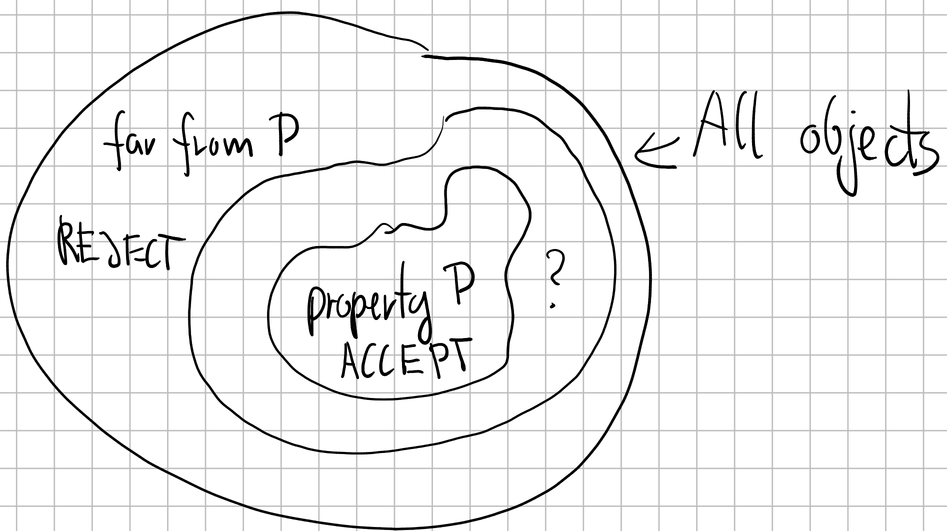
Today: Monotonicity testing

General area of study: property testing

Distinguish objects that have a property from those far from the property



Examples we have seen:

- uniformity testing
- does a graph have a perfect matching?
  (= all vertices can be matched)
  - if a graph has a perfect matching, accept w.p. $9/10$
  - if εn edges have to be added (i.e. matching size $\leq \frac{n}{2} - \varepsilon n$), reject w.p. $9/10$

Solution: approximate maximum matching size up to $\frac{\varepsilon n}{4}$ with $d^{O(1/\varepsilon^2)}$ queries (w.p. $9/10$)

- if it's $\geq \frac{n}{2} - \frac{\varepsilon n}{2}$, accept
- if it's $< \frac{n}{2} - \frac{\varepsilon n}{2}$, reject

Today: monotonicity testing

Input: Sequence $s_1, s_2, \ldots, s_n$

Parameter $\varepsilon \in (0,1)$

Goal:

- if $s_1 \leq s_2 \leq s_3 \leq \ldots \leq s_n$, accept w.p. $9/10$
- if εn elements of the sequence have to be deleted to make it monotone, reject w.p. $9/10$

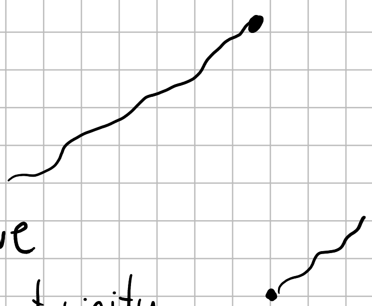
ε-far from monotone

$\boxed{\text{How many queries to } s_i\text{'s needed?}}$

Possible approaches:

① Query consecutive pairs to see if in order

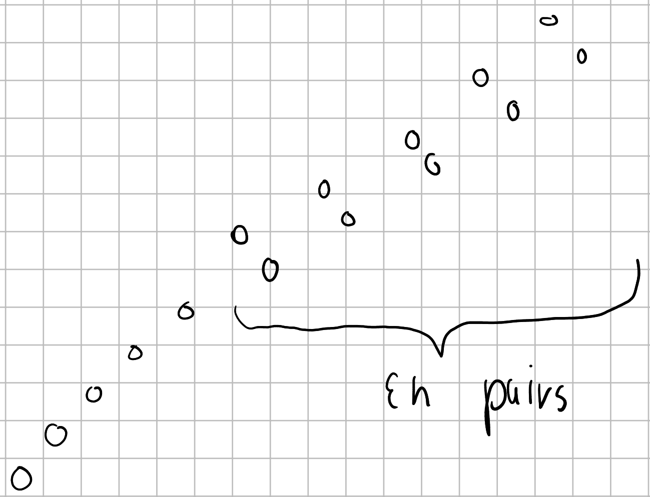
Obstacle:



only one consecutive pair could violate monotonicity

$\Omega(n)$ queries to find it

(2) Sample k random elements of the sequence & see if the subsequence is monotone



Via the birthday paradox, need $\Omega\left(\frac{1}{\varepsilon} \cdot \sqrt{\varepsilon n}\right)$ samples to sample both elements of one of the pairs

<u>This approach works with $O\left(\sqrt{\frac{n}{\varepsilon}}\right)$ samples</u>

Why?

