CS3000: Algorithms & Data
Jonathan Ullman

Lecture 4:

- Divide and Conquer: Selection, Binary Search

Jan 15, 2020

The "Master Theorem"

- Recipe for recurrences of the form:
  - $T(n) = \boldsymbol{a} \cdot T(n/\boldsymbol{b}) + Cn^{\boldsymbol{d}}$
- Three cases:
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) > 1 : T(n) = \Theta\left(n^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right)$ (Karatsuba $T(n) = 3T(\frac{n}{2}) + n$)
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) = 1 : T(n) = \Theta\left(n^{\boldsymbol{d}} \log n\right)$ (Mergesort $T(n) = 2T(\frac{n}{2}) + n$; Binary Search $T(n) = T(\frac{n}{2}) + 1$)
  - $\left(\frac{\boldsymbol{a}}{\boldsymbol{b}^{\boldsymbol{d}}}\right) < 1 : T(n) = \Theta\left(n^{\boldsymbol{d}}\right)$ ($T(n) = 2T(\frac{n}{2}) + 1$)

Ask the Audience!

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

- Use the Master Theorem to Solve:
  - $T(n) = 16 \cdot T\left(\frac{n}{4}\right) + Cn^2$ — $a = 16$, $b = 4$, $d = 2$; $\frac{a}{b^d} = 1 \Rightarrow \Theta(n^2 \log n)$
  - $T(n) = 21 \cdot T\left(\frac{n}{5}\right) + Cn^2$ — $a = 21$, $b = 5$, $d = 2$; $\frac{a}{b^d} < 1 \Rightarrow \Theta(n^2)$
  - $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + C$ — $a = 2$, $b = 2$, $d = 0$; $\frac{a}{b^d} > 1 \Rightarrow \Theta(n^{\log_b(a)}) = \Theta(n)$
  - $T(n) = 1 \cdot T\left(\frac{n}{2}\right) + C$

Divide and Conquer: Selection (Median)

Selection

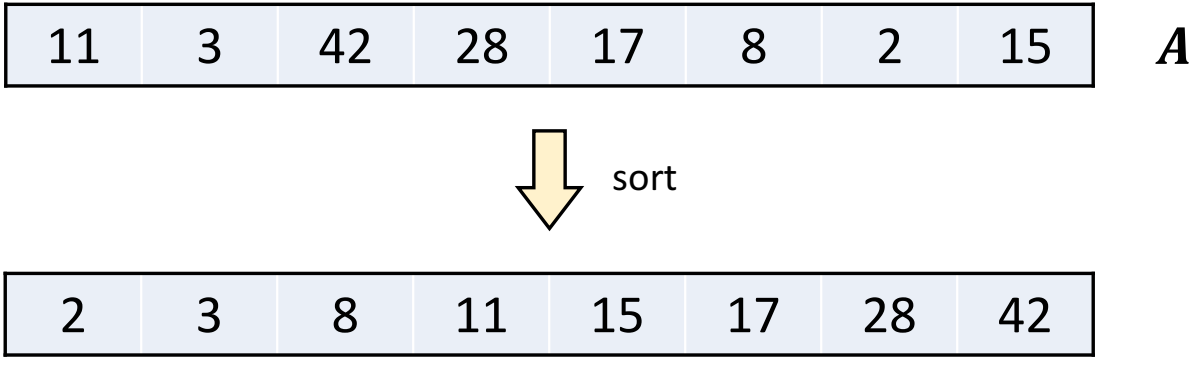
- Given an array of numbers $A[1:n]$, how quickly can I find the:
  - Smallest number? $O(n)$ time
  - Second smallest? $O(n)$ time
  - k-th smallest? $O(nk)$ time
  - median? $O(n^2)$ time

