What are the limitations to what computers can learn?

Do certain mathematical theorems have short proofs?

Can quantum mechanics be exploited to speed up computation?

Is every problem whose solution is efficiently verifiable also efficiently solvable? i.e. $P = NP$?
What are the limitations to what computers can learn?

Do certain mathematical theorems have short proofs?

Can quantum mechanics be exploited to speed up computation?

Is every problem whose solution is efficiently verifiable also efficiently solvable? i.e. $P = NP$?
Communication complexity
Cool Things About Communication Complexity

Many useful applications:

*machine learning, proof complexity, quantum computation, pseudorandom generators, data structures, game theory,* …

The setting is simple and neat.

Beautiful mathematics

*combinatorics, algebra, analysis, information theory,* …
Motivating Example 1: Checking Equality

How many bits need to be communicated?

Naively: \( n \)  
Actually: \( n \)

What if we allow 0.00000000001% probability of error?

Naively: \( \Omega(n) \)  
Actually: \( O(\log n) \)
Motivating Example 2: Auctions

Alice

$100

Bob

$1000
Defining the model a bit more formally
2 Player Model of Communication Complexity

Goal: Compute $F(x, y)$. (both players should know the value)

How: Sending bits back and forth according to a protocol.

Resource: Number of communicated bits.

(We assume players have unlimited computational power individually.)
Poll 1

$x, y \in \{0, 1\}^n$, \hspace{1em} PAR(x, y) = \text{parity of the sum of all the bits.}

(i.e. it’s 1 if the parity is odd, 0 otherwise.)

How many bits do the players need to communicate?
Choose the tightest bound.

\[
\begin{align*}
O(1) \\
O(\log n) \\
O(\log^2 n) \\
O(\sqrt{n}) \\
O(n/ \log n) \\
O(n)
\end{align*}
\]
Once \( \text{Bob} \) knows the parity of \( x \), he can compute \( \text{PAR}(x, y) \).

- \text{Alice} sends \( \text{PAR}(x) \) to \text{Bob}. \hspace{1cm} \text{1 bit}
- \text{Bob} computes \( \text{PAR}(x, y) \) and sends it to \text{Alice}. \hspace{1cm} \text{1 bit}

2 bits in total
2 Player Model of Communication Complexity

\[ F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\} \]

\( x \in \{0, 1\}^n \)\hspace{1cm} \( y \in \{0, 1\}^n \)

(Alice) [smiley face] [hand holding wrench]

known to both players

(Bob) [smiley face] [thumbs up]

Goal: Compute \( F(x, y) \). (both players should know the value)

How: Sending bits back and forth according to a protocol.

Resource: Number of communicated bits.

(we assume players have unlimited computational power individually.)
**Goal:** Compute $F(x, y)$. (both players should know the value)

**How:** Sending bits back and forth according to a **protocol**.

**Resource:** Number of communicated bits.

A protocol $P$ is the “strategy” players use to communicate. It determines what bits the players send in each round. $P(x, y)$ denotes the output of $P$. 
2 Player Model of Communication Complexity

**Goal:** Compute $F(x, y)$. (both players should know the value)

**How:** Sending bits back and forth according to a protocol.

**Resource:** Number of communicated bits.

A (deterministic) protocol $P$ computes $F$ if

$$\forall (x, y) \in \{0, 1\}^n \times \{0, 1\}^n, \quad P(x, y) = F(x, y)$$

Analogous to:

- algorithm (TM)
- decision problem

$$\forall x \in \Sigma^* \quad A(x) = F(x)$$
**2 Player Model of Communication Complexity**

**Goal:** Compute $F(x, y)$. (both players should know the value)

**How:** Sending bits back and forth according to a protocol.

**Resource:** Number of communicated bits.

A randomized protocol $P$ computes $F$ with $\epsilon$ error if

$$\forall (x, y) \in \{0, 1\}^n \times \{0, 1\}^n, \quad \Pr[P(x, y) \neq F(x, y)] \leq \epsilon$$

**Analogous to:** Monte Carlo algorithms
2 Player Model of Communication Complexity

**Goal:** Compute $F(x, y)$. (both players should know the value)

**How:** Sending bits back and forth according to a **protocol**.

**Resource:** Number of communicated bits.

\[
\text{cost}(P) = \max_{(x,y)} \# \text{ bits } P \text{ communicates for } (x, y)
\]

if $P$ is randomized, you take max over the random choices it makes.

