol\(^1\) that finds a proper coloring with a constant number of colors in \( O(\log^* n) \) rounds of communication. Initially, each party colors itself with its name. This is a proper coloring. The goal now is to iteratively reduce the number of colors.

\(^1\) Cole and Vishkin, 1986

In general, networks may be asynchronous, and one can either charge or not charge for the length of each message. Moreover, one can also study the model in which some of the parties do not follow the protocol, or the protocol is disrupted by adversarial actions. In this chapter we stick to the model of synchronous networks, where we count both the number of rounds of communication and the total communication. We also assume that all parties execute the protocol correctly, there are no errors in the communication, and there are no participants who do not follow the protocol.
In each round, the parties send all of their neighbors their current color. If \( a \in \{0,1\}^t \) denotes the color of one of the parties in a round, and \( b,c \in \{0,1\}^t \) denote the colors assigned to its neighbors, then the party sets \( i \) to be a number such that \( a_i \neq b_i \), and \( j \) to be a number such that \( a_j \neq c_j \). Its new color is set to be \((i,j,a_i,a_j)\). The new coloring is proper. Indeed, if the neighbor whose color was \( b \) also gets the color \((i,j,a_i,a_j)\), then we see that \( b_i = a_i \), contradicting the choice of \( i \).

In this way, the number of colors has been reduced from \( t \) to \( O(\lceil \log t \rceil^2) \). After \( O(\log^* n) \) rounds, the number of colors is constant.

This coloring protocol can be generalized to handle arbitrary graphs of constant degree \( d \). Any graph of degree \( d \) can be colored using \( d+1 \) colors. Here we give a protocol\(^\dagger\) that uses \( O(\log^* n) \) rounds to find a coloring using \( O(d^2 \log d) \) colors.

The protocol relies on the following combinatorial lemma:

**Lemma 12.1.** There is a sequence of \( t \) subsets \( T_1, \ldots, T_t \subseteq [m] \) with \( m = 5d^2 \lceil \log t \rceil \), such that for any distinct \( i_1, i_2, \ldots, i_{d+1} \in [t] \), \( T_i \) is not contained in the union of \( T_{i_2}, \ldots, T_{i_{d+1}} \).

**Proof.** Pick the \( t \) sets at random from \([m]\), where each element is included in each set independently with probability \( 1/d \).

For a particular choice of \( S_1, \ldots, S_{d+1} \) from the list of sets and \( j \in [m] \), let \( E_j \) denote the event that \( j \in S_1 \) and \( j \not\in \bigcup_{\ell > 1} S_\ell \). The lemma will be proved if we show that there is positive probability that for every choice of \( S_1, \ldots, S_{d+1} \), one of the events \( E_1, \ldots, E_m \) occurs.

The probability of \( E_j \) occurs

\[
\frac{1}{d} \left( 1 - \frac{1}{d} \right)^d \geq \frac{d}{4}. 
\]

Hence, the probability that no \( E_j \) occurs is at most

\[
\left( 1 - \frac{d}{4} \right)^m < e^{-d \log t}. 
\]

On the other hand, the number of choices for \( d+1 \) such sets from the family is \( \binom{t}{d+1} \leq t^{d+1} < 2^{d \log t} \). Thus, by the union bound, the probability that the family does not have the property we need is at most \( e^{-d \log t} \cdot 2^{d \log t} < 1 \).

\(\square\)

In each round of the protocol, each party sends its current color to all its neighbors. If there are \( t \) colors in a particular round, each party looks at the \( d \) colors its received and associates each with a set from the family promised by Lemma 12.1. Its new color is an element that belongs to her own set but not to any of the others. Thus, the
next round will have at most \(5d^2 \lceil \log t \rceil\) colors. Continuing in this way, the number of colors will be reduced to \(O(d^2 \log d)\) in \(O(\log^* n)\) rounds.

