# Lecture 17:

## Testing monotonicity of functions

# Property Tester for Monotonicity:

<u>Given</u> List $y_1 \cdots y_n$

<u>Output</u> sorted?

i.e. if $y_1 \le y_2 \le \dots \le y_n$ output PASS (with prob $\ge 3/4$)

if $y_1 \cdots y_n$ $\varepsilon$-far from sorted ← need to delete/change $\varepsilon n$ entries

output FAIL (w/ prob $\ge 3/4$)

<u>example</u>

Sorted 1 2 4 5 7 11 14 19 20 21 23

# Property Tester for Monotonicity:

<u>Given</u> List $y_1 \cdots y_n$

<u>Output</u> sorted?

i.e. if $y_1 \le y_2 \le \ldots \le y_n$ output PASS (with prob $\ge 3/4$)

if $y_1 \cdots y_n$ $\varepsilon$-far from sorted (need to delete/change $\varepsilon n$ entries)

output FAIL (w/ prob $\ge 3/4$)

<u>example</u>

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Sorted | 1 | 2 | 4 | 5 | 7 | 11 | 14 | 19 | 20 | 21 | 23 |
| Close | 1 | 4 | 2 | 5 | 7 | 11 | 14 | 19 | 20 | 39 | 23 |

# Property Tester for Monotonicity:

**Given** List $y_1 \cdots y_n$

**Output** sorted?

i.e. if $y_1 \le y_2 \le \cdots \le y_n$ output PASS (with prob $\ge 3/4$)

if $y_1 \cdots y_n$ $\varepsilon$-far from sorted (need to delete/change $\varepsilon n$ entries)

output FAIL (w/ prob $\ge 3/4$)

**example**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Sorted | 1 | 2 | 4 | 5 | 7 | 11 | 14 | 19 | 20 | 21 | 23 |
| Close | 1 | 4 | 2 | 5 | 7 | 11 | 14 | 19 | 20 | 39 | 23 |
| far | 45 | 39 | 23 | 1 | 38 | 4 | 5 | 21 | 20 | 19 | 2 |

Easy case: $y_i \in \{0,1\} \quad \forall i$

$$\begin{array}{l} 0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,1\,1\,1\,1\,1 \end{array}$$

HW $\Rightarrow$ poly$(\frac{1}{\varepsilon})$ queries

Comments:

- definition of close:

  delete vs. change } make sense over lists

  delete: easier

  change: possible with same query complexity

- why is this a fctn?

  $y_1 \cdots y_n \to f(1) \ldots f(n)$ } "delete" def of closeness doesn't make sense

First Attempt: Given $y_1 \ldots y_n$

Proposed algorithm: "neighbor test"

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

## First Attempt:

Proposed algorithm: "neighbor test"

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

## Behavior:

passes good inputs ✓

fails "far" input in example:

| 45 | 39 | 23 | 1 | 38 | 4 | 5 | 21 | 20 | 19 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| Fail | Fail | Fail | Pass | Fail | Pass | Pass | Fail | Fail | Fail | |

# First Attempt:

Proposed algorithm: "neighbor test"

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

## Behavior:

passes good inputs ✓

fails "far" input in example:

| 45 | 39 | 23 | 1 | 38 | 4 | 5 | 21 | 20 | 19 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| Fail | Fail | Fail | Pass | Fail | Pass | Pass | Fail | Fail | Fail | |

bad input for test:

$\approx \frac{3}{4}$-far from monotone ⟶ $1, 2, 3, 4, 5, \ldots \frac{n}{4}, 1, 2, 3 \ldots \frac{n}{4}, 1, 2, 3, \ldots, \frac{n}{4}, 1, 2, 3 \ldots \frac{n}{4}$

P P P P P F P P P … P F P P P … P F P P P …

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 $i<j$, test $y_i < y_j$

Behavior:

passes good inputs ✓

Fails a lot of pairs in:

