LECTURE 5

Last time
- Limitations of sublinear-time algorithms
- Yao’s Minimax Principle
  - Examples: testing 0* and sortedness

Today
- Limitations of sublinear-time algorithms
- Yao’s Minimax Principle
- Communication complexity

HW1 resubmission, HW3 out, project guidelines
Recall: Yao’s Minimax Principle

<table>
<thead>
<tr>
<th>Statement 1</th>
</tr>
</thead>
</table>
| For any **probabilistic** algorithm $A$ of complexity $q$ there exists an input $x$ s.t.
| $\Pr_{\text{coin tosses of } A} [A(x) \text{ is wrong}] > 1/3$. |

<table>
<thead>
<tr>
<th>Statement 2</th>
</tr>
</thead>
</table>
| There is a distribution $D$ on the inputs, s.t. for every **deterministic** algorithm of complexity $q$,
| $\Pr_{x \leftarrow D} [A(x) \text{ is wrong}] > 1/3$. |

- Need for lower bounds

**Yao’s Minimax Principle (easy direction):** Statement 2 $\Rightarrow$ Statement 1.

**NOTE:** Also applies to restricted algorithms

- 1-sided error tests
- nonadaptive tests
### Yao’s Minimax Principle as a game

**Players:** Evil algorithms designer Al and poor lower bound prover Lola.

<table>
<thead>
<tr>
<th><strong>Game1</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Move 1.</strong> Al selects a q-query <strong>randomized</strong> algorithm A for the problem.</td>
</tr>
<tr>
<td><strong>Move 2.</strong> Lola selects an input on which A errs with largest probability.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Game2</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Move 1.</strong> Lola selects a distribution on inputs.</td>
</tr>
<tr>
<td><strong>Move 2.</strong> Al selects a q-query <strong>deterministic</strong> algorithm with as large probability of success on Lola’s distribution as possible.</td>
</tr>
</tbody>
</table>
Testing Monotonicity of functions on Hypercube

Non-adaptive 1-sided error

Lower Bound
Boolean Functions $f : \{0, 1\}^n \to \{0, 1\}$

Graph representation: $n$-dimensional hypercube

- **vertices**: bit strings of length $n$
- **edges**: $(x, y)$ is an edge if $y$ can be obtained from $x$ by increasing one bit from 0 to 1

Each vertex $x$ is labeled with $f(x)$
**Boolean Functions** \( f : \{0, 1\}^n \rightarrow \{0, 1\} \)

Graph representation:

- **\(2^n\) vertices**: bit strings of length \(n\)
- **\(2^{n-1}n\) edges**: \((x, y)\) is an edge if \(y\) can be obtained from \(x\) by increasing one bit from 0 to 1
- **each vertex \(x\) is labeled with \(f(x)\)**
Monotonicity of Functions

[A function \( f : \{0,1\}^n \rightarrow \{0,1\} \) is monotone if increasing a bit of \( x \) does not decrease \( f(x) \).

- Is \( f \) monotone or \( \epsilon \)-far from monotone (\( f \) has to change on many points to become monontone)?
  - Edge \( x \rightarrow y \) is violated by \( f \) if \( f(x) > f(y) \).

Time:
- \( O(n/\epsilon) \), logarithmic in the size of the input, \( 2^n \)
- \( \Omega(\sqrt{n}/\epsilon) \) for 1-sided error, nonadaptive tests
- Advanced techniques: \( \Theta(\sqrt{n}/\epsilon^2) \) for nonadaptive tests, \( \Omega(3\sqrt{n}) \)

[Khot Minzer Safra 15, Chen De Servidio Tang 15, Chen Waingarten Xie 17]
## Hypercube 1-sided Error Lower Bound

**Lemma** [Fischer Lehman Newman Raskhodnikova Rubinfeld Samorodnitsky]

Every **1-sided error nonadaptive** test for monotonicity of functions $f : \{0,1\}^n \rightarrow \{0,1\}$ requires $\Omega(\sqrt{n})$ queries.

- 1-sided error test must accept if no violated pair is uncovered.

Violated pair: $1 \rightarrow 0$

- A distribution on far from monotone functions suffices.
Hypercube 1-sided Error Lower Bound

- Hard distribution: pick coordinate $i$ at random and output $f_i$.

\[
f_i(x) = \begin{cases} 
1 & \text{if } |x| > \frac{n}{2} + \sqrt{n} \\
1 - x_i & \text{if } |x| = \frac{n}{2} \pm \sqrt{n} \\
0 & \text{if } |x| < \frac{n}{2} - \sqrt{n}
\end{cases}
\]
Hypercube 1-sided Error Lower Bound

- Hard distribution: pick coordinate $i$ at random and output $f_i$.