**Computing the Diameter of the Network**

Suppose the parties in a network want to compute the diameter of the network; namely, the maximum distance between two vertices in the graph of the network. Here we show\(^3\):

**Theorem 12.2.** In any distributed algorithm for computing the diameter of an \(n\)-vertex graph, there are \(\Theta(n)\) edges on which \(\Omega(n^2)\) bits are transmitted in total.

This holds even if the goal is to distinguish whether the diameter of the graph is at most 2 or at least 3, and when the protocol is allowed to be randomized.

The proof is by reduction to the randomized communication complexity of disjointness, when there are just 2 parties. The idea for the reduction from \(n\) parties to 2 parties is quite general. We shall partition the \(n\) vertices of the graph to 2 parts, and think of one part as Alice and of the other part as Bob. The messages between the 2 parts will be viewed as communication between Alice and Bob.

Let \(X, Y \subseteq [n] \times [n]\) be two subsets. For every such pair of sets, we shall define a graph \(G_{X,Y}\). We show that if there is an efficient distributed algorithm for computing the diameter of \(G_{X,Y}\), then there is an efficient communication protocol for computing whether or not \(X\) and \(Y\) are disjoint.

The graph \(G_{X,Y}\) has \(4n + 2\) vertices. Let \(A = \{a_1, \ldots, a_n\}, B = \{b_1, \ldots, b_n\}, C = \{c_1, \ldots, c_n\}\) and \(D = \{d_1, \ldots, d_n\}\) be disjoint cliques, each of size \(n\). Let \(v\) be a vertex that is connected to all the vertices in \(A \cup B\) and \(w\) be a vertex that is connected to all the vertices in \(C \cup D\). We connect \(v\) and \(w\) with an edge as well. For each \(i\), we connect \(a_i\) to \(c_i\), and \(b_i\) to \(d_i\).

We encode \(X, Y\) as follows. We connect \(a_i\) to \(b_j\) if and only if \((i, j) \notin X\), and we connect \(c_i\) to \(d_j\) if and only if \((i, j) \notin Y\).

**Claim 12.3.** The diameter of \(G_{X,Y}\) is 2 if \(X, Y\) are disjoint, and 3 if \(X, Y\) are not disjoint.

**Proof.** The interesting part of the claim are the distances between \(A\) and \(D\), and between \(B\) and \(C\)—the distances between all other pairs of vertices are at most 2. Here we focus on the distances between \(A\) and \(D\) the case of \(B\) and \(C\) is similar. When \((i, j) \notin X\) or \((i, j) \notin Y\), the distance of \(a_i\) from \(d_j\) is at most 2; for example, if \((i, j) \notin X\), we have

\[^{3}\text{Frischknecht et al., 2012; and Holzer and Wattenhofer, 2012}\]
the path $a_i \rightarrow b_j \rightarrow d_j$. Otherwise, $(i, j) \in X \cap Y$, and in this case the distance from $a_i$ to $d_j$ is at least 3.

Consider the protocol obtained when Alice simulates all the vertices in $A, B$ and $v$, and Bob simulates all the vertices in $C, D$ and $w$. This protocol solves the disjointness problem, and so has communication at least $\Omega(n^2)$. This proves that the $O(n)$ edges that cross from the left to the right in the above network must carry at least $\Omega(n^2)$ bits of communication to compute the diameter of the graph.

**Computing the Girth of the Network**

Another basic measure associated with a graph is its girth, which is the length of the shortest cycle in the graph. Here we show\textsuperscript{4}:

**Theorem 12.4.** Any distributed protocol for computing the girth of an $n$ vertex graph must involve at least $\Omega(n^2 2^{-O(\sqrt{\log n})})$ bits of communication.

The lower bound holds even if the goal is to detect if the girth is 3 or more—it is even hard to determine if there is a single triangle in the graph.

\textsuperscript{4} Drucker et al., 2014
The proof is by reduction to disjointness in the number-on-forhead model with 3 parties. The reduction together with the lower bound from Theorem 5.12 complete the proof.