**Deterministic communication complexity**

$D(F) = \min \text{ cost of a (deterministic) protocol computing } F.$

**Randomized communication complexity**

$R^\epsilon(F) = \min \text{ cost of a randomized protocol computing } F$ with $\epsilon$ error.
2 Player Model of Communication Complexity

**Goal:** Compute $F(x, y)$. (both players should know the value)

**How:** Sending bits back and forth according to a protocol.

**Resource:** Number of communicated bits.

\[
\text{cost}(P) = \max \# \text{ bits } P \text{ communicates for } (x, y) \quad (\text{max over all } (x, y))
\]

*Deterministic communication complexity*:

\[
\mathcal{D}(F) = \min \text{ cost}(P) \quad (\text{min over all deterministic protocols computing } F).
\]

*Randomized communication complexity*:

\[
\mathcal{R}^\varepsilon(F) = \min \text{ cost}(P) \quad (\text{min over all randomized protocols computing } F \text{ with } \varepsilon \text{ error}).
\]

We usually fix $\varepsilon$ to some constant. 

*E.g.* $\varepsilon = 1/3$

We can always boost the success probability if we want.
What is considered hard or easy?

\[ F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\} \]

\[ 0 \leq R_2^c(F) \leq D_2(F) \leq n + 1 \]

\[ c \log^c(n) \quad n^\delta \quad \delta n \]
Example

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

\[ D(EQ) = n + 1. \quad R^{1/3}(EQ) = O(\log n). \]
$M A J(x, y) = 1$ iff majority of all the bits in $x$ and $y$ are set to 1.

What is $D(MAJ)$? Choose the tightest bound.

- $O(1)$
- $O(\log n)$
- $O(\log^2 n)$
- $O(\sqrt{n})$
- $O(n / \log n)$
- $O(n)$
\( \text{MAJ}(x, y) = 1 \) iff majority of all the bits in \( x \) and \( y \) are set to 1.

What is \( D(\text{MAJ})? \) Choose the tightest bound.

The result can be computed from

\[
\sum_{i \in \{1,2,\ldots,n\}} x_i + \sum_{i \in \{1,2,\ldots,n\}} y_i
\]

- Alice sends \( \sum_i x_i \) to Bob. \( \sim \log n \) bits

- Bob computes \( \text{MAJ}(x, y) \) and sends it to Alice. 1 bit

\( O(\log n) \) in total
Another example

Disjointness: \( DISJ(x, y) = \begin{cases} 
0 & \text{if } \exists i : x_i = y_i = 1 \\
1 & \text{otherwise}
\end{cases} \)

\[ R^{1/3}(DISJ) = \Omega(n). \] hard
The plan

1. Efficient randomized communication protocol for checking equality.

2. An application of communication complexity.

3. How to prove lower bounds.
Efficient randomized communication protocol for checking equality
The Power of Randomization

\[ R^{1/3}(EQ) = O(\log n). \]

Alice gets \( x \in \{0, 1\}^n \), Bob gets \( y \in \{0, 1\}^n \).

We treat \( x \) and \( y \) as numbers: \( 0 \leq x, y \leq 2^n - 1 \).

**The Protocol:**

- Let \( p_i \) be the \( i \)'th smallest prime number.
  \[ p_1 = 2, \ p_2 = 3, \ p_3 = 5, \ p_4 = 7, \ldots \]

- Alice picks a random \( i \in \{1, 2, \ldots, n^2\} \).

- Alice sends Bob: \( i, \ x \mod p_i \)

- Bob outputs 1 iff \( x \mod p_i = y \mod p_i \). \( (x \equiv_{p_i} y) \)
The Power of Randomization

\[ R^{1/3}(EQ) = O(\log n). \]

**Correctness:**

*Want to show:* For all inputs \((x, y)\), probability of error is \(\leq 1/3\).