- A “truncation” of an antidicator

\[ f_i(x) = \begin{cases} 
1 & \text{if } |x| > \frac{n}{2} + \sqrt{n} \\
1 - x_i & \text{if } |x| = \frac{n}{2} \pm \sqrt{n} \\
0 & \text{if } |x| < \frac{n}{2} - \sqrt{n}
\end{cases} \]
**The Fraction of Nodes in Middle Layers**

**Hoeffding Bound**

Let $Y_1, \ldots, Y_s$ be independently distributed random variables in [0,1].

Let $Y = \frac{1}{s} \cdot \sum_{i=1}^{s} Y_i$ (called *sample mean*). Then $\Pr[|Y - E[Y]| \geq \varepsilon] \leq 2e^{-2s\varepsilon^2}$.

<table>
<thead>
<tr>
<th>$E[Y]$=</th>
<th>$\varepsilon$=</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{n}{2} \pm \sqrt{n}$</td>
<td>$1 - \text{coordinate } i$</td>
</tr>
</tbody>
</table>
Hard Functions are Far

- Hard distribution: pick coordinate $i$ at random and output $f_i$.

$f_i(x) = \begin{cases} 
1 & \text{if } |x| > \frac{n}{2} + \sqrt{n} \\
1 - x_i & \text{if } |x| = \frac{n}{2} \pm \sqrt{n} \\
0 & \text{if } |x| < \frac{n}{2} - \sqrt{n}
\end{cases}$

Analysis

- The middle contains a constant fraction of vertices.
- Edges from $(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$ to $(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$ are violated if both endpoints are in the middle.
- All $n$ functions are $\varepsilon$-far from monotone for some constant $\varepsilon$. 
Hypercube 1-sided Error Lower Bound

• How many functions does a set of $q$ queries expose?

Naive Analysis

- # functions exposed by $q$ queries
  \[ \leq q^2 \cdot 2\sqrt{n} \]

- # functions that a query pair $(x, y)$ exposes
  \[ \leq \# \text{ coordinates on which } x \text{ and } y \text{ differ} \]
  \[ \leq 2\sqrt{n} \]

Only pairs of queries in the Green Band can be violated ⇒ disagreements $\leq 2\sqrt{n}$
Hypercube 1-sided Error Lower Bound

- How many functions does a set of \( q \) queries expose?

**Claim**

# functions exposed by \( q \) queries

\[
\leq (q - 1) \cdot 2^{\sqrt{n}}
\]

# functions that a query pair \((x, y)\) exposes

\[
\leq \# \text{ coordinates on which } x \text{ and } y \text{ differ}
\]

\[
\leq 2^{\sqrt{n}}
\]

Only pairs of queries in the Green Band can be violated \( \Rightarrow \) disagreements \( \leq 2^{\sqrt{n}} \)
Hypercube 1-sided Error Lower Bound

- How many functions does a set of \( q \) queries expose?

Let \( Q \) be the set of queries made. The tester catches a violation ⇨

\( Q \) contains comparable \( x, y \) that differ in coordinate \( i \)

Draw an undirected graph \((Q, E)\) by connected comparable queries

Consider its spanning forest.

\[ f \]

\( 2\sqrt{n} \)

Claim

# functions exposed by \( q \) queries

\[ \leq (q - 1) \cdot 2\sqrt{n} \]

Sufficient to consider adjacent vertices in a minimum spanning forest on the query set

\[ x, y \text{ exist} \]

\( \updownarrow \)

there are adjacent vertices on the path from \( x \) to \( y \) that differ in coordinate \( i \)
Hypercube 1-sided Error Lower Bound

- How many functions does a set of $q$ queries expose?

**Claim**

# functions exposed by $q$ queries

$\leq (q - 1) \cdot 2\sqrt{n}$

**Claim**

Every deterministic test that makes a set $Q$ of $q$ queries (in the middle) succeeds with probability $O\left(\frac{q}{\sqrt{n}}\right)$ on our distribution.
Communication Complexity

A Method for Proving Lower Bounds

[Blais Brody Matulef 11]

Use known lower bounds for other models of computation

Partially based on slides by Eric Blais
(Randomized) Communication Complexity

**Goal:** minimize the number of bits exchanged.

- **Communication complexity of a protocol** is the maximum number of bits exchanged by the protocol.
- **Communication complexity of a function** $C$, denoted $R(C)$, is the communication complexity of the best protocol for computing $C$. 