Suppose Alice, Bob and Charlie have 3 sets $X, Y, Z \subseteq U$ written on their foreheads, where $U$ is a set that we shall soon specify. We shall define a graph $G_{X,Y,Z}$ that will have a triangle (namely a cycle of length 3) if and only if $X \cap Y \cap Z$ is non-empty.

The vertex set of $G_{X,Y,Z}$ is $A \cup B \cup C$, where $A, B, C$ are disjoint sets, each of size $2n$. To construct $G_{X,Y,Z}$ we need the coloring promised by Theorem 4.2. This is a coloring of $[n]$ with $2^{O(\sqrt{\log n})}$ colors such that there are no 3-term monochromatic arithmetic progressions. Since such a coloring exists, there must be a subset $Q \subseteq [n]$ of size $n2^{-O(\sqrt{\log n})}$ that does not contain any non-trivial arithmetic progressions.

First define a graph $G$ on the vertex set $A \cup B \cup C$, where for each $a \in A, b \in B, c \in C,$

\[
\{a,b\} \in E(G) \iff b-a \in Q,
\{b,c\} \in E(G) \iff c-b \in Q,
\{a,c\} \in E(G) \iff c-a \in Q.
\]

**Claim 12.5.** The graph $G$ has at least $n|Q| = n^22^{-O(\sqrt{\log n})}$ triangles, and no two distinct triangles in $G$ share an edge.

**Proof.** For each element $q \in Q$, the vertices $a \in A, a+q \in B, a+2q \in C$ certainly form a triangle, as long as $a+2q \leq 2n$. No two of these triangles share an edge, since any edge of these triangles determines $a, q$.

We claim that there are no other triangles. Indeed, if $a, b, c$ was a triangle in the graph, then $b-a = q_1, c-b = q_2 \in Q$, and $q_1 + q_2 = c-b + b-a = c-a = q_3 \in Q$.

So $q_1, q_3, q_2 \in Q$ form an arithmetic progression. The only way this can happen is if $q_1 = q_2 = q_3$. In this case we recover one of the triangles above.

The universe $U$ is the set of triangles in $G$. The graph $G_{X,Y,Z}$ is the subgraph of $G$ defined by

\[
\{a,b\} \in E(G_{X,Y,Z}) \iff \text{a triangle of } G \text{ containing } \{a,b\} \text{ is in } Z,
\{b,c\} \in E(G_{X,Y,Z}) \iff \text{a triangle of } G \text{ containing } \{b,c\} \text{ is in } X,
\{a,c\} \in E(G_{X,Y,Z}) \iff \text{a triangle of } G \text{ containing } \{a,c\} \text{ is in } Y.
\]

The following simple property holds:

Figure 12.2: The graph $G$.

Figure 12.3: The graph $G_{X,Y,Z}$.
Claim 12.6. $G_{X,Y,Z}$ contains a triangle if and only if $X \cap Y \cap Z \neq \emptyset$.

Proof. If $a, b, c$ are the vertices of a triangle in $X \cap Y \cap Z$, then they form a triangle in $G_{X,Y,Z}$. Conversely, if $G_{X,Y,Z}$ contains a triangle, then each edge of the triangle is contained in a single triangle of $G$, by Claim 12.5. This implies that the 3 edges define a triangle in $X \cap Y \cap Z$. \qed

Given sets $X, Y, Z$ as input, Alice, Bob and Charlie execute the protocol for detecting triangles in the network $G_{X,Y,Z}$, with Alice, Bob, Charlie simulating the behavior of the nodes in $A, B, C$ of the network, respectively. Each of the players knows enough information to simulate the behavior of these nodes—for example, Alice knows the neighbors of $A$, since she knows $Y, Z$. By Theorem 5.12, the total communication of the protocol must be at least $\Omega(|U|)$. 


Bibliography


Pavel Hrubeš. Personal communication, 2016.