For all \((x, y)\) with \(x = y\):

\[ \Pr[\text{error}] = \Pr_i[p_i \neq y] = 0. \]

For all \((x, y)\) with \(x \neq y\):

\[ \Pr[\text{error}] = \Pr_i[x \equiv p_i \ y] = \Pr_i[p_i \text{ divides } x - y] \]

**Claim:** \(x - y\) has at most \(n\) distinct prime factors.

\[ \Pr[\text{error}] = \Pr[p_i \text{ is a prime factor of } x - y] \leq \frac{n}{n^2} = \frac{1}{n}. \]
The Power of Randomization

\[ \mathbb{R}^{1/3}(EQ) = O(\log n). \]

Cost:
The only communication is:
- Alice sends Bob: \( i, x \mod p_i \)

The first number \( i \) is such that \( i \leq n^2 \).
Can represent it using \( \sim \log_2 n^2 = 2 \log_2 n = O(\log n) \) bits.

The second number \( x \mod p_i \) is at most \( p_n^2 \).
By the Prime Number Theorem: \( p_n^2 \sim n^2 \log n^2 \leq 2n^3 \)
Can represent \( p_n^2 \) using at most \( \log(2n^3) = O(\log n) \) bits.
The plan

1. Efficient randomized communication protocol for checking equality.

2. An application of communication complexity.

3. How to prove lower bounds.
An application of communication complexity
Applications of Communication Complexity

- circuit complexity
- time/space tradeoffs for Turing Machines
- VLSI chips
- machine learning
- game theory
- data structures
- proof complexity
- pseudorandom generators
- pseudorandomness
- branching programs
- data streaming algorithms
- quantum computation
- lower bounds for polytopes representing NP-complete problems
Applications of Communication Complexity

- circuit complexity
- time/space tradeoffs for Turing Machines
- VLSI chips
- machine learning
- game theory
- data structures
- proof complexity

- pseudorandom generators
- pseudorandomness
- branching programs
- data streaming algorithms
- quantum computation
- lower bounds for polytopes representing NP-complete problems
How Communication Complexity Comes In

**Setting:** Solve some task while minimizing some resource.

   e.g. find a fast algorithm, design a small circuit,
   find a short proof of a theorem, …

**Goal:** Prove lower bounds on the resource needed.

**Sometimes:**

   efficient solution to our problem

   efficient communication protocol for a certain function.

i.e. no efficient protocol for the function

   no efficient solution to our problem.
Time/space tradeoffs for TMs
Recall Turing Machines

- **Input tape**: (read only) $000001111$
- **Work tape**: (memory) $100\ldots$

$T(n)$ time: # steps the machine takes

$S(n)$ space: # work tape cells the machine uses
An observation

What information do I need to continue the computation from where you left it?

1. current state
2. positions of tape heads
3. contents of work tape

Suppose we both know the TM $M$ and the input $w$.

You start running $M(w)$.

You pause after a certain number of steps.

What information do I need to continue the computation from where you left it?
An observation

You start running.
You pause after a certain number of steps.

Suppose we both know the TM $M$ and the input $w$.

What information do I need to continue the computation from where you left it?

1. current state
2. positions of tape heads
3. contents of work tape

\[
O(1) \quad O(\log n) + O(\log S(n)) \quad O(S(n))
\]
Let \( L = \{ x\#|x| : x \in \{0, 1\}^* \} \)

- 000###000 \( \in \) \( L \)
- 1010###1010 \( \in \) \( L \)
- 001###000 \( \notin \) \( L \)
- 000###000 \( \notin \) \( L \)

**Theorem:**
If a \( \text{TM} \ M \) decides \( L \) in \( T(n) \) time and \( S(n) \) space on inputs of size \( 3n \), then \( T(n) \cdot S(n) = \Omega(n^2) \).
Time/space tradeoff for a simple language

Let \( L = \{x \# |x| x : x \in \{0, 1\}^*\} \)

**Theorem:**
If a TM \( M \) decides \( L \) in \( T(n) \) time and \( S(n) \) space on inputs of size \( 3n \), then \( T(n) \cdot S(n) = \Omega(n^2) \).

**Strategy:**

Using \( M \), we design a communication protocol for \( EQ \) of cost \( \leq c \frac{T(n)S(n)}{n} \) for some constant \( c \).