**Input:** $x$

**Input:** $y$

---

**Compute** $C(x, y)$

**Shared random string**

1101000101110101101101010110110...
Example: Set Disjointness $\text{DISJ}_k$

**Input:** $S \subseteq [n], |S| = k$.

Compute $\text{DISJ}_k(S, T)$

$$= \begin{cases} 
\text{accept} & \text{if } S \cap T = \emptyset \\
\text{reject} & \text{otherwise}
\end{cases}$$

**Theorem** [Kalyanasundaram Schmitger 92, Razborov 92]

$$R(\text{DISJ}_k) \geq \Omega(k) \text{ for all } k \leq \frac{n}{2}.$$
A lower bound using CC method

Testing if a Boolean function is a k-parity
A Boolean function \( f: \{0,1\}^n \rightarrow \{0,1\} \) is *linear* (also called *parity*) if
\[
f(x_1, ..., x_n) = a_1 x_1 + \cdots + a_n x_n \quad \text{for some } a_1, ..., a_n \in \{0,1\}
\]

- Work in finite field \( \mathbb{F}_2 \)
  - Other accepted notation for \( \mathbb{F}_2 \): \( GF_2 \) and \( \mathbb{Z}_2 \)
  - Addition and multiplication is mod 2
  - \( x=(x_1, ..., x_n), y=(y_1, ..., y_n), \) that is, \( x, y \in \{0,1\}^n \)
  - \( x + y=(x_1 + y_1, ..., x_n + y_n) \)

**Example**

\[
\begin{array}{c}
001001 \\
011001 \\
\hline
010000
\end{array}
\]
A Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$ is linear (also called parity) if

$$f(x_1, \ldots, x_n) = a_1 x_1 + \cdots + a_n x_n$$

for some $a_1, \ldots, a_n \in \{0,1\}$

$$f(x_1, \ldots, x_n) = \sum_{i \in S} x_i$$

for some $S \subseteq [n]$.

Notation: $\chi_S(x) = \sum_{i \in S} x_i$. 

[n] is a shorthand for \{1, ... n\}.
**Testing if a Boolean function is Linear**

**Input:** Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$

**Question:**

Is the function linear or $\varepsilon$-far from linear

($\geq \varepsilon 2^n$ values need to be changed to make it linear)?

Later in the course:

Famous BLR (Blum Lubi Rubinfeld 90) test runs in $O\left(\frac{1}{\varepsilon}\right)$ time
A function $f : \{0,1\}^n \rightarrow \{0,1\}$ is a $k$-parity if

$$f(x) = \chi_S(x) = \sum_{i \in S} x_i$$

for some set $S \subseteq [n]$ of size $|S| = k$. 
Testing if a Boolean Function is a k-Parity

Input: Boolean function \( f : \{0,1\}^n \rightarrow \{0,1\} \) and an integer \( k \)

Question: Is the function a \( k \)-parity or \( \epsilon \)-far from a \( k \)-parity

\( (\geq \epsilon 2^n \) values need to be changed to make it a \( k \)-parity)?)

Time:

\( O(k \log k) \) [Chakraborty Garcia–Soriano Matsliah]

\( \Omega(\min(k, n - k)) \) [Blais Brody Matulef 11]

• Today: \( \Omega(k) \) for \( k \leq n/2 \)

Today’s bound implies \( \Omega(\min(k, n - k)) \)
**Important Fact About Linear Functions**

**Fact.** Two different linear functions disagree on half of the values.

- Consider functions $\chi_S$ and $\chi_T$ where $S \neq T$.
  - Let $i$ be an element on which $S$ and $T$ differ (w.l.o.g. $i \in S \setminus T$)
  - Pair up all $n$-bit strings: $(x, x^{(i)})$
    where $x^{(i)}$ is $x$ with the $i^{th}$ bit flipped.
  - For each such pair, $\chi_S(x) \neq \chi_S(x^{(i)})$
    but $\chi_T(x) = \chi_T(x^{(i)})$
  - So, $\chi_S$ and $\chi_T$ differ on exactly one of $x, x^{(i)}$.
  - Since all $x$'s are paired up,
    $\chi_S$ and $\chi_T$ differ on half of the values.

