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

if $y_1 \cdots y_n$ ε-far from sorted (need to delete/change εn 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
far	45	39	23	1	38	4	5	21	20	19	2

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

0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1

0 0 0 1 0 0 0 1 1 1 0 1 1 1 1 1

H.W. $\Rightarrow$ poly$(1/\varepsilon)$ 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 fctn?

  $y_1 \dots y_n \rightarrow f(1) \dots f(n)$ with $f(1) = y_1$, …, $f(n) = y_n$

  "delete" defn of closeness doesn't make sense

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:

$\frac{3}{4}$-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 $y_i < y_j$

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 keeps ≤ 1 in each group; size $n/4$
- to fail, must pick i,j in same group $\Rightarrow$ prob $\leq \frac{1}{n}$
  (if take $\sqrt{n}$ samples & compare all pairs, prob is $\Theta(1)$)

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

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

virtually append i to each x_i
(at runtime)

note break ties w/o changing order:
if $x_i \leq x_{i+1}$ then $(x_i, i) \leq (x_{i+1}, i+1)$

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

A test:

given $y_1 \cdots y_n$

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

Why does it work?

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

- To show:

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

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

equivalent: if likely to pass, then ε-close to monotone

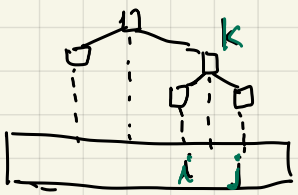
<u>def</u> i is "good" if bin search for $z \leftarrow x_i$ successful

restatement of test: Pick $O(\frac{1}{\varepsilon})$ i's + pass if all good 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,

since x_i, x_j good

$\left.\begin{array}{l}\text{search for } x_i \text{ goes left} \\ x_j \text{ goes right}\end{array}\right\} \Rightarrow x_i < x_k < x_j$ ∎

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

$\Updownarrow$

set of bad elts small

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

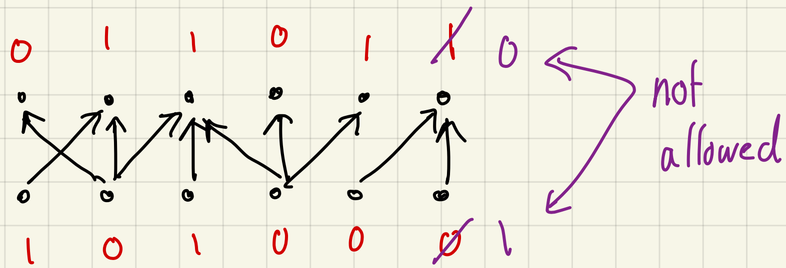
$\Rightarrow$ if test passes can assume $< \varepsilon$-fraction of i's are bad

Monotonicity over Posets:

