CS3000: Algorithms & Data
Jonathan Ullman

Lecture 5:

- Divide-and-Conquer: more examples

Sep 21, 2018

Divide-and-Conquer:
Binary Search

Binary Search

Is 28 in this list? If so, where.

2	3	8	$\boxed{11}$	15	17	28	42	A

$l = 1 \quad r = 8 \quad m = 4$

15	$\boxed{17}$	28	42

$l = 5 \quad r = 8, \quad m = 6$

$\boxed{28}$	42

$l = 7 \quad r = 8 \quad m = 7$

yes $A[7] = 28$

Binary Search

```
Search(A,t):
  // A[1:n] sorted in ascending order
  Return BS(A,1,n,t)

BS(A,ℓ,r,t):
  If(ℓ > r): return FALSE

  m ← ℓ + ⌊(r−ℓ)/2⌋

  If(A[m] = t): Return m
  ElseIf(A[m] > t): Return BS(A,ℓ,m-1,t)
  Else: Return BS(A,m+1,r,t)
```

$T(n)$ = time to search array of size n

$T(n) = T\left(\frac{n}{2}\right) + C \qquad T(1) = C$

Proof of Correctness of Binary Search

Clm: $\forall n \in \mathbb{N}$ $\forall\ \ell, r$ s.t. $r - \ell \leq n$, $\forall A$, $\forall t$

$$BS(A, \ell, r, t) = \begin{cases} i \text{ s.t. } A[i] = t \\ \perp \text{ if } t \notin A \end{cases}$$

H(n)

Inductive Hyp

H(0) ...

Base Case: H(1) the algorithm is correct

Inductive Step: Assume H(n) is true

Suppose that we get $BS(A, \ell, r, t)$ and $r - \ell \leq n+1$

$m \leftarrow \ell + \lfloor \frac{r-\ell}{2} \rfloor$

$\boxed{\text{case 1:}}$ $A[m] = t$, $BS(A, \ell, r, t) = m$ ✓

$\boxed{\text{case 2:}}$ $A[m] < t$, $BS(A, \ell, r, t) =$
$BS(A, \underbrace{m+1, r}_{r-m-1 \leq n}, t)$

By the inductive hyp, we either find t or return $\perp$

- If we get $\perp$ then $t \notin A[m+1, r]$
  $\Rightarrow t \notin A[\ell, r]$

$\boxed{\text{case 3:}}$ $A[m] > t$, same as case 2

Ask the Audience

- What is the running time of binary search?
  - What is the recurrence? $T(n) = T(\frac{n}{2}) + c \quad T(1) = c$
  - What is the solution to the recurrence?

Maste Thm: $T(n) = a \cdot T(\frac{n}{b}) + c n^d$

$a = 1$

$b = 2$

$d = 0$

$\frac{a}{b^d} = \frac{1}{2^0} = 1$

$T(n) = \Theta(n^d \log n)$

$= \Theta(\log n)$

Binary Search Wrapup

- Search a sorted array in time $O(\log n)$
- Divide-and-conquer approach
  - Find the middle of the list, recursively search half the list
  - **Key Fact:** eliminate half the list each time
- Prove correctness via induction
- Analyze running time via recurrence
  - $T(n) = T(n/2) + C$

Selection (Median)

Selection

- Given an array of numbers $A[1, \dots, n]$, how quickly can I find the:
  - Smallest number? $O(n)$
  - Second smallest? $O(n)$
  - k-th smallest? $O(nk)$
  - median? $\lfloor \frac{n}{2} \rfloor^{nd}$ smallest $O(n^2)$

first	11	3	3	3	3	3	2	2
second	$\emptyset$	11	11	11	11	8	3	$\boxed{3}$

11	3	42	28	17	8	2	15	$\boldsymbol{A}$

Selection

$O(nk)$ $O(n \log n)$

$O(n \log k)$

- **Fact:** can select the k-th smallest in $O(n \log n)$ time
  - Sort the list and look up $A[k]$

11	3	42	28	17	8	2	15	$\boldsymbol{A}$

sort

2	3	8	$\boxed{11}$	15	17	28	42

- **Today:** select the k-th smallest in $O(n)$ time

Median Algorithm: Take I

- If $k < r$ then k^{th} smallest is in $A[1:r-1]$
- If $k > r$ then k^{th} smallest $\in A[r+1:n]$

