# CS 341: ALGORITHMS

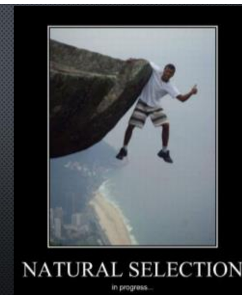
**Lecture 4: divide & conquer III**

Readings: see website

Trevor Brown
https://student.cs.uwaterloo.ca/~cs341
trevor.brown@uwaterloo.ca



## THE **SELECTION** PROBLEM






## THE **SELECTION** PROBLEM

- Input: An array **A** containing **n <u>distinct</u>** integer values, and an integer **k** between 1 and **n**
- Output: The **k**-th **smallest** integer in **A**
- **Minimum** is a special case where $k = 1$
- **Median** is a special case where $k = \frac{n}{2}$
- **Maximum** is a special case where $k = n$
- Simple algorithm for solving selection?



Suppose we choose a pivot element $y$ in the array $A$, and we restructure $A$ so that all elements less than $y$ precede $y$ in $A$, and all elements greater than $y$ occur after $y$ in $A$. (This is exactly what is done in *Quicksort*, and it takes linear time.)

$y$ = 23

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ | 12 | 4 | 6 | 27 | 23 | 17 | 40 | 9 |
| | 12 | 4 | 6 | 27 | 23 | 17 | 40 | 9 |
| $Restructure(A, y)$ | 12 | 4 | 6 | 17 | 9 | 23 | 27 | 40 |

$y$ = 6

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ | 12 | 4 | 6 | 27 | 23 | 17 | 40 | 9 |
| | 12 | 4 | 6 | 27 | 23 | 17 | 40 | 9 |
| $Restructure(A, y)$ | 4 | 6 | 12 | 27 | 23 | 17 | 40 | 9 |

Number of elements on each side depend on the **value** $y$...



$A$ after $Restructure(A, y)$

| 1 | 2 | 3 | ... | | $i_y$ | ... | $n$ |
|---|---|---|---|---|---|---|---|
| 12 | 4 | 6 | 17 | 9 | 23 | 27 | 40 |

$A_L$ (elements 12, 4, 6, 17, 9), $y$ (23), $A_R$ (27, 40)

Number of elements in this range (through $y$) = $i_y$

- **What's the $k$-th smallest element of $A$?**
  - If $k = i_y$ then $y$
  - If $k < i_y$ then the $k$th smallest in $A_L$
  - If $k > i_y$ then the $(k - i_y)$th smallest in $A_R$

Recursive calls



```
QuickSelect(k, A[1..n])
    if n = 1 then return A[1]   // base case

    y = A[1]                    // pick an arbitrary pivot
    (AL, AR, iy) = Restructure(A, y)

    if k == iy return y
    else if k < iy then     return QuickSelect(k, AL)
    else /* k > iy */       return QuickSelect(k - iy, AR)

Restructure(A[1..n], y)
    AL = new array[1..n]        // allocate more than enough
    AR = new array[1..n]        // to avoid need for expansion
    nL = 0, nR = 0

    for i = 1 .. n
        if   A[i] < y then  AL[nL++] = A[i]
        else A[i] > y then  AR[nR++] = A[i]

    return (AL, AR, nL+1)       // nL+1 is the new index of y
```

Precondition: $1 \le k \le n$

## OVERLY OPTIMISTIC ANALYSIS ☺

$A$ after $Restructure(A, y)$

| 12 | 4 | 6 | 17 | 9 | 23 | 27 | 40 |
|---|---|---|---|---|---|---|---|

$A_L$ $y$ $A_R$

- If $i_y = \frac{n}{2}$, then we recurse on $\sim\frac{n}{2}$ elements,
- If we could **always** recurse on $\frac{n}{2}$ elements then
  - We would get $T(n) = T\left(\frac{n}{2}\right) + \Theta(n)$
  - Which would yield $a = 1, b = 2, y = 1, x = \log_2 1 = 0$, $y > x$ and $T(n) \in \Theta(n^y) = \boldsymbol{\Theta(n)}$ by the Master theorem.

