CS3000: Algorithms & Data
Jonathan Ullman

Lecture 3:

- Divide and Conquer: Karatsuba
- Solving Recurrences

Jan 13, 2020

The "Master Theorem"

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

Ask the Audience!

- Use the Master Theorem to Solve:
  - $T(n) = 16 \cdot T\left(\frac{n}{4}\right) + Cn^2$
  - $T(n) = 21 \cdot T\left(\frac{n}{5}\right) + Cn^2$
  - $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + C$
  - $T(n) = 1 \cdot T\left(\frac{n}{2}\right) + C$

Divide and Conquer: Selection (Median)

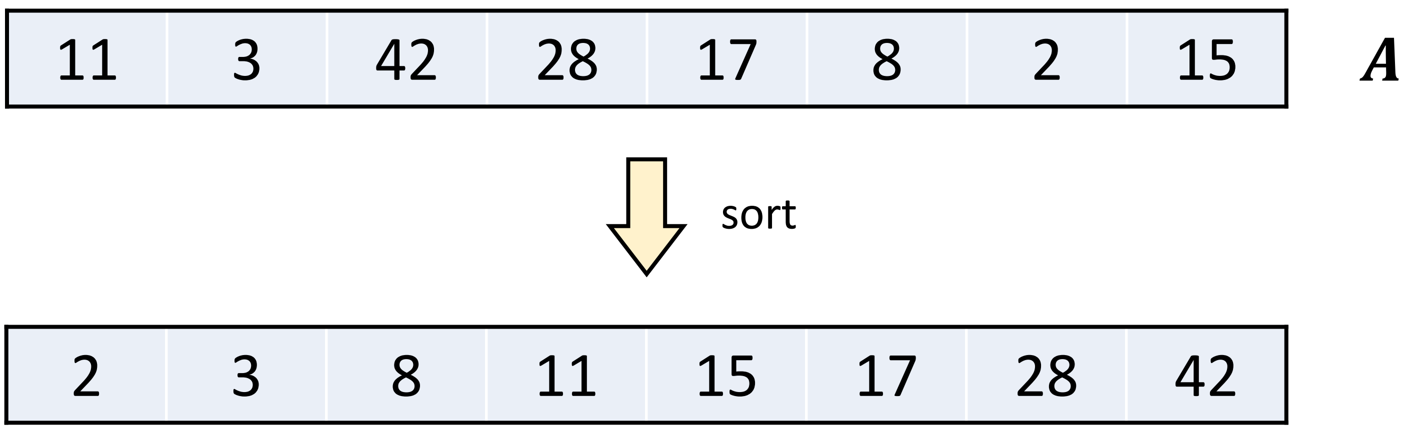
Selection

- Given an array of numbers $A[1:n]$, how quickly can I find the:
  - Smallest number?
  - Second smallest?
  - k-th smallest?

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

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

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

11	3	5	13	2	8	17	28	42

```
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)
```

Median Algorithm: Take I

1	2	3	4	5	6	7	8	9	A

Median Algorithm: Take II

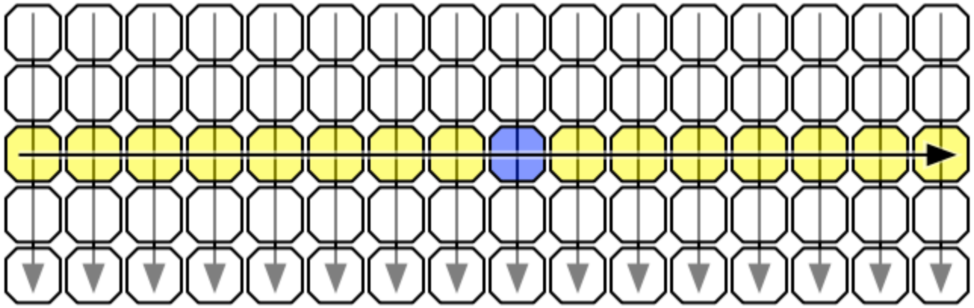
- **Problem:** we need to find a good pivot element

Median of Medians

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

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

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

Recursion Tree

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

Proof by Induction

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

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

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?

2	3	8	11	15	17	28	42	A

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