$p = 17 = A[1]$

17	3	42	11	28	8	2	15	13	***A***

$r = 7$

11	3	5	13	2	8	17	28	42

Partitioning means partially sorting into $\boxed{<p \mid p \mid >p}$

```
Select(A[1:n],k):
  If(n = 1): return A[1]

  Choose a pivot p = A[1]
  Partition around the pivot, let p = A[r]

  If(k = r): return A[r]
  ElseIf(k < r): return Select(A[1:r-1],k)
  ElseIf(k > r): return Select(A[r+1:n],k-r)
```

$O(n)$ time

- $\Theta(n^2)$
- $\Theta(n \log n)$
- $\Theta(n)$

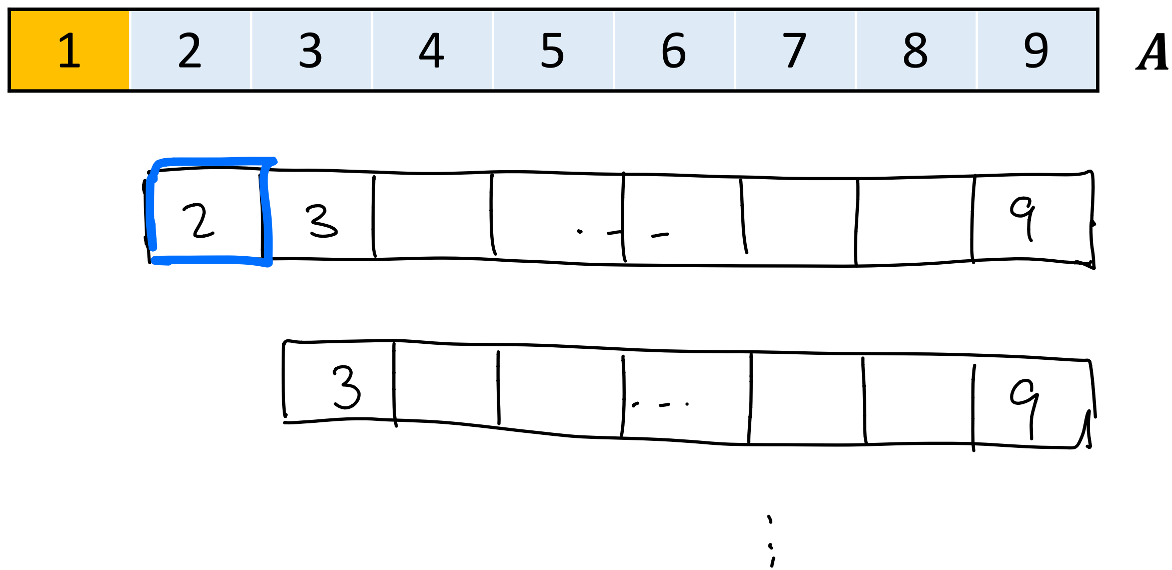
Why n ?

$$T(n) = T\left(\cancel{\tfrac{n}{2}}\right) + cn \quad T\left(\tfrac{3n}{4}\right)$$