45 39 23 1 38 4 5 21 20 19 2

## Second Attempt:

Proposed algorithm: "random pair test"

pick random $i<j$, test $y_i < y_j$

## Behavior:

passes good inputs ✓

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

$\underbrace{4, 3, 2, 1}, \underbrace{8, 7, 6, 5}, \underbrace{12, 11, 10, 9}, \underbrace{16, 15, 14, 13}, \ldots$

- 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 $y_1 \cdots y_n$ distinct $\forall i \neq j,\ y_i \neq y_j$

Claim this is wlog

why? old trick

$$x_1 \cdots x_n \rightarrow (x_1, 1)\,(x_2, 2)\ \cdots\ (x_i, i)\ \cdots\ (x_n, n)$$

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

break ties w/o changing order:

$x_i \leq x_{i+1}$ then $\underbrace{(x_i, i) \leq (x_{i+1}, i+1)}^{\leq}$

$x_i = x_{i+1}$ then $\underbrace{(x_i, i) \leq (x_{i+1}, i+1)}_{\leq}$

A test: given $y_1 \cdots y_n$

runtime:

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

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

Pick $i \in_r [n]$

$z \leftarrow y_i$

do binary search on $y_1 \cdots y_n$ for $z$

if see inconsistency FAIL + halt

e.g. left > right

if end up at loc $j \neq i$ FAIL + halt

Pass

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Sorted | 1 | 2 | 4 | 5 | 7 | 11 | 14 | 19 | 20 | 21 | 23 |
| Close | 1 | 4 | 2 | 5 | 7 | 11 | 14 | 19 | 20 | 39 | 23 |
| far | 45 | 39 | 23 | 1 | 38 | 4 | 5 | 21 | 20 | 19 | 2 |

## Why does it work?

- if $y_1 < y_2 < \dots < y_n$ always passes

  where we use distinctness

- To show:

  if need to delete $> \varepsilon n$ $y_i$'s to make monotone then fail whp

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

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

Pick $i \in_r [n]$

$z \leftarrow y_i$

do binary search on $y_1 \cdots y_n$ for $z$

if see inconsistency FAIL + halt

e.g. left > right

if end up at loc $j \neq i$ FAIL + halt

def $i$ is "good" if bin search for $z \leftarrow x_i$ successful (no inconsistencies on way find $z$ in locn $i$)

index

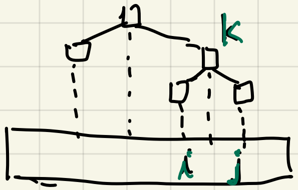
restatement of test: Pick $O(\frac{1}{\varepsilon})$ $i$'s + test that they are all good

**def** $i$ is "good" if bin search for $z \leftarrow x_i$ 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 $x_k$

since $i,j$ good: search for $x_i$ goes left, " " $x_j$ " right $\Big\} \Rightarrow x_i \le x_k < x_j$

$\Rightarrow x_i < x_j$

all pairs in right order $\Rightarrow$ whole set in right order $\blacksquare$

Need to show test passes $\Rightarrow$ set of good elts is large

$\Updownarrow$

set of bad elts is small

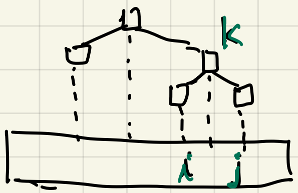
**Claim** if $\geq \varepsilon$ fraction of $i$'s are bad, then test fails with prob $\geq 3/4$

$\Rightarrow$ if test passes, can assume $\begin{cases} < \varepsilon \text{ fraction of bad } i\text{'s} \\ \geq 1-\varepsilon \text{ " " good } i\text{'s} \end{cases}$

def $i$ is "good" if bin search for $z \leftarrow x_i$ successful

Main Observation: set of good elements forms increasing sub sequence

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
$x_j$ goes right

since $x_i, x_j$ good

$\Rightarrow x_i < x_k < x_j$ ∎

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

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

$\exists$ $i \ldots i+c$ s.t. very unlikely to query $x_i$ & $x_j$ s.t. $2^i \le j-1 \le 2^{i+c}$