But we **don't** always recurse on $\frac{n}{2}$ elements!

7

## WORST-CASE ANALYSIS

$A$ after $Restructure(A, y)$

| 12 | 4 | 6 | 17 | 9 | 23 | 27 | 40 |
|---|---|---|---|---|---|---|---|

$A_L$ $y$ $A_R$

- If we always get $i_y = 1$ and recurse on the right, then
- We get $T(n) = T(n-1) + \Theta(n)$
- By the substitution method this is $\boldsymbol{\Theta(n^2)}$

- So, sometimes the pivot is good, sometimes it's bad…
- **What about the average case?**

8

## AVERAGE-CASE ANALYSIS

- Definition: we say a pivot $y$ is **good** if $i_y \in \left(\frac{n}{4}, \frac{3n}{4}\right)$

1 $\frac{n}{4}$ $\frac{3n}{4}$ $n$

| … | … | … | … | … | y | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

$\sim\frac{n}{4}$ **elements** $\sim\frac{n}{2}$ **elements** $\sim\frac{n}{4}$ **elements**

| | | | | | y | … | … | … | … | … | … | … | … | … | … |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

- For **any good pivot** we recurse on at most $\frac{3n}{4}$ elements

Reducing the size of the subproblem by at least 1/4

- Probability of an arbitrary pivot being **good**?

p = 1/2

9

## PROOF SKETCH

- Since probability of a good pivot is ½,
- on average, every **two** recursive calls, we will encounter a **good pivot**
- Encountering a good pivot reduces problem size to at most $\frac{3n}{4}$
- So, problem size is reduced to $\frac{3n}{4}$ after **expected linear work**
- **Average case recurrence:** $T(n) = T\left(\frac{3n}{4}\right) + \Theta(n)$
  - $T(n) \in \Theta(n)$

10

Here is a more rigorous proof of the average-case complexity: We say the algorithm is in phase $j$ if the current subarray has size $s$, where

$$n\left(\frac{3}{4}\right)^{j+1} < s \le n\left(\frac{3}{4}\right)^{j}.$$

Let $X_j$ be a random variable that denotes the amount of computation time occurring in phase $j$. If the pivot is in the middle half of the current subarray, then we transition from phase $j$ to phase $j+1$. This occurs with probability 1/2, so the expected number of recursive calls in phase $j$ is 2. The computing time for each recursive call is linear in the size of the current subarray, so $E[X_j] \le 2cn(3/4)^j$ (where $E[\cdot]$ denotes the expectation of a random variable). The total time of the algorithm is given by $X = \sum_{j\ge0} X_j$. Therefore

$$E[X] = \sum_{j\ge0} E[X_j] \le 2cn \sum_{j\ge0} (3/4)^j = 8cn \in O(n).$$

$$\sum_{k=0}^{\infty} ar^k = \frac{a}{1-r}, \text{ for } |r| < 1.$$

This is just for your notes, in case you want to know how you'd do this analysis formally

11

## TAKING SELECTION FURTHER

- We just showed:
  - QuickSelect with **average case** runtime in $O(n)$
- Next up:
  - Median-of-medians QuickSelect (MOMQuickSelect)
  - **worst case** runtime in $O(n)$

Relies on getting a **good pivot** within $O(1)$ recursive calls **on average**

Must get a **good pivot** within $O(1)$ recursive calls **always**

The algorithm we will see picks a **good pivot** in **every** recursive call

12

## HIGH LEVEL ALGORITHM

- Similar to QuickSelect
  - **Choose** a pivot
  - Move smaller elements to the left of the pivot, and larger elements to the right of the pivot
  - Recursively call MOMQuickSelect on one subarray
- Only difference is **how** we choose the pivot
  - **Always** want to pick a **good pivot**

13

## ALWAYS PICKING A **GOOD PIVOT**

Example input A[1…50]: 11, 38, 6, 21, 20, 17, 14, 9, 7, 5, 8, 34, 49, 47, 28, 18, 44, 31, 46, 48, 27, 4, 2, 50, 23, 45, 3, 13, 43, 22, 10, 32, 35, 41, 24, 1, 30, 12, 15, 26, 16, 19, 36, 33, 37, 39, 25, 40, 29, 42