**Corollary.** A $k'$—parity function, where $k' \neq k$, is $\frac{1}{2}$-far from any $k$-parity.
Reduction from \( \text{DISJ}_{k/2} \) to Testing \( k \)-Parity

- Let \( T \) be the best tester for the \( k \)-parity property for \( \varepsilon = 1/2 \)
  - query complexity of \( T \) is \( q \) (testing \( k \)-parity).
- We will construct a communication protocol for \( \text{DISJ}_{k/2} \) that runs \( T \) and has communication complexity \( 2 \cdot q(\text{testing } k\text{-parity}) \).

Then \( 2 \cdot q(\text{testing } k\text{-parity}) \geq R(\text{DISJ}_{k/2}) \geq \Omega(k/2) \) for \( k \leq n/2 \)
\[ \Downarrow \]
\( q(\text{testing } k\text{-parity}) \geq \Omega(k) \) for \( k \leq n/2 \)

\[ \text{[Kalyanasundaram Schnitger 92]} \]

holders for CC of every protocol for \( \text{DISJ}_k \)
Reduction from $\text{DISJ}_{k/2}$ to Testing $k$-Parity

**Input:** $S \subseteq [n], |S| = k/2$.
Compute: $f = \chi_S$

Output $T$'s answer

* $T$ receives its random bits from the shared random string.
**Analysis of the Reduction**

**Queries:** Alice and Bob exchange 2 bits for every bit queried by $T$

**Correctness:**
- $h = f + g \pmod{2} = \chi_S + \chi_T \pmod{2} = \chi_{S \Delta T}$
- $|S \Delta T| = |S| + |T| - 2|S \cap T|$

- $|S \Delta T| = \begin{cases} k & \text{if } S \cap T = \emptyset \\ \leq k - 2 & \text{if } S \cap T \neq \emptyset \end{cases}$

$h$ is
- $k$-parity if $S \cap T = \emptyset$
- $k'$-parity where $k' \neq k$ if $S \cap T \neq \emptyset$

1/2-far from every $k$-parity

**Summary:** $q(\text{testing } k\text{-parity}) \geq \Omega(k)$ for $k \leq n/2$
Testing Lipschitz Property on Hypercube

Lower Bound
**Lipschitz Property of Functions** \( f : \{0,1\}^n \to \mathbb{R} \)

- A function \( f : \{0,1\}^n \to \mathbb{R} \) is **Lipschitz** if changing a bit of \( x \) changes \( f(x) \) by at most 1.

- Is \( f \) Lipschitz or \( \varepsilon \)-far from Lipschitz
  - Edge \( x - y \) is **violated** by \( f \) if \( |f(x) - f(y)| > 1 \).

**Time:**
- \( O(n/\varepsilon) \), logarithmic in the size of the input, \( 2^n \) 
  [Chakrabarty Seshadhri]
  
- \( \Omega(n) \) [Jha Raskhodnikova]
**Theorem**

Testing Lipschitz property of functions $f: \{0,1\}^n \rightarrow \{0,1,2\}$ requires $\Omega(n)$ queries.

Prove it.
Summary of Lower Bound Methods

• Yao’s Principle
  – testing membership in 1*, sortedness of a list and monotonicity of Boolean functions

• Reductions from communication complexity problems
  – testing if a Boolean function is a $k$-parity
Other Models of Sublinear Computation
Tolerant Property Tester [Rubinfeld Parnas Ron]

Randomized Algorithm

YES
Accept with probability $\geq \frac{2}{3}$

NO
Reject with probability $\geq \frac{2}{3}$

Tolerant Property Tester

YES
Accept with probability $\geq \frac{2}{3}$

$\epsilon_1$-close to YES
Don’t care

$\epsilon_2$-far from YES
Reject with probability $\geq \frac{2}{3}$
Sublinear-Time “Restoration” Models

Local Decoding
Input: A slightly corrupted codeword
Requirement: Recover individual bits of the closest codeword with a constant number of queries per recovered bit.

Program Checking
Input: A program $P$ computing $f$ correctly on most inputs.
Requirement: Self-correct program $P$: for a given input $x$, compute $f(x)$ by making a few calls to $P$.

Local Reconstruction
Input: Function $f$ nearly satisfying some property $P$
Requirement: Reconstruct function $f$ to ensure that the reconstructed function $g$ satisfies $P$, changing $f$ only when necessary. For each input $x$, compute $g(x)$ with a few queries to $f$.