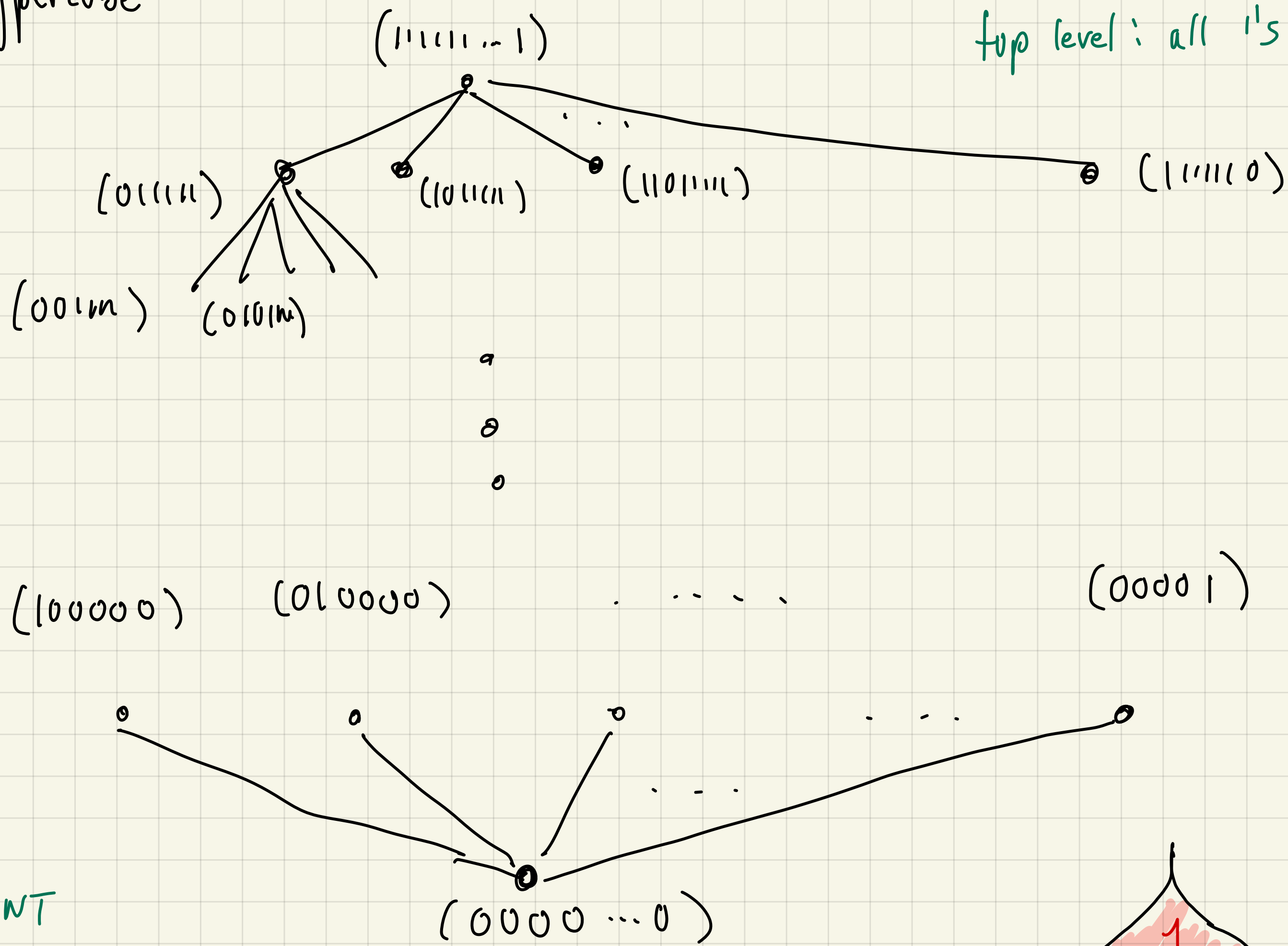
def f is "monotone over poset P" if $\forall\, x \prec y$, then $f(x) \le f(y)$

examples: (represent via dags)

- bipartite posets



- hypercube



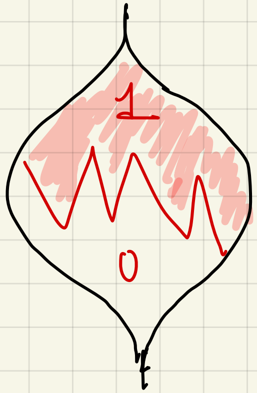
top level: all 1's

level n-1: n-1 1's

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.

⇑

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 $A = \{a_1 \dots a_n\}$ $a_i \in \{T, F\}$

- PASS if $\varphi(A) = T$
- FAIL if $\forall A'$ s.t. A & A' ε-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 $\underbrace{\varepsilon\text{-close}}_{\#\text{ nodes in } U' \Delta U \leq \varepsilon \cdot n}$ 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 ε-close to U', U' not clique

Thm For bipartite graphs
can test monotonicity in $O(\sqrt{n/\varepsilon})$

Pf. h.w.

Thm test requires n^{α}, for small const α, queries nonadaptive

past h.w. $\Longrightarrow$ $\Omega(\log n)$ queries adaptive

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

Grids:

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

Time to test

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

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

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

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

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

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