$$T(n) = \Theta(n)$$

Median Algorithm: Take I

$k = n$



n times

$$T(n) = T(n-1) + Cn$$

$$T(n) = C \cdot \sum_{i=1}^{n} i = \Theta(n^2)$$

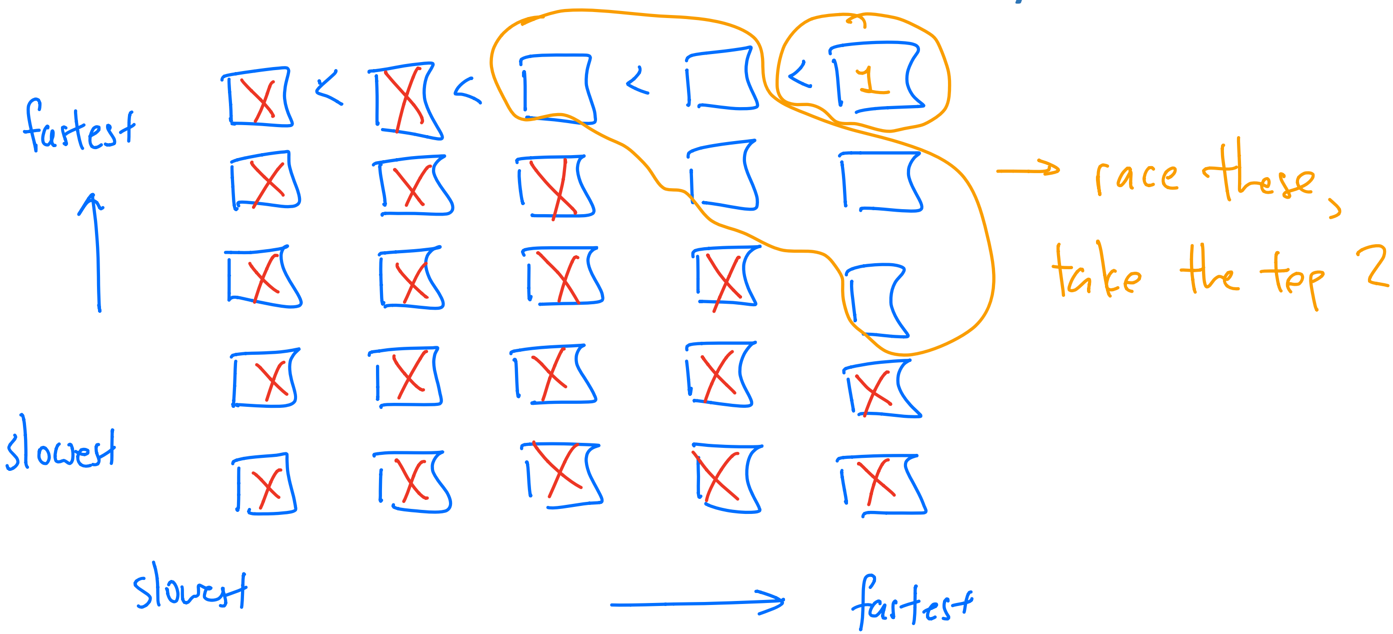
Median Algorithm: Take II

- **Problem:** we need to find a good pivot element

- Best possible pivot is the median
- Enough to find an element in the middle 75% of the list

Warmup

- You have 25 horses and want to find the 3 fastest
- You have a racetrack where you can race 5 at a time
  - In: $\{1, 5, 6, 18, 22\}$ Out: $(6 \succ 5 \succ 18 \succ 22 \succ 1)$
- **Problem:** find the 3 fastest with only seven races

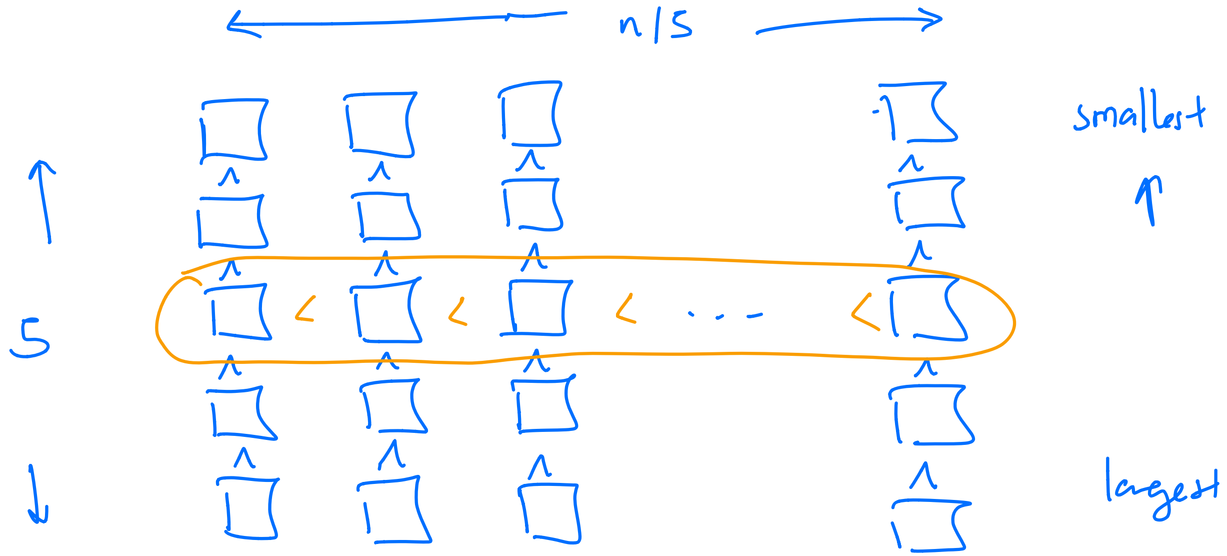


Median of Medians

```
MOM(A[1:n]):
  Let m ← ⌈n/5⌉
  For i = 1,…,m:
    M[i] ← median{A[5i-4],…,A[5i]}
    p ← Select(M[1:m],⌊m/2⌋)
```