Group into **rows of 5**

| | | | | |
|---|---|---|---|---|
| 11 | 38 | 6 | 21 | 20 |
| 17 | 14 | 9 | 7 | 5 |
| 8 | 34 | 49 | 47 | 28 |
| 18 | 44 | 31 | 46 | 48 |
| 27 | 4 | 2 | 50 | 23 |
| 45 | 3 | 13 | 43 | 22 |
| 10 | 32 | 35 | 41 | 24 |
| 1 | 30 | 12 | 15 | 26 |
| 16 | 19 | 36 | 33 | 37 |
| 39 | 25 | 40 | 29 | 42 |

**Time complexity** for this step?

Find **median** of **each row**

| | | | | |
|---|---|---|---|---|
| 11 | 38 | 6 | 21 | **20** |
| 17 | 14 | **9** | 7 | 5 |
| 8 | **34** | 49 | 47 | 28 |
| 18 | **44** | 31 | 46 | 48 |
| 27 | 4 | 2 | 50 | **23** |
| 45 | 3 | 13 | 43 | **22** |
| 10 | **32** | 35 | 41 | 24 |
| 1 | 30 | 12 | **15** | 26 |
| 16 | 19 | 36 | **33** | 37 |
| **39** | 25 | 40 | 29 | 42 |

**Time complexity** for this step?

Build **array of medians**

20, 9, 34, 44, 23, 22, 32, 15, 33, 39

Time complexity?

**Recursively** find the median of these medians: **23**

Recursive problem size?

14

## **HOW GOOD** IS THE PIVOT **23**?

Recall: median of each row

| | | | | |
|---|---|---|---|---|
| 11 | 38 | 6 | 21 | **20** |
| 17 | 14 | **9** | 7 | 5 |
| 8 | **34** | 49 | 47 | 28 |
| 18 | **44** | 31 | 46 | 48 |
| 27 | 4 | 2 | 50 | *23* |
| 45 | 3 | 13 | 43 | **22** |
| 10 | **32** | 35 | 41 | 24 |
| 1 | 30 | 12 | **15** | 26 |
| 16 | 19 | 36 | **33** | 37 |
| **39** | 25 | 40 | 29 | 42 |

**Imagine** sorting each row:

| | | |
|---|---|---|
| 6 | 11 | **20** |
| 5 | 7 | **9** |
| 8 | 28 | **34** |
| 18 | 31 | **44** |
| 2 | 4 | *23* |
| 3 | 13 | **22** |
| 10 | 24 | **32** |
| 1 | 12 | **15** |
| 16 | 19 | **33** |
| 25 | 29 | **39** |

**Then** ordering rows by medians:

| | | | | |
|---|---|---|---|---|
| 5 | 7 | **9** | 14 | 17 |
| 1 | 12 | **15** | 26 | 30 |
| 6 | 11 | **20** | 21 | 38 |
| 3 | 13 | **22** | 43 | 45 |
| 2 | 4 | *23* | 27 | 50 |
| 10 | 24 | **32** | 35 | 41 |
| 16 | 19 | **33** | 36 | 37 |
| 8 | 28 | **34** | 47 | 49 |
| 25 | 29 | **39** | 40 | 42 |
| 18 | 31 | **44** | 46 | 48 |

# elements ≤ 23 is **at least** 3(5). This is at least 3/10ths of our 50-element input, or $\mathbf{3n/10}$.

# elements ≥ 23 is **at least** 3(6). This is > **3/10ths** of our 50-element input.

So, after restructuring, pivot 23 must have **at least** $\mathbf{3n/10}$ elements **before *and* after** it

This is a **good** pivot!

