Lecture 17:

Testing monotonicity of functions
Property Tester for Monotonicity:

Given List \(y_1, \ldots, y_n\)

Output sorted?

i.e. if \(y_1 \leq y_2 \leq \ldots \leq y_n\) output PASS (with prob \(\geq \frac{3}{4}\))

if \(y_1, \ldots, y_n\) far from sorted \(-\) need to delete/change \(\geq n\) entries

output FAIL (w/prob \(\geq \frac{3}{4}\))

Example

Sorted 1 2 4 5 7 11 14 19 20 21 23
Property Tester for Monotonicity:

**Given** List $y_1, \ldots, y_n$

**Output** sorted? 

i.e. if $y_1 \leq y_2 \leq \ldots \leq y_n$ output PASS (with prob $\frac{3}{4}$)

if $y_1, \ldots, y_n$ far from sorted (need to delete/change $\geq n$ entries) output FAIL (w/prob $\geq \frac{3}{4}$)

**Example**

<table>
<thead>
<tr>
<th>Sorted</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>11</th>
<th>14</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>14</td>
<td>19</td>
<td>20</td>
<td>39</td>
<td>23</td>
</tr>
</tbody>
</table>
Property Tester for Monotonicity:

**Given**
List \( y_1, \ldots, y_n \)

**Output**
sorted?

i.e., if \( y_1 \leq y_2 \leq \ldots \leq y_n \)
output \text{PASS} (with prob \( \frac{2}{3} \))

if \( y_1 \ldots y_n \) far from sorted (need to delete/change \( \geq n \) entries)
output \text{FAIL} (w/prob \( \geq \frac{2}{3} \))

**Example**

<table>
<thead>
<tr>
<th>Sorted</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>11</th>
<th>14</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>14</td>
<td>19</td>
<td>20</td>
<td>39</td>
<td>23</td>
</tr>
<tr>
<td>Far</td>
<td>45</td>
<td>39</td>
<td>23</td>
<td>1</td>
<td>38</td>
<td>4</td>
<td>5</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>2</td>
</tr>
</tbody>
</table>
Easy case: \( y_i \in \{0,1\}^3 \quad A \)

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1
\end{bmatrix}
\]

\( H_w \Rightarrow \text{poly}(\frac{1}{\varepsilon}) \text{ queries} \)

Comments:
- Definition of close:
  - Delete vs. change
  - \( \exists \) make sense over lists
  - Easier
  - \( \uparrow \) possible with same query complexity
- Why is this a fact?

\( y_1 \ldots y_n \Rightarrow f(1) \ldots f(n) \)  
\( \exists \) "delete" def of closeness

doesn't make sense
First Attempt: Given $y_1 \ldots y_n$

Proposed algorithm: "Neighbor test"

Pick random $i$, test $y_i \leq y_{i+1}$
First Attempt:

Proposed algorithm: "Neighbor test"

Pick random $i$, test $y_i \leq y_{i+1}$

Behavior:

passes good inputs ✓

fails "far" input in example:

<table>
<thead>
<tr>
<th>45</th>
<th>39</th>
<th>23</th>
<th>1</th>
<th>38</th>
<th>4</th>
<th>5</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
</tr>
</tbody>
</table>
First Attempt:

Proposed algorithm: "Neighbor test"

Pick random $i$, test $y_i \leq y_{i+1}$

Behavior:

- passes good inputs ✓
- fails "far" input in example:

<table>
<thead>
<tr>
<th>45</th>
<th>39</th>
<th>23</th>
<th>1</th>
<th>38</th>
<th>4</th>
<th>5</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
</tr>
</tbody>
</table>

bad input for test:

\[ \frac{3}{4} \text{ far from monotone} \]

Only 3 choices of $i$ fail test
Second Attempt:

Proposed algorithm: "random pair test"

pick random $i < j$, test $y_i < y_j$
Second Attempt:

Proposed algorithm: "random pair test"

pick random $i \leq j$ test $y_i < y_j$.

