

Lecture 25

PROPERTY TESTING

Object: $\text{string} / \text{function} / \text{list}$
of length n

Goal: Test if object satisfies
property P ??

Sorted? bipartite??

Runtime $\frac{\# \text{Queries}}{\text{Total runtime}}$
 $<< n$
 $\Theta(\log n)$

Guarantee: Object satisfies P
 \Rightarrow test accepts $\omega \cdot P^{0.99}$

Object is FAR from satisfying
from $P \Rightarrow$ test rejects $\delta 0.99$

SORTEDNESS

Example:

Input: $a_1, \dots, a_n \in \mathbb{Z}$

To Test:

Is it sorted??

"closeto"

\equiv flipped

Example:

$a_1 \leq a_2 - - a_i \leq a_{i+1}$

$\langle a_1, \dots, a_n \rangle$

Defn: An increasing subsequence

$a_1 \backslash \backslash \backslash \backslash \backslash a_n$

$a_{i_1} < a_{i_2} < a_{i_3} < \dots < a_{i_T}$

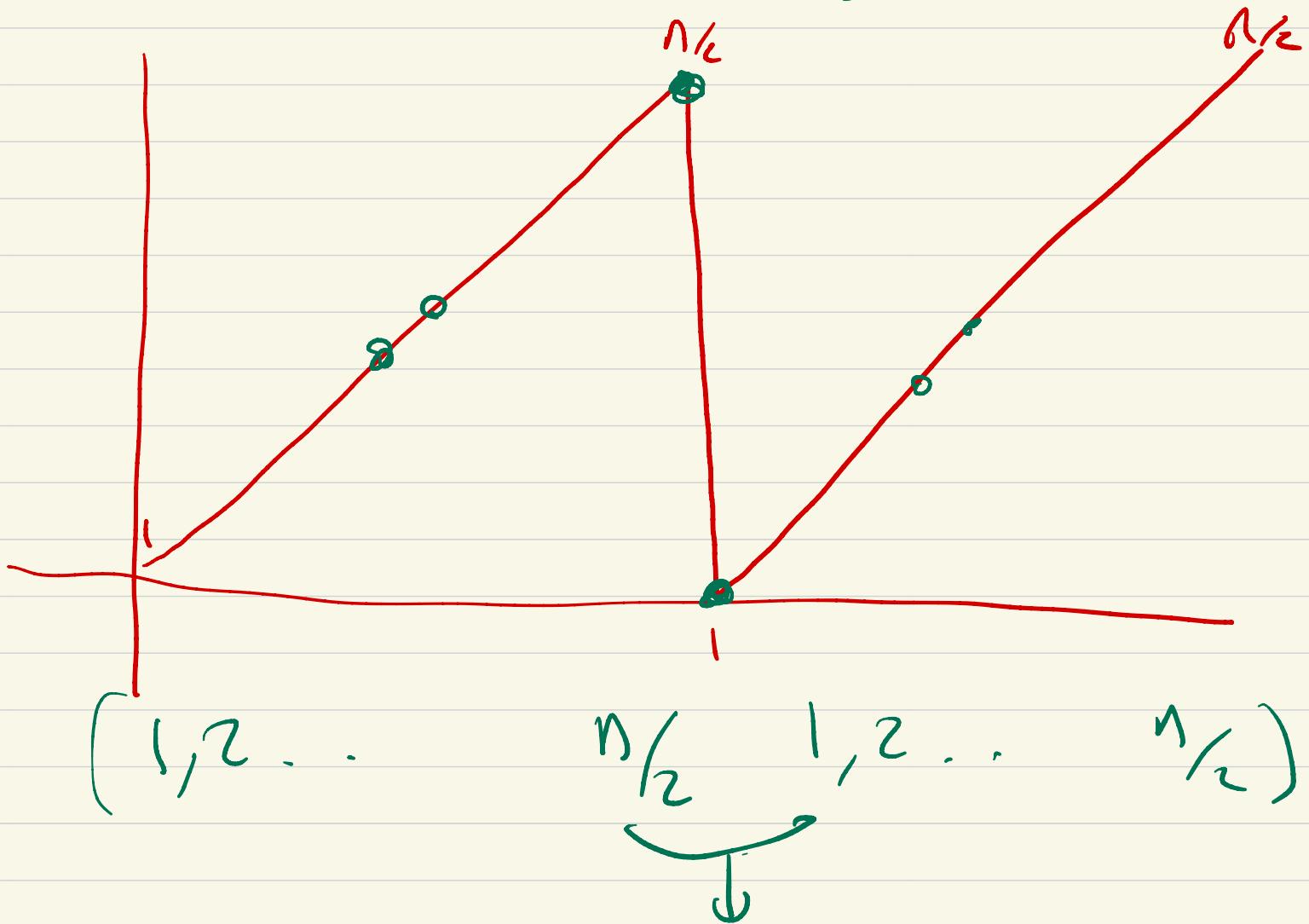
Defn: A sequence a_1, \dots, a_n is

k -away from sorted if longest
increasing subsequence is length $n-k$

Test:

Pick random i

check if $a_i \leq a_{i+1}$?



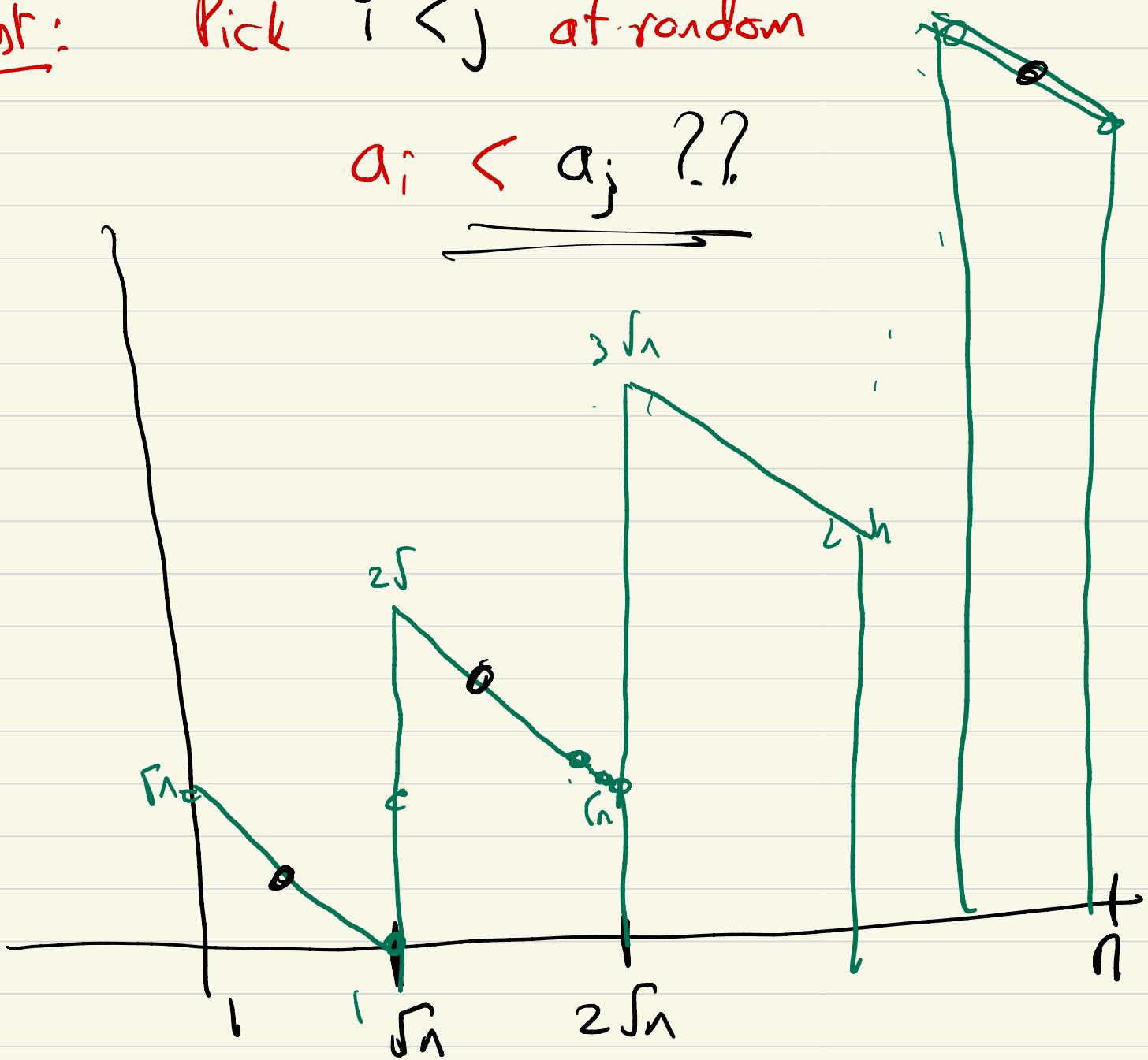
Sequence is $\frac{1}{2}$ far from sorted

but test rejects w.p. $O(1/n)$

Test:

Pick $i < j$ at random

$a_i < a_j ??$



Alg:

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

- Pick $i \in \{1, \dots, n\}$ at random
- Run Binary search to find A_i^*
- reject if you find a violation during binary search.

Queries: $O\left(\frac{\log n}{\epsilon}\right)$

Proof: Alg accepts w.p 0.99

\Rightarrow List is ϵn -close to sorted

Def: A_i is a "good" element if
Binary Search (A_i) no violations
are detected.

Claim 1: Alg accepts w.p 0.99

$\Rightarrow \# \text{Good Elements} \geq (1-\varepsilon) n$

Proof:

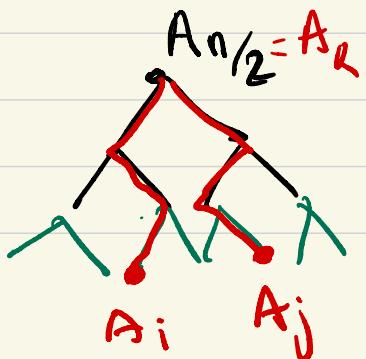
$$|\text{Good Elements}| < (1-\varepsilon)n$$

$$\Pr[\text{Test } \overset{\text{accept}}{\cancel{\text{reject}}} \text{ }] \leq (1-\varepsilon)^{\frac{100}{\varepsilon}}$$

$$\leq (1-\varepsilon)^{\frac{100}{\varepsilon}} < e^{-100}$$

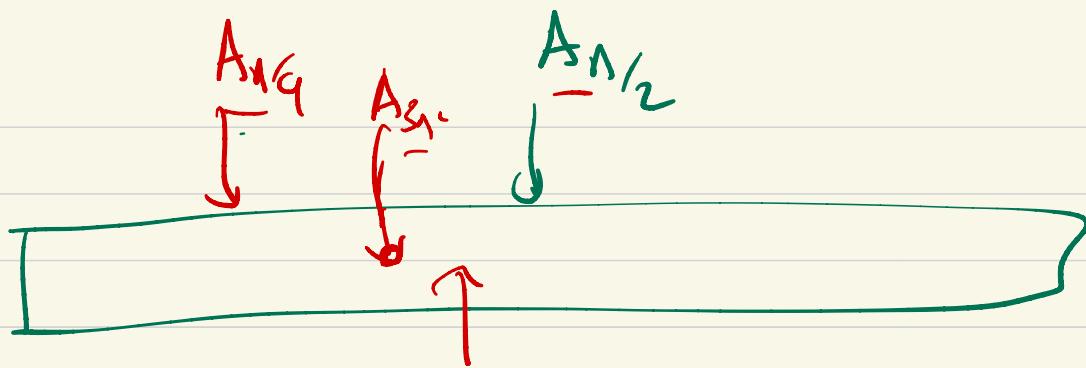
Claim 2: Good Elements form an increasing sequence

Proof: $A_i \& A_j$ are good elements



Let A_K be the LCA

$A_i < A_K \& A_K < A_j$



$$\Rightarrow \underline{\underline{A_i < A_j}} \quad !!$$

Maximal Matching

Vertex Cover / Const
time alg that
gives 2-approximation

Input: Graph $G = (V, E)$

maximum degree in $G \leq d$.

Goal: Estimate

size of a Maximal Matching

Alg out puts t , $\exists M_{\text{maximal}} \mid M \mid \approx t \pm \epsilon$

Maximal matching: $M \subseteq E$, M

is a matching, $\forall e \in E \setminus M$

$M \cup \{e\}$ is NOT
a matching.

ALG1

1) Pick a random permutation of E

$$\pi: E \rightarrow \{1, \dots, |E|\}$$

2) For $i = 1$ to $|E|$

Add i^{th} edge e_i to M
if it's allowed
else throw it out.