Fact: if $\geq \varepsilon n$ elements have to be deleted to make the sequence monotone, there are $\geq \frac{\varepsilon n}{2}$ disjoint out of order pairs

Proof:

- greedily remove pairs out of order
- until $\frac{\varepsilon n}{2}$ pairs ($= \varepsilon n$ elements) removed, sequence not monotone, so there is one more out of order pair ∎

Now $O\left(\frac{1}{\varepsilon}\right) \cdot \underbrace{O\left(\sqrt{n\varepsilon}\right)}$ samples suffice to detect one of them w.p. $9/10$

Overhead to hit elements in our pairs

Birthday paradox: if we sample from elements involved in our $\frac{\varepsilon n}{2}$ pairs

---

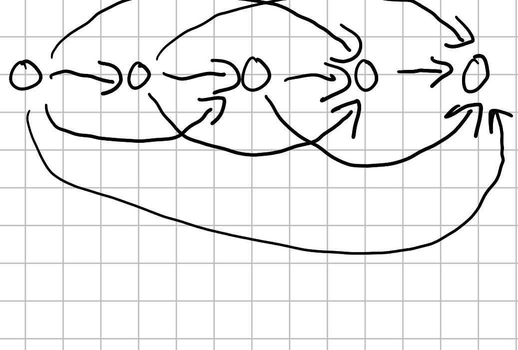
Our goal: $O\left(\frac{1}{\varepsilon}\log n\right)$ queries

Transitive closure: All pairs that should be in order explicitly connected

$x \to y$ means $x \leq y$

$$s_1 \to s_2 \to s_3 \to s_4 \to s_5$$

$\Downarrow$ transitive closure



$\binom{n}{2}$ edges

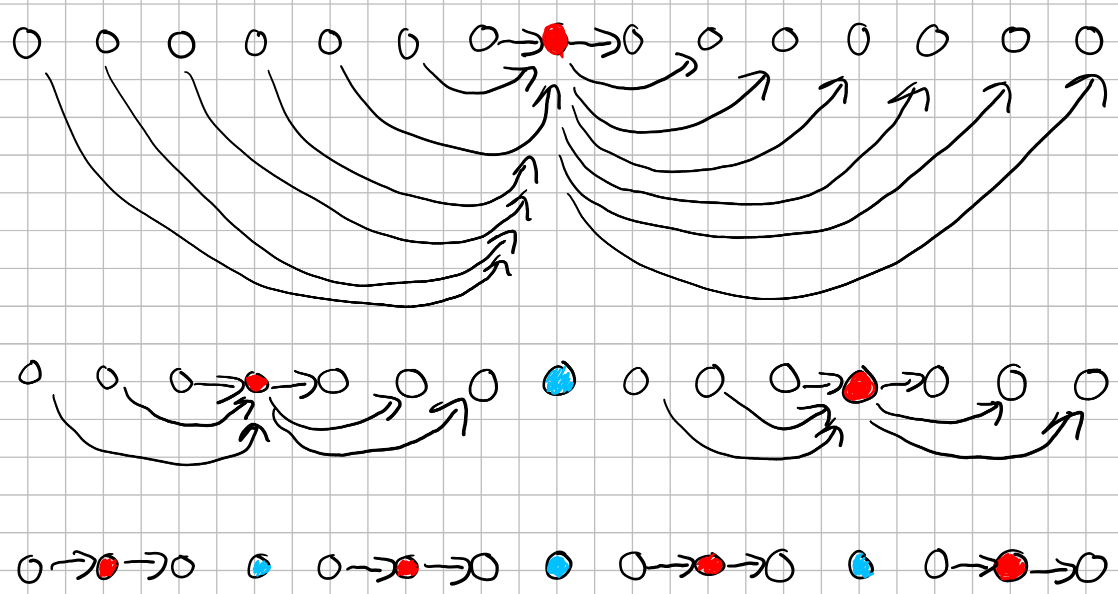
(and $\geq \frac{\varepsilon n}{2}$ disjoint edges violating the ordering if far from monotone)

2-transitive-closure spanner ("spanner")

= subgraph of transitive closure s.t. for each edge $x \to y$ in transitive closure there is a directed path from x to y of length at most 2

Construction of spanner with $O(n \log n)$ edges

Idea: - connect everyone to the middle point
- recurse on the left and right sides



- $O(\log n)$ levels of recursion
- $\leq n$ edges on each level
- $O(n \log n)$ edges overall

Why valid spanner?

- Every pair separated by a middle point at some point
- So there is a directed path of length 1 or 2 from one to the other on some level

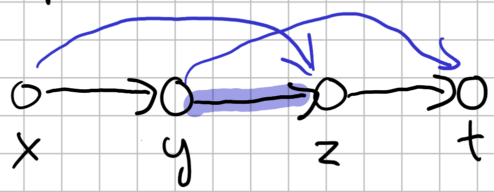
---

Consider sequence ε-far from monotone and any spanner

- if pair x, y violates the ordering, it's connected by path of length ≤ 2 in the spanner and at least one of the edges on the path violates the ordering as well
- $\geq \frac{\varepsilon n}{2}$ disjoint pairs that violate the ordering

$\Rightarrow \geq \frac{\varepsilon n}{4}$ edges in the spanner violate the ordering as well

Why: at most two out of order disjoint pairs can claim the same out of order edge in the spanner



- $x \to z$ & $y \to t$ are vertex disjoint out of order pairs
- this could happen with just $y \to z$ in the spanner being out of order but no three such pairs can rely on just one edge in the spanner

---

Algorithm: sample $O(\frac{1}{\varepsilon}\log n)$ edges in the spanner and see if in the right order

- monotone sequence always accepted ("one-sided error")
- ε-far sequence:
  - $\geq \frac{\varepsilon n}{4}$ edges in the spanner violate the ordering
  - $O(n \log n)$ edges in the spanner
  - fraction of violating edges $\Omega\left(\frac{\varepsilon n/4}{n \log n}\right) = \Omega\left(\frac{\varepsilon}{\log n}\right)$
  - $O\left(\frac{\log n}{\varepsilon}\right)$ samples suffice to detect such an edge with probability $9/10$