Lecture 2 Jan 16, 2025

- Sortedness Testing (general case)

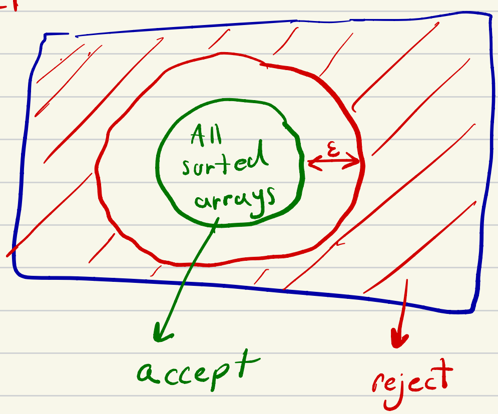
Testing sortedness (general case)

Recall def

Give an array A,

Design an algorithm $\mathcal{A}$ s.t. with prob. $1-\delta$

- if A is sorted, $\mathcal{A}$ outputs accept
- if A is ε-far from being sorted, $\mathcal{A}$ outputs reject



what does ε-far mean here?

distance between two array of size n:

$$\mathrm{dist}(A, A') = \frac{\text{entries we need to chang to change } A \text{ to } A'}{n}$$

$$P = \{\text{all sorted arrays}\}$$

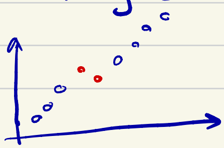
$$\mathrm{dist}(A, P) = \min_{A' \in P} \mathrm{dist}(A, A')$$

ε-far from sortedness = We need to change $\geq \varepsilon . n$ entries in A to get a sorted array.

---

why randomly throwing darts in the dark won't work in general case?

unable to detect local changes



New algorithm

Binary search base algorithm.

Sorted $\Rightarrow$ binary search works.

" $\overset{?}{\Leftarrow}$ "

Assumption: WLOG, entries of A are distinct.

Try s times

pick a random $i \in [n]$

$\ell \leftarrow$ Binary search$(A, A[i])$

if $(\ell \neq i)$

return reject

return accept.

* If A is sorted $\Rightarrow$

all calls to binary search work correctly

$\Rightarrow$ the algorithm returns accept w. prob 1.

* If A is ε-far from sorted $\overset{?}{\Rightarrow}$

$p :=$ the probability of binary search fails when we are ε-far

what we need:

$$\Pr[\text{outputting accept} \mid \varepsilon\text{-far}] = (1-p)^s \leq \delta$$

by setting $s = \dfrac{\log(1/\delta)}{p}$

if $p < \varepsilon \Rightarrow (1-\varepsilon).n$ many entries are nice

$\Downarrow$ Lemma 1

$(1-\varepsilon).n$ many entries are sorted

$\Downarrow$

A is not ε-far from being sorted

$$\Rightarrow \quad p \geq \varepsilon \quad \Rightarrow \quad s = \frac{\log 1/\delta}{\varepsilon}$$

would be enough.

Binary_ search (array A, value x, indices h, t)

if $(t < h)$

return h

$m \leftarrow \left\lfloor \frac{h+t}{2} \right\rfloor$

if $(A[m] = x)$

return m

if $(A[m] > x)$

return binary_search $(A, x, h, m-1)$

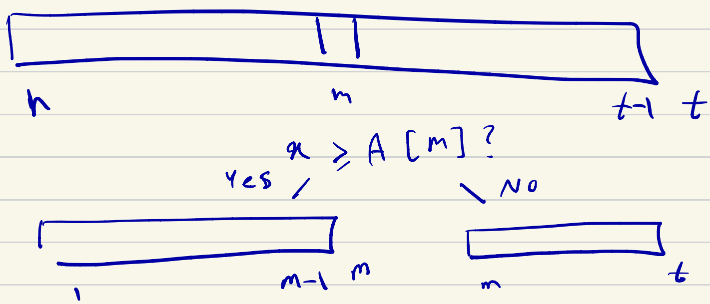
if $(A[m] < x)$

return binary_search $(A, x, m+1, t)$

Binary search on a sorted A returns the smallest i such that

$A[i] \geq x$

Binary search $(A, x, 1, n+1)$ could return $n+1$.



$A[m]$ is called pivot

We say i is nice if the binary search on $x = A[i]$ returns i

Lemma 1 Suppose we have two nice indices i and $j \in [n]$. If $i < j$ then $A[i] < A[j]$

Proof

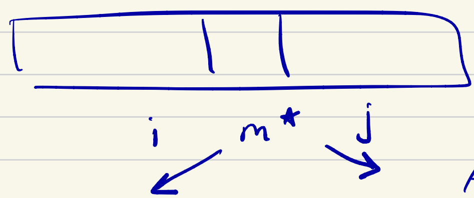
pivots of i

$$m_1^{(i)}, \; m_2^{(i)}, \; \dots, \; m_{k_i}^{(i)} = i$$

pivots of j.

$$m_1^{(j)}, \; m_2^{(j)}, \; \dots, \; m_{k_j}^{(j)} = j$$

$m^* =$ last mutual pivot



$A[i] < A[m^*] < A[j]$