Lecture 17:

Testing monotonicity of functions
Property Tester for Monotonicity:

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

**Output** \( \text{sorted?} \)

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

if \( y_1 \ldots y_n \) \( \epsilon \) far from sorted \( \text{FAIL} \) (w/ prob \( \geq 3/4 \))

**Example**

<table>
<thead>
<tr>
<th>Sorted</th>
<th>1 2 4 5 7 11 14 19 20 21 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>1 4 2 5 7 11 14 19 20 39 23</td>
</tr>
<tr>
<td>Far</td>
<td>45 39 23 1 38 4 5 21 20 19 2</td>
</tr>
</tbody>
</table>
Easy case: \[ y_i \in \mathbb{E}, \forall i \]

\[ 0000000011111111 \]

\[ 0001000111011111 \]

\[ \exists H.W. \Rightarrow \text{poly}(\epsilon) \text{ queries} \]

Comments:

- Definition of close:
  - delete vs. change the \( y_i \)'s
  - turns out to be easier today
  - but "change" is also possible with same query complexity

- Why is this a fact?
  - \( y_1, \ldots, y_n \rightarrow f(i), \ldots, f(mn) \)
  - \( y_i \), \( y_n \)
  - \( \exists \) "delete" defn of closeness doesn't make sense
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:

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

\[
\rightarrow \quad 1, 2, 3, 4, 5, \ldots, \frac{n}{4}, 1, 2, 3, \ldots, \frac{n}{4}, 1, 2, 3, \ldots, \frac{n}{4}
\]

\[
F \quad P \quad P \quad P \quad P \quad F \quad P \quad \ldots \quad F \quad P \quad P \quad P \quad P \quad P \quad \ldots
\]

only 3 choices of $i$ fail test
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:

  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 keeps \( i \leq 1 \) in each group's size \( n/4 \)
- to fail, must pick \( i, j \) in same group \( \Rightarrow \) prob \( \leq \frac{1}{n} \)

  (if take \( \sqrt{n} \) samples \( \Rightarrow \) compare all pairs, prob is \( \Theta(1) \))
Minor simplification:

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

Claim this is wlog

**why?** old trick

$$X_1 \ldots X_n \rightarrow (x_{i,1}) \ (x_{2,2}), \ldots, (x_{i,i}), \ldots (x_{n,n})$$

virtually append $i$ to each $X_i$ (at runtime)

- note break ties w/o changing order!

if $X_i \preceq X_{i+1}$, then $(X_{i,i}) \preceq (X_{i+1,i+1})$
Repeat \( O(\frac{1}{2}) \) times

A test: given \( y \in [n] \)

\[ z \leftarrow y_i \]

do binary search on \( y_1..y_n \) for \( z \)

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

e.g. left \( \not> \) right

if end up at loc \( j \not= i \) \( \text{FAIL} \) and halt

Pass
Why does it work?

- if \( y_1 < y_2 < \ldots < y_n \) always passes

To show:

- if need to delete \( \exists \) \( n \) \( y_i \)'s to make monotone then fail whp

Equivalent, if likely to pass, then \( \epsilon \)-close to monotone

\[ \text{def } i \text{ is "good" if bin search for } z \leq x_i \text{ successful} \]

Restatement of test: Pick \( O(\epsilon) \) \( i \)'s + pass if all good


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
search for $x_j$, goes right $\Rightarrow x_i < x_k < x_j$

Need to show: test passes $\Rightarrow$ set of good elements is large
set of bad els small

**Claim**: if $\geq \varepsilon$ fraction of $i$'s are bad, test fails with prob $\geq 3/4$

$\Rightarrow$ if test passes, can assume $< \varepsilon$-fraction of $i$'s are bad
Monotonicity over Posets:

def \( f \) is "monotone over poset \( P \)" if \( A, x \leq y, \) then \( f(x) \leq f(y) \)

Examples: (represent via dag)

- bipartite posets

![Diagram of a poset with arrows indicating the order relation and a comment indicating that certain orders are not allowed.](image)
**hypercube**

- (1111...1)
- (0111...1)
- (0011...1)
- (0001...1)

**top level:** all 1's

**level n-1:** n-1 1's

**level 1:** 1 1's

(100000) (010000)...

(00000) (00001)

**IMPORTANT**

This poset describes monotone Boolean funs.
H.W. : Show testing monotonicity of arbitrary poset can be transformed into “equivalent” monotonicity testing problem on bipartite poset.

In a sense in which bipartite poset testing is “complete” for monotonicity testing over posets.
If can test monotonicity over posets, can also test:

1) Given 2CNF formula along with assignment $A = \exists a_1 \ldots a_n \in T, F$
   
   • Pass if $\varphi(A) = T$
   • Fail if $\forall A' \ s.t. A \pm A' \ \varepsilon$-close, $\varphi(A') = F$

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

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

Proof. h.w.

Thm. Test requires $n^2$ for constant queries nonadaptive.

Past h.w. $\Rightarrow \Theta(\log n)$ queries adaptive.

Open: can we improve this to $\alpha = 1/2$? adaptive case?
Grids:

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

Time to test

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

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

\[ O(\frac{d^{1/2}}{\text{poly}(\varepsilon)} \text{ poly}(\log d)) \]

\[ f: 2^d \rightarrow \mathbb{S}_0(1, 3) \]

\[ \Omega(d^{1/4}) \text{ even for adaptive} \]