    $\lceil \frac{n}{2} \rceil$-nd smallest

11	3	42	28	17	8	2	15	A

Selection

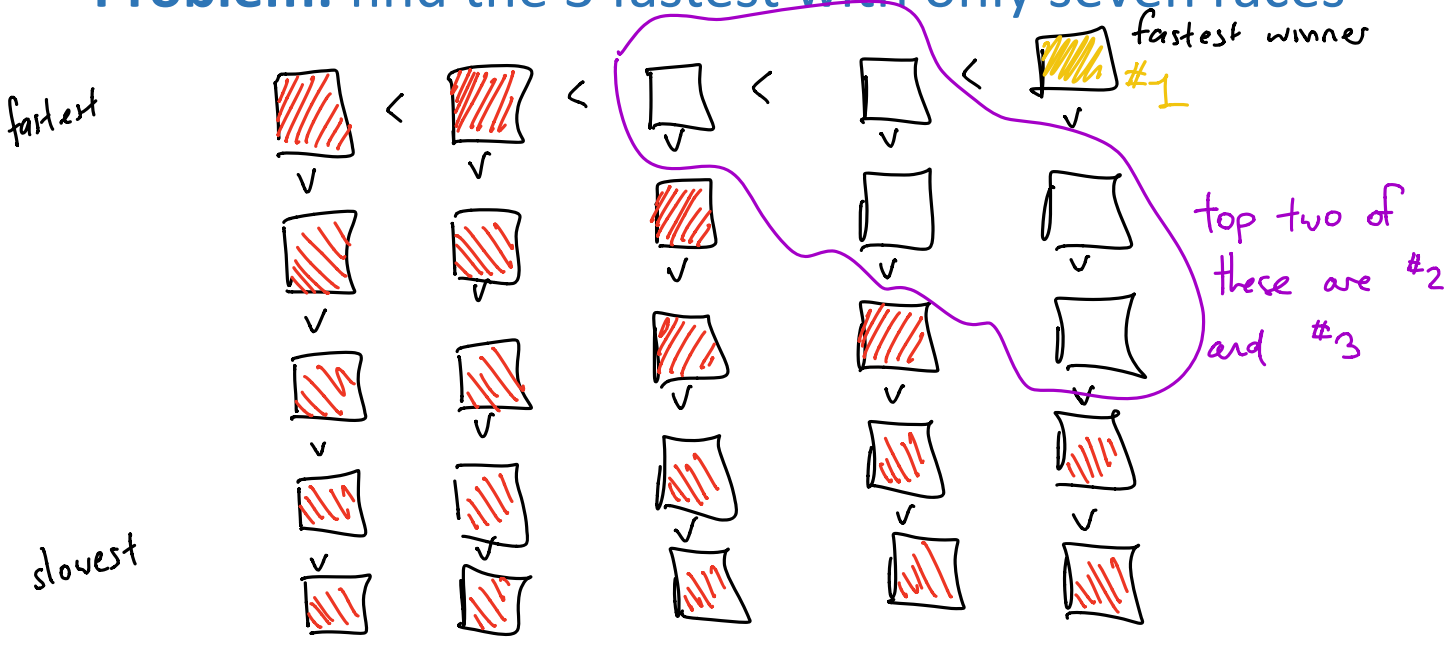
- **Fact:** can select the k-th smallest in $O(nk)$ time
- **Fact:** can select the k-th smallest in $O(n \log n)$ time
  - Sort the list, then return $A[k]$



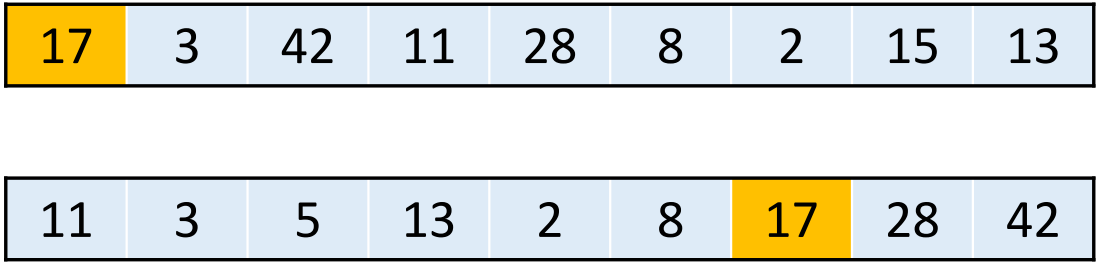
- **Today:** select the k-th smallest in $O(n)$ time