We recurse on $A_L$ or $A_R$, and both have size **at most** $\mathbf{7n/10}$

15

$MOMQuickSelect(k = 11, n = 14, A)$

11, 38, 6, 21, 20, 17, 14, 9, 7, 5, 8, 34, 49, 47

5, 6, 7, 9, 7, 11, 14, 17, 20, 21, 34, 38, 47, 49

```
MOMQuickSelect(k, n, A)
    // base case
    if n <= 14 then sort(A) and return A[k]

    // divide and conquer to find medians
    r = (n-5) / 10
    medians[1..(2*r+1)] = new array
    for i = 1..(2*r+1)
        B[1..5] = A[(5*(i-1)+1)..(5*i)]
        sort(B)
        medians[i] = B[3]

    y = MOMQuickSelect(r+1, 2*r+1, medians)

    // divide and conquer to find rank k
    (AL, AR, iy) = Restructure(A, y)
    if       k == iy      then return y
    else if k < iy        then return MOMQuickSelect(k, iy-1, AL)
    else /* k > iy */     then return MOMQuickSelect(k-iy, n-iy, AR)
```

16

$MOMQuickSelect(k = 11, n = 21, A)$

11, 38, 6, 21, 20, 17, 14, 9, 7, 5, 8, 34, 49, 47, 28, 18, 44, 31, 46, 48, 27

$r = \left\lfloor \frac{21-5}{10} \right\rfloor = 1$

Not considering at most 9 elements

| | | | | | |
|---|---|---|---|---|---|
| $B$ | 11 | 38 | 6 | 21 | 20 |
| $sort(B)$ | 6 | 11 | **20** | 21 | 38 |
| | 17 | 14 | 9 | 7 | 5 |
| | 5 | 7 | **9** | 14 | 17 |
| | 8 | 34 | 49 | 47 | 28 |
| | 8 | 28 | **34** | 47 | 49 |

$medians$: 20, 9, 34

$y = MOMQuickSelect(2, 3, [20, 9, 34]) \Rightarrow 20$

```
MOMQuickSelect(k, n, A)
    // base case
    if n <= 14 then sort(A) and return A[k]

    // divide and conquer to find medians
    r = (n-5) / 10
    medians[1..(2*r+1)] = new array
    for i = 1..(2*r+1)
        B[1..5] = A[(5*(i-1)+1)..(5*i)]
        sort(B)
        medians[i] = B[3]

    y = MOMQuickSelect(r+1, 2*r+1, medians)

    // divide and conquer to find rank k
    (AL, AR, iy) = Restructure(A, y)
    if       k == iy      then return y
    else if k < iy        then return MOMQuickSelect(k, iy-1, AL)
    else /* k > iy */     then return MOMQuickSelect(k-iy, n-iy, AR)
```

