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	11	15	17	28	42	A

Is this 28? No.

Is it bigger or smaller? Smaller.

15	17	28	42

28	42

Found it!

Return 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   1

  m ← ℓ + ⌊(r−ℓ)/2⌋          1

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

$\Big\}\ T(\frac{n}{2})$

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

$T(n) = T(\frac{n}{2}) + c \qquad T(1) = c$

Correctness of Binary Search

Clm: Binary Search is correct

$\forall n \in \mathbb{N}$ $\forall$ arrays A of size n $\forall t$

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

bot

$\forall$ arrays A $\forall n \in \mathbb{N}$ $\forall \ell, r$ s.t. $r-\ell \le n$ $\forall t$

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

$H(n)$ Inductive Hypothesis

Base Case: $H(1)$ is correct

Inductive Step: Assume $H(n)$ let ℓ, r s.t. $r-\ell = n+1$

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

case 1: $A[m]=t$, $BS(A,\ell,r,t) = m$ which is correct

case 2: $A[m] < t$, we output $\boxed{BS(A, m+1, r, t)}$

correct by $H(n)$

- if we get i s.t. $A[i]=t$
- if we get $\perp$, then $t \notin A[m+1:r]$

we also know $t \notin A[\ell:m]$ because A is sorted and $A[m]<t$

$\Rightarrow t \notin A[\ell:r]$ so we are correct

case 3: $A[m] > t$ is symmetric $\square$

Ask the Audience

- What is the running time of binary search?
  - What is the recurrence?
  - What is the solution to the recurrence?

$$T(n) = T\left(\frac{n}{2}\right) + c$$

$$T(1) = c$$

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + Cn^d$$

$a = 1$

$b = 2$

$d = 0$

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

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

$$= \Theta(\log n)$$

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 $A[1, \ldots, n]$, how quickly can I find the:
  - Smallest number? $O(n)$ time
  - Second smallest? $O(n)$
  - k-th smallest? $O(nk)$
  - median? $\lfloor \frac{n}{2} \rfloor$-th smallest number $\Theta(n^2)$

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

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

$O(n\log k)$

- **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	$\boldsymbol{A}$

$\Downarrow$ sort

2	3	8	11	15	17	28	42

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

Median Algorithm: Take I

17	3	42	11	28	8	2	15	13	A

r=7   A[r]=p

11	3	5	13	2	8	17	28	42

partitioning: splitting into | <p | p | >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

- $O(n^2)$

$$T(n) = T\left(\frac{n}{2}\right) + Cn$$

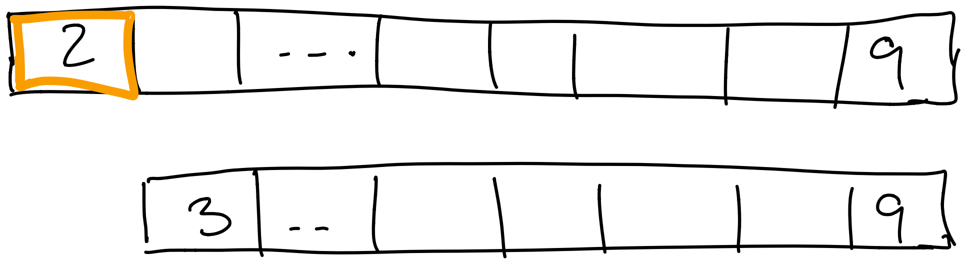
$$T(n) = O(n)$$

$$T(n) = T\left(\frac{3n}{4}\right) + Cn$$

$$= O(n)$$

Median Algorithm: Take I

1	2	3	4	5	6	7	8	9	A



time to find k^{th} smallest

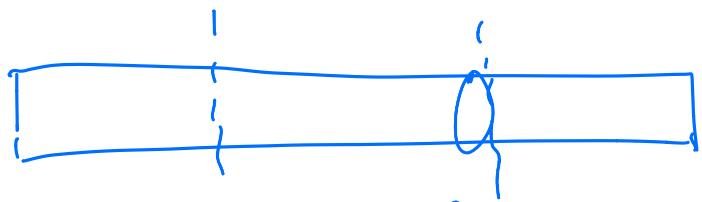
$$\sum_{i=0}^{k} n-i \approx nk - k^2/2 = \Theta(nk)$$