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



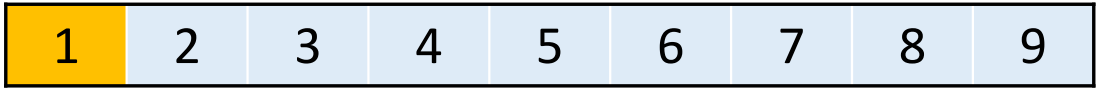
Median Algorithm: Take I



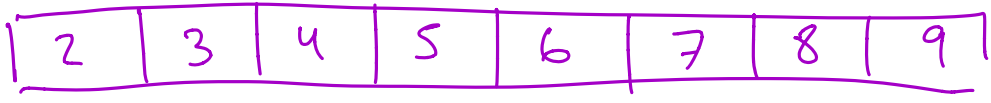

```
Select(A[1:n],k):
  If(n = 1): return A[1]

  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

Median Algorithm: Take I



$k=9$



$k=8$

$k=7$

$\vdots$

$n + (n-1) + (n-2) + \ldots + 2 + 1 = \Theta(n^2)$

$k=1$

Median Algorithm: Take II

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

The median is the best pivot, but also it's the problem we're trying to solve.

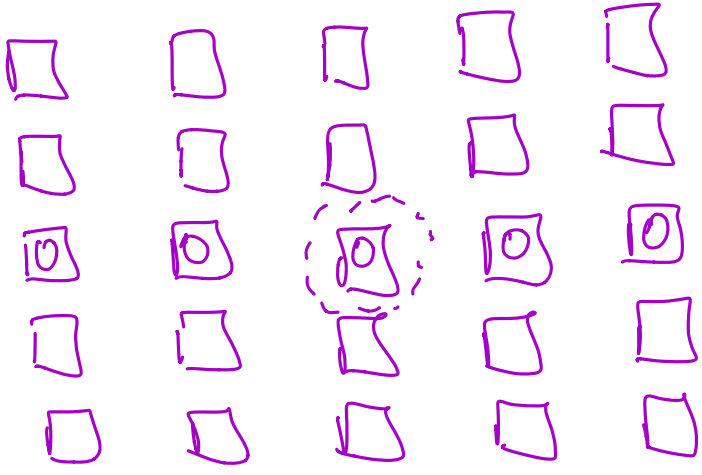
It's enough to find an element in the middle half of the list.

Median of Medians

Time: $O(n) + T(\frac{n}{5})$

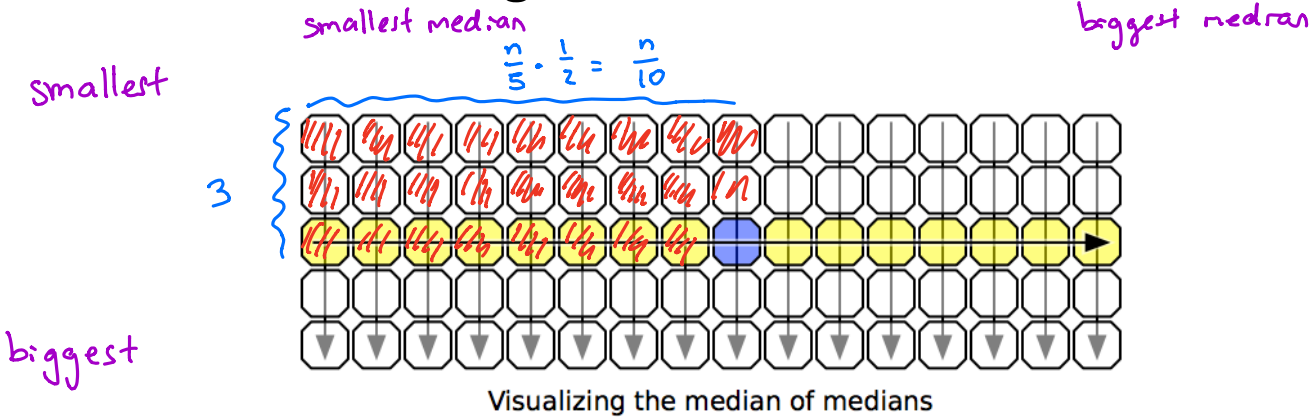
```
MOM(A[1:n]):
  Let m ← ⌈n/5⌉
  For i = 1,…,m:                                   O(n)
    Meds[i] = median{A[5i-4],A[5i-3],…,A[5i]}      } O(1)
  Let p ← Select(Meds[1:m],⌊m/2⌋)                  T(n/5)
```

Median of the medians



Median of Medians

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

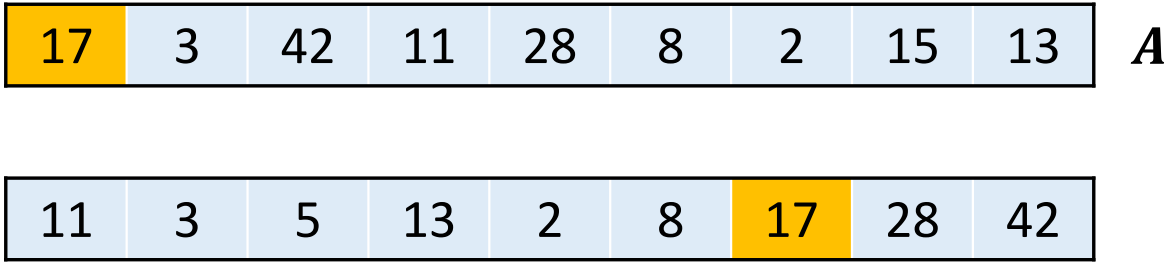


Visualizing the median of medians

All red elts are smaller than the MOM

# of red elts is $3 \times \frac{n}{10} = \frac{3n}{10}$

Median Algorithm: Take II

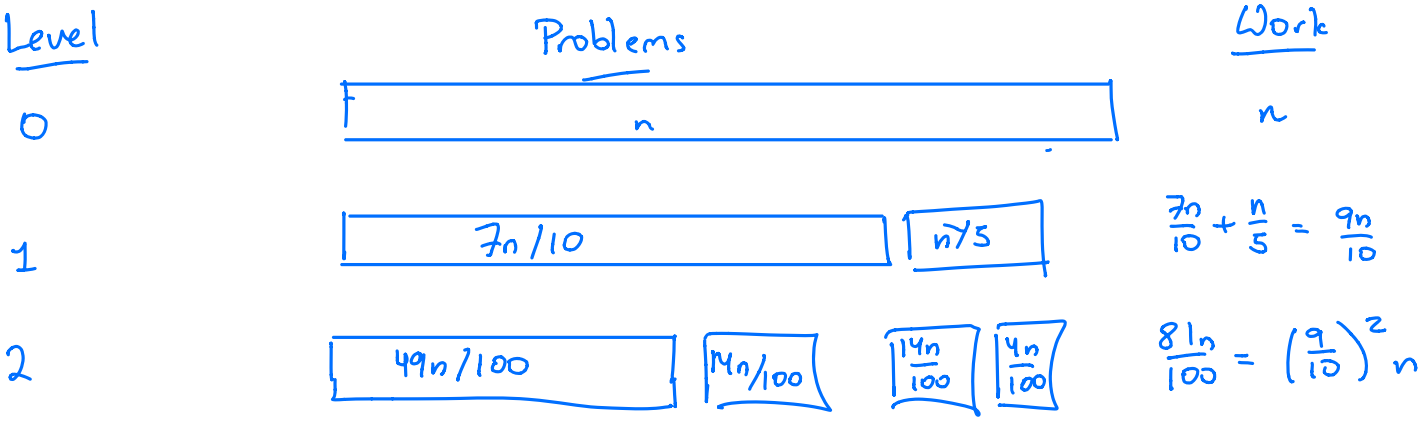