17

$MOMQuickSelect(k = 11, n = 21, A)$

11, 38, 6, 21, 20, 17, 14, 9, 7, 5, 8, 34, 49, 47, 28, 18, 44, 31, 46, 48, 27

$Restructure(A, y = 20) \Rightarrow$

$A_L = [11, 6, 17, 14, 9, 7, 5, 8, 18]$

$A_R = [38, 21, 34, 49, 47, 28, 44, 31, 46, 48, 27]$

$i_y = |A_L| + 1 = 10$

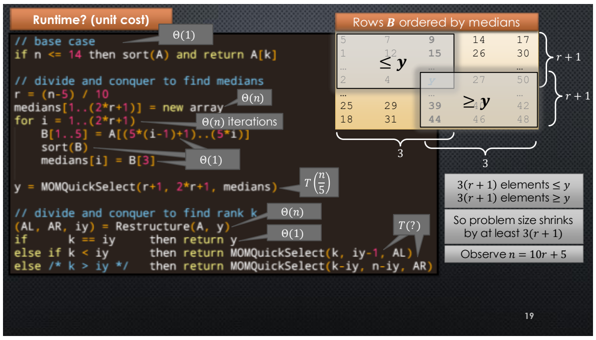
$k = 11 \quad > \quad i_y = 10$

$k - i_y = 1 \qquad n - i_y = 10$

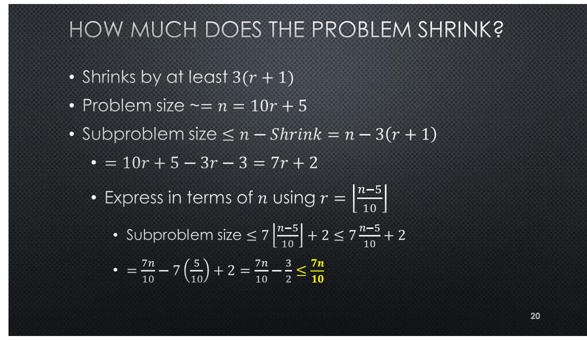
$MOMQuickSelect(1, 10, A_R) \Rightarrow 21$

```
MOMQuickSelect(k, n, A)
    // base case
    if n <= 14 then sort(A) and return A[k]

    // divide and conquer to find medians
    r = (n-5) / 10
    medians[1..(2*r+1)] = new array
    for i = 1..(2*r+1)
        B[1..5] = A[(5*(i-1)+1)..(5*i)]
        sort(B)
        medians[i] = B[3]

    y = MOMQuickSelect(r+1, 2*r+1, medians)

    // divide and conquer to find rank k
    (AL, AR, iy) = Restructure(A, y)
    if       k == iy      then return y
    else if k < iy        then return MOMQuickSelect(k, iy-1, AL)
    else /* k > iy */     then return MOMQuickSelect(k-iy, n-iy, AR)
```

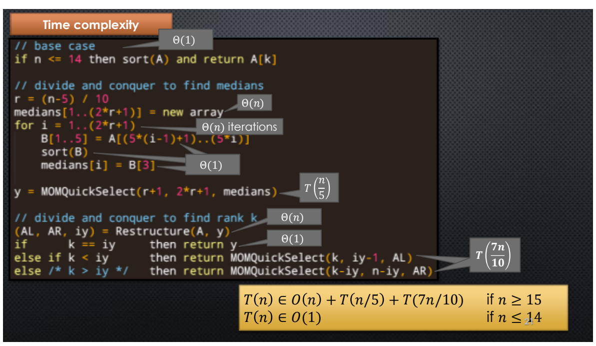
18



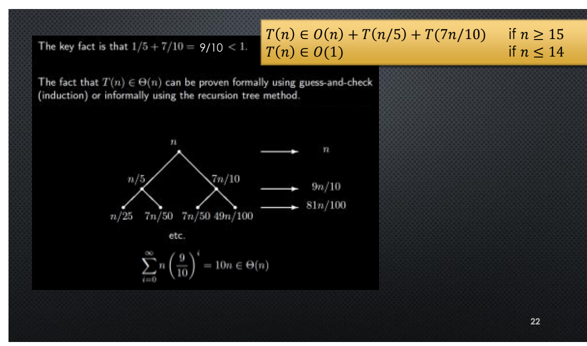
Runtime? (unit cost)
// base case
if n <= 14 then sort(A) and return A[k]
// divide and conquer to find medians
r = (n-5) / 10
medians[1..(2*r+1)] = new array
for i = 1..(2*r+1)
B[1..5] = A[(5*(i-1)+1)..(5*i)]
sort(B)
medians[i] = B[3]
y = MOMQuickSelect(r+1, 2*r+1, medians)
// divide and conquer to find rank k
(AL, AR, iy) = Restructure(A, y)
if k == iy then return y
else if k < iy then return MOMQuickSelect(k, iy-1, AL)
else /* k > iy */ then return MOMQuickSelect(k-iy, n-iy, AR)
$\Theta(1)$
$\Theta(n)$
$\Theta(n)$ iterations
$\Theta(1)$
$T\left(\frac{n}{5}\right)$
$\Theta(n)$
$\Theta(1)$
$T(?)$
Rows $B$ ordered by medians
5 7 9 14 17
1 12 15 26 30
2 4 y 27 50
25 29 39 40 42
18 31 44 46 48
$\leq y$
$\geq y$
$r+1$
$r+1$
3
3
$3(r+1)$ elements $\leq y$
$3(r+1)$ elements $\geq y$
So problem size shrinks by at least $3(r+1)$
Observe $n = 10r + 5$
19