Behavior:

- Passes good inputs: $\checkmark$
- Fails a lot of pairs in:

<table>
<thead>
<tr>
<th></th>
<th>45</th>
<th>39</th>
<th>23</th>
<th>1</th>
<th>38</th>
<th>4</th>
<th>5</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>2</th>
</tr>
</thead>
</table>

Second Attempt:

Proposed algorithm: "random pair test"
pick random $i<j$, test $yi, yj$

Behavior:

passes good inputs $\checkmark$
Fails a lot of pairs in:

45 39 23 1 38 4 5 21 20 19 2

bad input for test: $\frac{n}{4}$ groups of 4 decreasing elements

4, 3, 2, 1, 8, 7, 6, 5, 12, 11, 10, 9, 16, 15, 14, 13, ...

- largest monotone subsequence. Keep $\leq 1$ elt from each group, size $\leq n/4$
  
  $\Rightarrow \frac{3}{4}$ - far from monotone

- to fail, must pick $i,j$ in same group $\Rightarrow$ prob $\leq \frac{1}{n}$
  
  if take $\sqrt{n}$ elts & check in order, leads to good test
Minor simplification:

Assume $y_1, ..., y_n$ distinct $\forall i \neq j, y_i \neq y_j$

Claim this is wlog

Why? old trick

$$X_1, ..., X_n \rightarrow (X_{i_1}, 1) (X_{i_2}, 2) \cdots (X_{i_k}, k) \cdots (X_{i_n}, n)$$

"virtually" append log bits describing \"i\" to each $X_i$
(at runtime)

break ties w/o changing order:

- $X_i \preceq X_{i+1}$ then $(X_{i_1}, i) \preceq (X_{i+1, i+1})$
- $X_i = X_{i+1}$ then $(X_{i_1}, i) \preceq (X_{i+1, i+1})$
A test: given $y_1, \ldots, y_n$

Repeat $O(\frac{1}{\epsilon})$ times

Pick $i \in [n]$

$z \leftarrow y_i$

do binary search on $y_i, \ldots, y_n$ for $z$

if see inconsistency fail and halt

\[ \text{e.g. left > right} \]

if end up at loc $j \neq i$ fail and halt

Pass

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorted</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>14</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>Close</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>14</td>
<td>19</td>
<td>20</td>
<td>39</td>
<td>23</td>
</tr>
<tr>
<td>Far</td>
<td>45</td>
<td>39</td>
<td>23</td>
<td>1</td>
<td>38</td>
<td>4</td>
<td>5</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>2</td>
</tr>
</tbody>
</table>
Why does it work?

- if \( y_1 < y_2 < \ldots < y_n \) always passes
  where we use distinctness

To show:

if need to delete \( \exists n \) \( y_i \)'s to make monotone then fail w/hp

equivalent: if likely to pass, then \( \varepsilon \)-close to monotone

\[
def \text{index} i \text{ is "good" if bin search for } z \leq x_i \text{ successful (no inconsistencies on way find } z \text{ in locn } i) \]

Restatement of test: Pick \( O(\varepsilon^3) \) \( x \)'s & test that they are all good

Repeat \( O(\varepsilon) \) times

Pick i.e., \( [n] \)

\( Z \leq y_i \)

do binary search on \( y_1 \ldots y_n \) for \( Z \)

if see inconsistency \( \text{FAIL halt} \)

e.g., \( \text{left > right} \)

if end up at \( \text{locn } j \neq i \text{ FAIL halt} \)
\begin{align*}
\text{def } i \text{ is "good" if bin search for } & z \leq x_i \text{ successful} \\
\text{Main Observation: set of good elements forms increasing subsequence} \\
\text{Proof: for } i < j \text{ both good, let } k \text{ be "least common ancestor" in binary search tree} \\
\text{when hit } x_k \\
\text{since } i, j \text{ good } \Rightarrow \text{ search for } x_i \text{ goes left } \Rightarrow x_i \leq x_k \leq x_j \\
\text{all pairs in right order } \Rightarrow \text{ whole set in right order} \\
\text{Need to show test passes } \Rightarrow \text{ set of good elts is large} \\
\text{set of bad elts is small} \\
\text{Claim: if } \geq \frac{3}{4} \text{ fraction of } i \text{ 's are bad, then test fails with prob } \geq \frac{3}{4} \\
\Rightarrow \text{ if test passes, can assume } \leq \frac{3}{4} \text{ fraction of } \text{ bad } i \text{ 's, good } i \text{ 's}
\end{align*}
def \( i \) is "good" if bin search for \( z < x_i \) successful