We know \( EQ \) requires \( \geq n \) bits of communication.

\[ \quad \implies c \frac{T(n)S(n)}{n} \geq n \implies c T(n)S(n) \geq n^2 \]
Time/space tradeoff for a simple language

Let \( L = \{ x \# | x | x : x \in \{ 0, 1 \}^* \} \). \( M \) decides \( L \).

Protocol for \( EQ \):

Given input \( x \in \{ 0, 1 \}^n \) to Alice, and \( y \in \{ 0, 1 \}^n \) to Bob. They want to decide if \( x = y \). They will make use of \( M \).

Let \( w = x \#^n y \).

They simulate \( M(w) \).

If \( M(w) \) accepts, they output 1.

If \( M(w) \) rejects, they output 0. A correct protocol.
Time/space tradeoff for a simple language

Let \( L = \{x \# |x|x : x \in \{0, 1\}^*\} \). \( M \) decides \( L \).

Protocol for \( EQ \):

Given input \( x \in \{0, 1\}^n \) to Alice, and \( y \in \{0, 1\}^n \) to Bob.

They want to decide if \( x = y \). They will make use of \( M \).

Let \( w = x \#^n y \).

They simulate \( M(w) \).

If \( M(w) \) accepts, they output 1.

If \( M(w) \) rejects, they output 0.

A correct protocol.

How do they simulate \( M \)?

What is the cost?
Let \( L = \{ x\#|x| x : x \in \{0, 1\}^* \} \). \( M \) decides \( L \).

**Protocol for \( EQ \):**

They simulate \( M(x\#^ny) \).

Alice starts the simulation.

When input tape head reaches \( y \) symbol, she sends

1. current state
2. position of work tape head
3. contents of work tape
Let $L = \{ x \# | x | x : x \in \{0, 1\}^* \}$. $M$ decides $L$.

**Protocol for $EQ$:**

They simulate $M(x\#^n y)$.

**Bob** continues the simulation.

When input tape head reaches $x$ symbol, he sends
1. current state
2. position of work tape head
3. contents of work tape

This continues until $M$ halts.
Analysis:

It is clear the protocol is correct. What is the cost?

In each transmission, players send

1. current state $\rightarrow O(1)$
2. position of work tape head $\rightarrow O(\log S(n))$
3. contents of work tape $\rightarrow O(S(n))$

What is the number of transmissions?

For each transmission, $M$ takes $\geq n$ steps.

So $T(n) \geq (\# \text{ transmissions}) \cdot n$.

$\implies \# \text{ transmissions} \leq T(n)/n$.

Total cost: $O(S(n)T(n)/n)$. 
1. Efficient randomized communication protocol for checking equality.

2. An application of communication complexity.

3. How to prove lower bounds.
Lower bounds for deterministic communication complexity
The function matrix

$$F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$$

$$M_F = \begin{array}{cccccccc}
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 
\end{array}
$$

$$M_F[x, y] = F(x, y)$$

$2^n$ by $2^n$ matrix
The function matrix

Equality: \[ EQ(x, y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases} \]

\[ n = 3 \]

\[ M_{EQ} = \begin{array}{cccccccccc}
000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
000 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
001 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
010 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
011 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
100 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
101 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
110 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
111 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array} \]

\[ 2^n \text{ by } 2^n \text{ matrix} \]
The function matrix

How do you prove lower bounds on communication complexity?

You study this matrix!

$$\begin{align*}
M_F[x, y] &= F(x, y) \\
\end{align*}$$
IMPORTANT(!) property of communication protocols:

A protocol partitions function matrix into monochromatic rectangles. What is a rectangle?

A rectangle is of the form $S \times T$ for $S, T \subseteq \{0, 1\}^n$
IMPORTANT(!) property of communication protocols:

A protocol partitions function matrix into monochromatic rectangles.

What is a rectangle?

A **rectangle** is of the form \( S \times T \) for \( S, T \subseteq \{0, 1\}^n \).
Suppose we have a deterministic protocol of cost $c$ that computes a function $F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$.

This protocol partitions $M_F$ into at most $2^c$ monochromatic rectangles.