$i \in [0 \ldots \log n]$


show implies unlikely to be in different small but same medium group

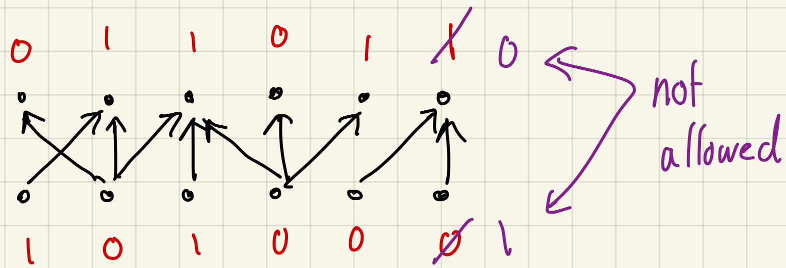
- if pick $i, j$ in same small group
  PASS
- if pick $i$ & $j$ in different medium group
  PASS
- far from monotone

# Monotonicity over Posets:

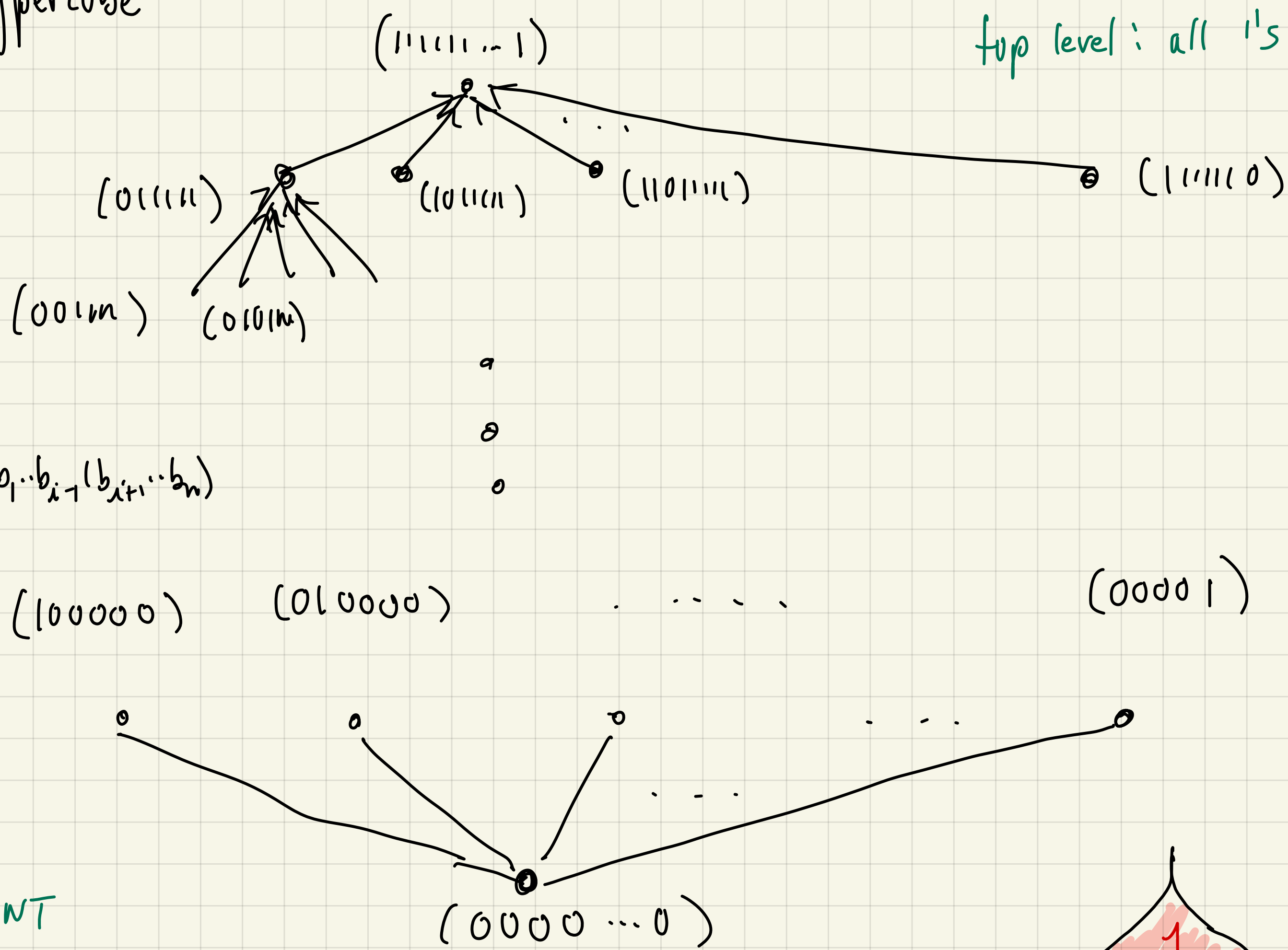
<u>def</u> $f$ is "monotone over poset $P$" if $\forall\, x \propto y$, then $f(x) \leq f(y)$