Find medians in $\frac{n}{5} \times O(1) = O(n)$

$T\left(\frac{n}{5}\right)$

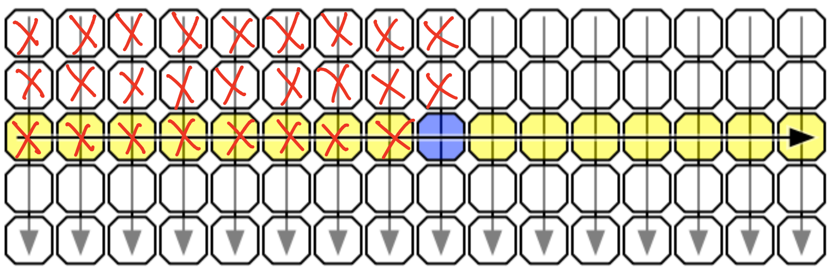


Median of Medians

- **Claim:** For every $\boldsymbol{A}$ here are at least $\frac{3n}{10}$ items that are smaller than $\mathbf{MOM}(\boldsymbol{A})$

$\left(\frac{n}{10}\right.$ columns are left of the MOM $\left.\right) \times \left(\right.$3 elts per column that are smaller than MOM$\left.\right)$

$= \frac{3n}{10}$ elts are smaller than MOM



Visualizing the median of medians

- Also $\frac{3n}{10}$ elts are larger than the MOM

Median Algorithm: Take II

17	3	42	11	28	8	2	15	13	*A*

11	3	5	13	2	8	17	28	42

40%

```
MOMSelect(A[1:n],k):
  If(n ≤ 25): return median{A}

  Let p = MOM(A)          T(n/5) + cn
  Partition around the pivot, let p = A[r]     cn

  If(k = r): return A[r]
  ElseIf(k < r): return MOMSelect(A[1:r-1],k)     T(7n/10)
  ElseIf(k > r): return MOMSelect(A[r+1:n],k-r)
```

Running Time Analysis

- $T(n)$ = time to find k^{th} smallest out of n

- $T(n) = \underbrace{T\left(\frac{n}{5}\right)}_{\text{MOM}} + \underbrace{T\left(\frac{7n}{10}\right)}_{\text{recurse}} + \underbrace{cn}_{\text{partitioning + MOM}}$

  $T(1) = c$

Recursion Tree

$$T(n) = T\left(\frac{7n}{10}\right) + T\left(\frac{2n}{10}\right) + Cn$$
$$T(1) = C$$

Level	Size	Work
0	n	Cn
1	$2n/10$, $7n/10$	$C\left(\frac{2n}{10} + \frac{7n}{10}\right) = \frac{9}{10} Cn$
2	$\frac{4n}{100}$, $\frac{14n}{100}$, $\frac{14n}{100}$, $\frac{49n}{100}$	$\frac{81}{100} Cn = \left(\frac{9}{10}\right)^2 Cn$

$$T(n) = Cn \times \sum_{i=0}^{\log_{\frac{10}{7}}(n)} \left(\frac{9}{10}\right)^i$$

$$\leq Cn \times \sum_{i=0}^{\infty} \left(\frac{9}{10}\right)^i = \Theta(n)$$

Ask the Audience

- If we change MOM so that it uses $\frac{n}{3}$ blocks of size 3, how many items can we eliminate?

- What is the new running time of the algorithm?

Selection Wrapup

- Find the k-th largest element in $O(n)$ time
  - Selection is strictly easier than sorting!
- Divide-and-conquer approach
  - Find a pivot element that splits the list roughly in half
  - **Key Fact:** median-of-medians-of-five is a good pivot
- Can sort in $O(n \log n)$ time using same technique
  - Algorithm is called **`Quicksort`**
- Analyze running time via recurrence
  - Master Theorem does not apply
- **Fun Fact:** a random pivot is also a good pivot!