**Main Observation:** set of good elements forms increasing subsequence

**Proof:**

for \( i < j \) both good, let \( k \) be "least common ancestor" in binary search tree

when hit \( x_k \), search for \( x_i \) goes left \( \Rightarrow x_i < x_k < x_j \) since \( x_i, x_j \) good

\( x_j \) goes right \( \Rightarrow x_i < x_k < x_j \)

Need to show: test passes \( \Rightarrow \) set of good elements is large
lower bound: (idea)

assume $o(\log n)$ query monotonicity tester

exists $i < j$ s.t. very unlikely to query $x_i + x_j$ s.t.

$i, j \in \{0, \ldots, \log n\}$.

Show implies unlikely to be in different small but same medium group.

If pick $i, j$ in same small group PASS.

If pick $i, j$ in different medium group PASS.

Far from monotone.
Monotonicity over Posets:

\[ \text{def } f \text{ is "monotone over poset } P \text{" if } A x \preceq y, \text{ then } f(x) \preceq f(y) \]

Examples: (represent via dag)

- Bipartite posets

\[ \text{not allowed} \]
• hypercube

\[ \begin{aligned}
(1111111) & \quad \text{top level: all 1's} \\
(1111000) & \\
(1101000) & \\
(0101000) & \\
(0011000) & \\
(0001000) & \\
(0000100) & \quad \text{level } n-1; \quad n-1 \text{'s}
\end{aligned} \]

\[ \begin{aligned}
(1000000) & \\
(0100000) & \\
(0010000) & \\
(0001000) & \\
(0000100) & \quad \text{level } 1; \quad 1 \text{'s}
\end{aligned} \]

\[ \begin{aligned}
(0000000) & \\
(0000000) & \\
(0000000) & \\
(0000000) & \\
(0000000) & \quad \text{level 0; } 0 \text{'s}
\end{aligned} \]

\[ \begin{aligned}
& \quad \text{IMPORTANT} \\
& \quad \text{This poset describes monotone} \\
& \quad \text{Boolean functions.}
\end{aligned} \]
H.W.: Show testing monotonicity of arbitrary poset can be transformed into "equivalent" monotonicity testing problem on bipartite poset.

\[ \uparrow \]

monotonicity of functions on posets is "complete"
If can test monotonicity over posets, can also test:

1) Given 2CNF formula along with assignment \( A = \exists a_1 \ldots \exists a_n \) \( a_i \in \{ \top, \bot \} \)
   - Pass if \( \psi(A) = \top \)
   - Fail if \( \forall A' \) s.t. \( A \lor A' \varepsilon \text{-close} \), \( \psi(A') = \bot \)

2) Given \( G \) with \( U \subseteq V \)
   - Pass if \( U \) is vertex cover
   - Fail if \( \forall U' \) s.t. \( U \varepsilon \text{-close to } U' \), \( U' \) not V.C.

3) Given \( G \) with \( U \subseteq V \)
   - Pass if \( U \) is clique
   - Fail if \( \forall U' \) s.t. \( U \varepsilon \text{-close to } U' \), \( U' \) not clique
**Thm** For bipartite graphs, can test monotonicity in $O(\sqrt{nE})$.

**Pf.** h.w.

**Thm** bipartite test requires $n^\alpha$, for some small constant $\alpha$, queries nonadaptive.

Open: Improve to $\alpha = \frac{1}{2}$ adaptive case?

Grids:

$f: [n] \times [n] \rightarrow [m]$

Time to test

$O(\frac{1}{\varepsilon} \log n \log m)$

$d \geq \Theta(\varepsilon^{1/3})$

$f: [n]^d \rightarrow [m]$

$O(d^{1/2} \text{poly}(\varepsilon) \text{poly}(\log d))$

$\Omega(d^{1/4})$ even for adaptive