<u>examples</u>: (represent via dags)

- bipartite posets



- hypercube




top level: all 1's

level $n-1$: $n-1$ 1's

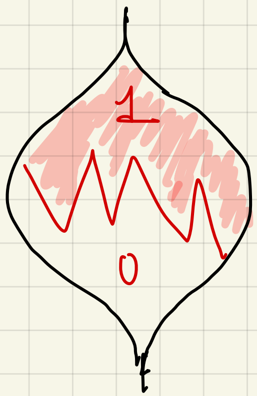
$X \to y$

$$(b_1 \dots b_{i-1} 0 b_{i+1} \dots b_n) \to (b_1 \dots b_{i-1} 1 b_{i+1} \dots b_n)$$

level 1: 1 1's

IMPORTANT

This poset describes monotone Boolean fctns.



H.W.: Show testing monotonicity of arbitrary poset can be transformed into "equivalent" monotonicity testing problem on bipartite poset.

$\Uparrow$

a sense in which testing monotonicity of functions on bipartite posets is "complete"

If can test monotonicity over posets, can also test:

1) Given 2CNF formula along with assignment $A = \{a_1 \cdots a_n\}$ $a_i \in \{T, F\}$

- PASS if $\varphi(A) = T$
- FAIL if $\forall A'$ s.t. $A$ & $A'$ $\varepsilon$-close, $\varphi(A') = F$

$\Leftrightarrow$ whp

2) Given $G$ with $U \subseteq V$

- PASS if $U$ is vertex cover
- FAIL if $\forall U'$ s.t. $U$ $\underbrace{\varepsilon\text{-close}}_{\#\text{ nodes in } U' \Delta U \,\le\, \varepsilon \cdot n}$ 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 $O(\sqrt{n/\varepsilon})$

Pf. h.w.

Thm bipartite test requires $n^{\alpha}$, for small const $\alpha$, queries nonadaptive

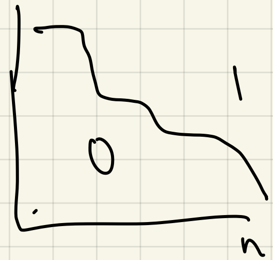
Open improve to $\alpha = 1/2$

adaptive case?

## Grids:

## Time to test

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




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

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

$O\left(\frac{d}{\varepsilon} \log n \log m\right)$

$f : 2^d \to \{0,1\}$

$O\left(\frac{d^{1/2}}{\mathrm{poly}(\varepsilon)} \mathrm{poly}(\log d)\right)$

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