```
MOMSelect(A[1:n],k):
  If(n ≤ 25): return median{A}

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

  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)   ] T(7n/10)
```

Running Time Analysis

Recursion Tree

$$T(n) = T\left(\frac{7n}{10}\right) + T\left(\frac{n}{5}\right) + cn$$
$$T(1) = c$$



Proof by Induction

$$T(n) = T\left(\frac{7n}{10}\right) + T\left(\frac{n}{5}\right) + n$$
$$T(1) = 1$$

- **Claim:** $T(n) = O(n)$

($T(n) \le cn$ for some c I'll choose later)

Base Case: $T(1) = 1 \le cn$ (as long as $c \ge 1$)

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

$$\le c \cdot \frac{7n}{10} + c \cdot \frac{n}{5} + n$$

$$= \left(c \cdot \frac{7}{10} + c \cdot \frac{1}{5} + 1\right) \cdot n$$

$$= \left(c \cdot \frac{9}{10} + 1\right) \cdot n$$

$$\le c \cdot n$$

(as long as $c \ge 10$)

$\frac{9c}{10} + 1 \le c$

$9c + 10 \le 10c$

$10 \le c$

Ask the Audience

- If we change MOM so that it uses n/3 blocks of size 3, would Select still run in O(n) time?

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!

Divide and Conquer: Binary Search

Binary Search

Is 28 in this list?

Binary Search

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

Search(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 Search(A,ℓ,m-1,t)
  Else: return Search(A,m+1,r,t)
```

Running Time Analysis

$$T(n) = T(n/2) + C$$
$$T(1) = C$$

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
  - $T(n) = T(n/2) + C$