ALG1 constructs a maximal matching

ALG2: Pick $\frac{1}{\varepsilon^2}$ random edges
(Wishful Thinking)

$$e_1, \dots, e_l \quad l = \frac{1}{\varepsilon^2}$$

~ Estimate $\frac{(|\{e_1, \dots, e_l\} \cap M|)}{l}$

- Output $|E|$

ORACLE!

Given edge e

Test

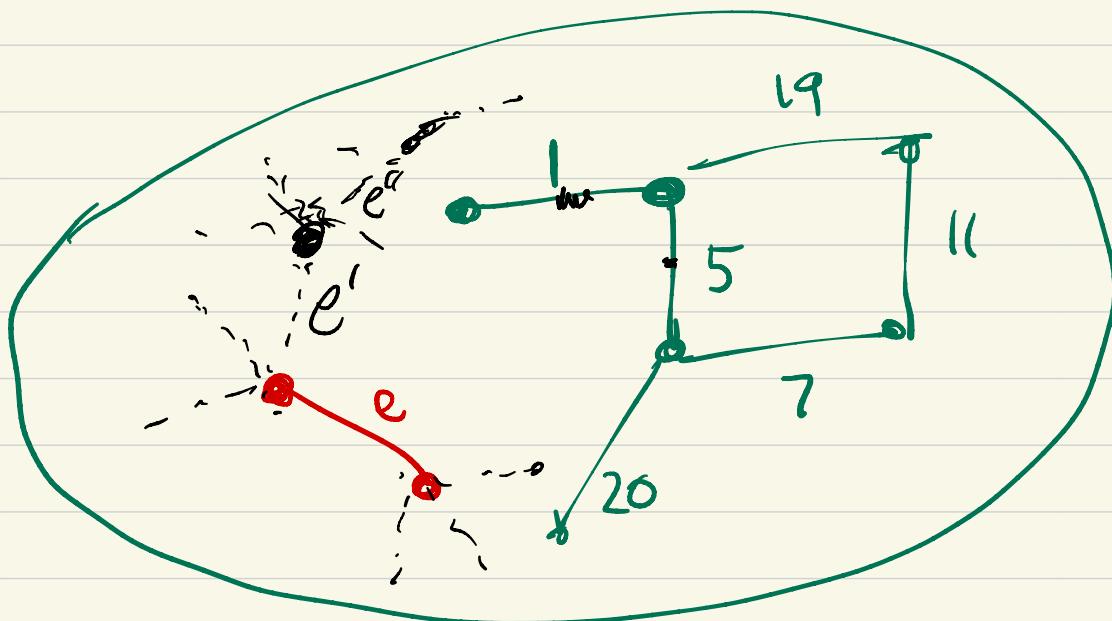
$e \in M$

??

output by ALG1

2^d
queries

(in constant time depending
on $d \& E$)



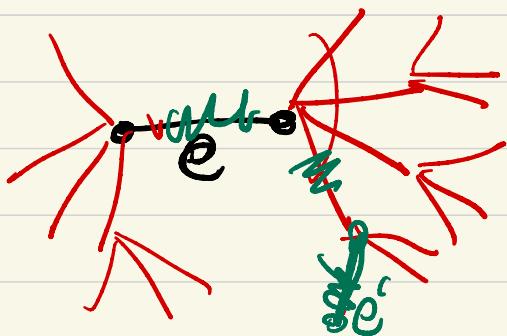
e added \Rightarrow ^{only if} no neighboring edge
 e' is added

If $\pi(e) < \text{all } \pi(e')$
for every neighboring
edge e'

$\Rightarrow e \in M$

If $\pi(e') < \pi(e)$

Test if e' is added
or not.



On $\text{test}(e)$:

if we query e'

- 1) π is decreasing
on path from $e \rightsquigarrow e'$

Path from $e \sim e'$ in length K .

$$\Pr \left[\begin{array}{l} \pi \text{ is decreasing} \\ \text{on path} \end{array} \right] < \frac{1}{K!}$$

$$\# \text{ of edges at } \underset{\text{distance } K}{\approx} 2d^K$$

$$E \left[\# \text{ edges queried} \right] \leq \sum_{k=1}^{\infty} \frac{2d^k}{k!}$$

$$\leq 2e^d$$

Total Runtime = $2e^d / \varepsilon$

Szemerédi Regularity Lemma