You will prove this in the homework.
Example

\[ \text{PAR}(x, y) = \sum_{i=1}^{n} x_i + y_i \pmod{2} \]

Protocol:

Alice sends the parity of her input bits.

Bob sends the output of the function.

The cost of the protocol is 2 bits.

This protocol partitions the function matrix into at most 4 monochromatic rectangles.
Example

\[ PAR(x, y) = \sum_{i=1}^{n} x_i + y_i \pmod{2} \]

<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>011</th>
<th>110</th>
<th>101</th>
<th>111</th>
<th>100</th>
<th>010</th>
<th>001</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**Example**

\[ PAR(x, y) = \sum_{i=1}^{n} x_i + y_i \pmod{2} \]

<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>011</th>
<th>110</th>
<th>101</th>
<th>111</th>
<th>100</th>
<th>010</th>
<th>001</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**Example**

\[
PAR(x, y) = \sum_{i=1}^{n} x_i + y_i \pmod{2}
\]
Example

\[ PAR(x, y) = \sum_{i=1}^{n} x_i + y_i \pmod{2} \]

<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>011</th>
<th>110</th>
<th>101</th>
<th>111</th>
<th>100</th>
<th>010</th>
<th>001</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example

\[ PAR(x, y) = \sum_{i=1}^{n} x_i + y_i \pmod{2} \]
Example

\[ PAR(x, y) = \sum_{i=1}^{n} x_i + y_i \pmod{2} \]
Lower bound for Equality function

A protocol for $EQ$ of cost $c$ partitions $M_{EQ}$ into at most $2^c$ monochromatic rectangles.

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Observe:**

Any protocol computing $EQ$ must cover the 1’s with monochromatic rectangles.
Lower bound for Equality function

A protocol for \( EQ \) of cost \( c \) partitions \( M_{EQ} \) into at most \( 2^c \) monochromatic rectangles.

Claim: No two 1’s can be in the same monochromatic rectangle

Proof: Suppose two 1’s are in the same rectangle.
A protocol for $EQ$ of cost $c$ partitions $M_{EQ}$ into at most $2^c$ monochromatic rectangles.

**Claim:** No two 1’s can be in the same monochromatic rectangle

**Proof:**
Suppose two 1’s are in the same rectangle.
Lower bound for Equality function

A protocol for $EQ$ of cost $c$ partitions $M_{EQ}$ into at most $2^c$ monochromatic rectangles.

Claim: No two 1’s can be in the same monochromatic rectangle

Proof: Suppose two 1’s are in the same rectangle. Then there must also be 0’s in the rectangle. Contradiction.

Observe: Any protocol computing $EQ$ must cover the 1’s with monochromatic rectangles.

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
A protocol for \( EQ \) of cost \( c \) partitions \( M_{EQ} \) into at most \( 2^c \) monochromatic rectangles.

**Claim:** No two 1's can be in the same monochromatic rectangle.

**Conclusion:** We need a separate rectangle for each 1. \( \implies \) We need at least \( 2^n \) rectangles to cover the 1s.
A protocol for $EQ$ of cost $c$ partitions $M_{EQ}$ into at most $2^c$ monochromatic rectangles.

**Conclusion**: We need a separate rectangle for each 1.

$\implies$ We need at least $2^n$ rectangles to cover the 1s.

We also need at least one rectangle to cover the 0s.

$$2^c \geq 2^n + 1$$

$\implies c \geq n + 1$

This is true for any protocol computing $EQ$. In particular, it is true for the most efficient protocol.

$$D(EQ) \geq n + 1.$$
Summary of the lower bound technique

Let \( F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\} \).

A lower bound on the number of monochromatic rectangles needed to partition \( M_F \).

A lower bound on \( D(F) \).

Interesting corollary (not hard to prove):

\[
D(F) \geq \log_2 \text{rank}(M_F)
\]
The plan

1. Efficient randomized communication protocol for checking equality.

2. An application of communication complexity.

3. How to prove lower bounds.
Take-Home Message

Communication complexity studies natural distributed tasks.

Communication complexity (lower bounds) has many interesting applications.

Lower bounds can be proved using a variety of tools: combinatorial, algebraic, analytic, information theoretic,…