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

Median Algorithm: Take II

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

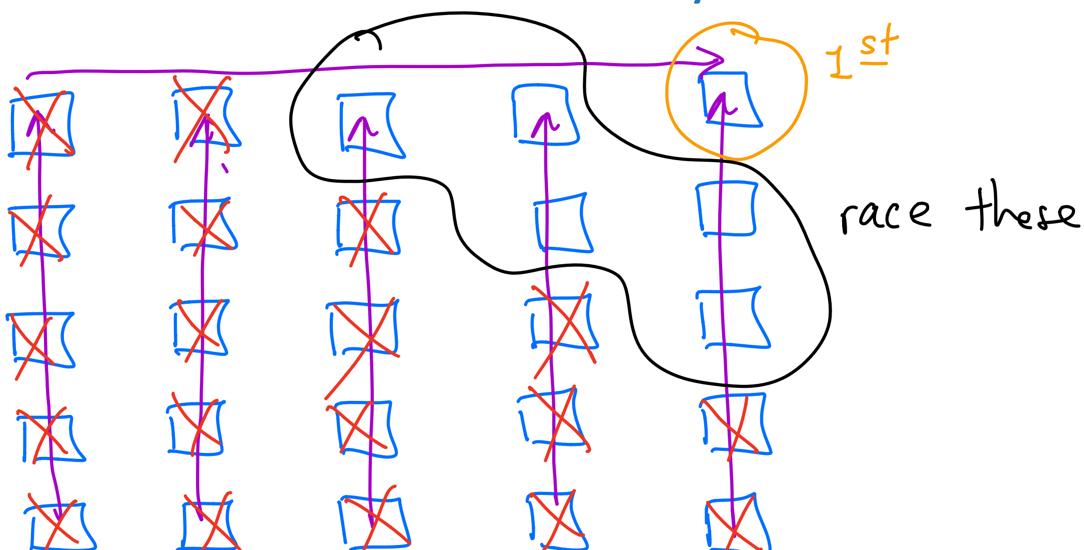
- Perfect pivot elt is the median
- Enough to have an elt in the middle half of the list



... as long as we can find it quickly

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

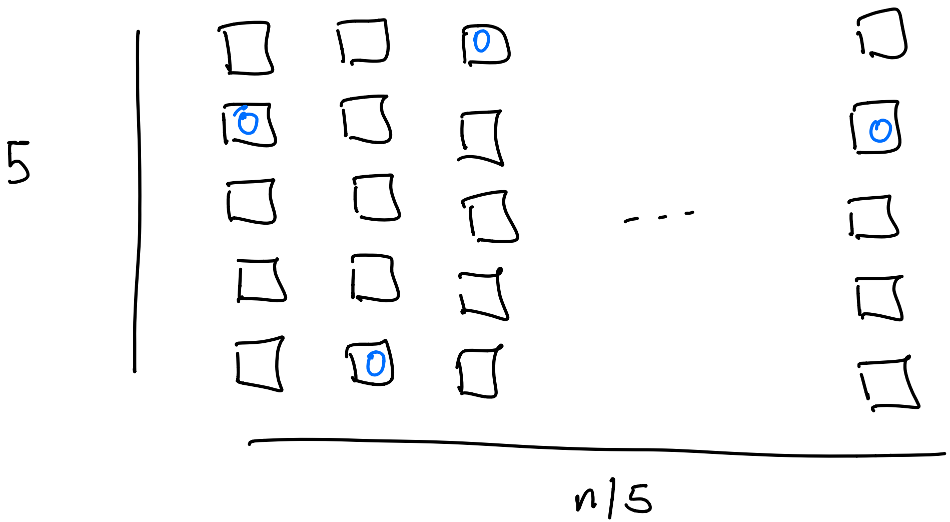
MOM(A[1:n]):

  Let $m \leftarrow \lceil n/5 \rceil$

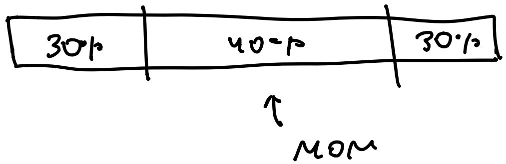
  For i = 1,…,m:

    M[i] ← median{A[5i-4],…,A[5i]} $\frac{n}{5} \times O(1) = O(n)$

    p ← Select(M[1:m], $\lfloor m/2 \rfloor$**)** $T(\frac{n}{5})$

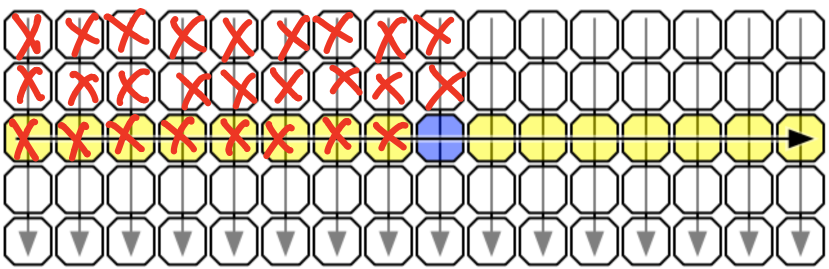


Median of Medians



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

$$\left(3 \text{ rows of elts}\right) \times \left(\frac{n}{10} \text{ columns}\right) = \frac{3n}{10}$$



Visualizing the median of medians

- Also $\frac{3n}{10}$ are large 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

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

  Let p = MOM(A)
  Partition around the pivot, let p = A[r]

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

Running Time Analysis

$T(n)$ = time to select with n elts

$$T(n) = T\left(\frac{7n}{10}\right) + [\text{time to find the pivot}] + Cn$$

$$= T\left(\frac{7n}{10}\right) + \left[T\left(\frac{n}{5}\right) + Cn\right] + Cn$$

$$= T\left(\frac{7n}{10}\right) + T\left(\frac{n}{5}\right) + Cn$$

Recursion Tree

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

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

$$Cn \cdot \sum_{i=0}^{\ell} \left(\frac{9}{10}\right)^i \le Cn \cdot \sum_{i=0}^{\infty} \left(\frac{9}{10}\right)^i = 10Cn$$
$$= O(n)$$

Ask the Audience

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

$2 \times \left(\frac{n}{6}\right) = \frac{n}{3}$ elts

- What is the new running time of the algorithm?

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$$

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

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!