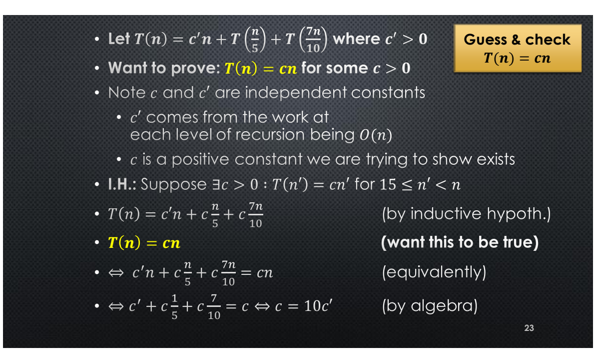
HOW MUCH DOES THE PROBLEM SHRINK?
Shrinks by at least $3(r+1)$
Problem size ~= $n = 10r + 5$
Subproblem size $\leq n - Shrink = n - 3(r+1)$
$= 10r + 5 - 3r - 3 = 7r + 2$
Express in terms of $n$ using $r = \left\lfloor \frac{n-5}{10} \right\rfloor$
Subproblem size $\leq 7\left\lfloor \frac{n-5}{10} \right\rfloor + 2 \leq 7\frac{n-5}{10} + 2$
$= \frac{7n}{10} - 7\left(\frac{5}{10}\right) + 2 = \frac{7n}{10} - \frac{3}{2} \leq \frac{7n}{10}$
20

Time complexity
// base case
if n <= 14 then sort(A) and return A[k]
// divide and conquer to find medians
r = (n-5) / 10
medians[1..(2*r+1)] = new array
for i = 1..(2*r+1)
B[1..5] = A[(5*(i-1)+1)..(5*i)]
sort(B)
medians[i] = B[3]
y = MOMQuickSelect(r+1, 2*r+1, medians)
// divide and conquer to find rank k
(AL, AR, iy) = Restructure(A, y)
if k == iy then return y
else if k < iy then return MOMQuickSelect(k, iy-1, AL)
else /* k > iy */ then return MOMQuickSelect(k-iy, n-iy, AR)
$\Theta(1)$
$\Theta(n)$
$\Theta(n)$ iterations
$\Theta(1)$
$T\left(\frac{n}{5}\right)$
$\Theta(n)$
$\Theta(1)$
$T\left(\frac{7n}{10}\right)$
$T(n) \in O(n) + T(n/5) + T(7n/10)$ if $n \geq 15$
$T(n) \in O(1)$ if $n \leq 14$

$T(n) \in O(n) + T(n/5) + T(7n/10)$ if $n \geq 15$
$T(n) \in O(1)$ if $n \leq 14$
The key fact is that $1/5 + 7/10 = 9/10 < 1$.
The fact that $T(n) \in \Theta(n)$ can be proven formally using guess-and-check (induction) or informally using the recursion tree method.
$n$
$n/5$
$7n/10$
$n/25$ $7n/50$ $7n/50$ $49n/100$
$n$
$9n/10$
$81n/100$
etc.
$\sum_{i=0}^{\infty} n\left(\frac{9}{10}\right)^i = 10n \in \Theta(n)$
22

Let $T(n) = c'n + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$ where $c' > 0$
Want to prove: $T(n) = cn$ for some $c > 0$
Note $c$ and $c'$ are independent constants
$c'$ comes from the work at each level of recursion being $O(n)$
$c$ is a positive constant we are trying to show exists
I.H.: Suppose $\exists c > 0 : T(n') = cn'$ for $15 \leq n' < n$
$T(n) = c'n + c\frac{n}{5} + c\frac{7n}{10}$ (by inductive hypoth.)
$T(n) = cn$ (want this to be true)
$\Leftrightarrow c'n + c\frac{n}{5} + c\frac{7n}{10} = cn$ (equivalently)
$\Leftrightarrow c' + c\frac{1}{5} + c\frac{7}{10} = c \Leftrightarrow c = 10c'$ (by algebra)
Guess & check
$T(n) = cn$
23