THE SELECTION PROBLEM

• Input: An array $A$ containing $n$ distinct 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 $k = n/2$
• Maximum is a special case where $k = n$
• Simple algorithm for solving selection?

Recursive calls

QuickSelect($k$, $A$, $n$)

if $n = 1$ then return $A[1]$ // base case

$y = A[1]$ // pick an arbitrary pivot

if $k = y$ then return $A[1]$ // if $k$ is the pivot

else if $k < y$ then return QuickSelect($k$, $A[1:]$) // if $k$ is before the pivot

else $k > y$ then return QuickSelect($k - y$, $A[2:]$) // if $k$ is after the pivot

Restructure($A[1:]$, $y$) // allocate more than enough

for $i = 0$ to $n$ do

$A[i] = AL = new array[1:n]$ // to avoid need for expansion

if $A[i] < y$ then

else $A[i] > y$ then

return $A[1:]$ // ML+1 is the new index of $y$

Recursive calls

Restructure($A[1:]$, $y$)}
OVERLY OPTIMISTIC ANALYSIS

If \( i_y = \frac{n}{2} \) then we recurse on \(-\frac{n}{2}\) elements.

If we could always recurse on \( \frac{n}{2} \) elements then

We would get \( T(n) = T(\frac{n}{2}) + \Theta(n) \)

Which would yield \( a = 1, b = 2, y = 1, x = \log_2 1 = 0 \).

\( y > x \) and \( T(n) \in \Theta(n^y) = \Theta(n) \) by the Master theorem.

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

WORST-CASE ANALYSIS

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 \( \Theta(n^2) \)

So, sometimes the pivot is good, sometimes it's bad...

What about the average case?

AVERAGE-CASE ANALYSIS

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

For any good pivot we recurse on at most \( \frac{3n}{4} \) elements

Probability of an arbitrary pivot being good? \( \frac{1}{2} \)

Reducing the size of the subproblem by at least \( 1/4 \)

PROOF SKETCH

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

10

TAKING SELECTION FURTHER

We just showed:

QuickSelect with average case runtime in \( \Theta(n) \)

Next up:

Median-of-medians QuickSelect (MOMQuickSelect)

worst case runtime in \( \Theta(n) \)

The algorithm we will use to get a good pivot in every recursive call:}

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 \( x \), where

\[
\frac{n}{4} \leq x \leq \frac{3n}{4}.
\]

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 \( \frac{1}{2} \), 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 \( E[X_j] \leq 2nX_j/4 \) (where \( E[X_j] \) denotes the expected value of a random variable). The total time of the algorithm is given by \( X = \sum_{j \geq 2} X_j \). Therefore

\[
E[X] = \sum_{j \geq 2} E[X_j] \leq 2n \sum_{j \geq 2} (3/4)^j = 8n \in \Theta(n).
\]

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

How Good is the Pivot

Recall: median of each row

```
| 1 | 38 | 3 | 20 |
| 2 | 8  | 34 | 9  |
| 3 | 34 | 9  | 20 |
| 4 | 34 | 9  | 20 |
| 5 | 34 | 9  | 20 |
| 6 | 34 | 9  | 20 |
```

Imagine sorting each row

```
| 1 | 38 | 3 | 20 |
| 2 | 8  | 34 | 9  |
| 3 | 34 | 9  | 20 |
| 4 | 34 | 9  | 20 |
| 5 | 34 | 9  | 20 |
| 6 | 34 | 9  | 20 |
```

- If elements ≤ 23 is at least 3/5
  - This is ≥ 3/10ths of our 50-element input
  - So, after restructuring, pivots must have at least 3n/10 elements before and after!

Always Picking a Good Pivot

```
Example input A[1...50]:
```

```
46, 48, 27, 4, 2, 50, 23, 45, 3, 15, 43, 22, 15, 32, 35, 41, 24,
1, 30, 32, 36, 39, 37, 35, 39, 35, 39, 33, 37
```

```
Group into rows of 5
```

```
| 11 | 38 | 6 | 21 |
| 12 | 14 | 9 | 7  |
| 13 | 34 | 9  | 47 |
| 14 | 44 | 34 | 48 |
| 15 | 44 | 34 | 48 |
| 16 | 32 | 35 | 41 |
| 17 | 30 | 12 | 20 |
| 18 | 19 | 36 | 33 |
| 19 | 25 | 40 | 23 |
| 20 | 21 |
```

```
Find medians of each row
```

```
| 11 | 38 | 6 | 21 |
| 12 | 14 | 9 | 7  |
| 13 | 34 | 9  | 47 |
| 14 | 44 | 34 | 48 |
| 15 | 44 | 34 | 48 |
| 16 | 32 | 35 | 41 |
| 17 | 30 | 12 | 20 |
| 18 | 19 | 36 | 33 |
| 19 | 25 | 40 | 23 |
| 20 | 21 |
```

```
Lots of medians
```

```
| 11 | 38 | 6 | 21 |
| 12 | 14 | 9 | 7  |
| 13 | 34 | 9  | 47 |
| 14 | 44 | 34 | 48 |
| 15 | 44 | 34 | 48 |
| 16 | 32 | 35 | 41 |
| 17 | 30 | 12 | 20 |
| 18 | 19 | 36 | 33 |
| 19 | 25 | 40 | 23 |
| 20 | 21 |
```

```
# elements ≤ 23 is at least 3/5
```

```
This is ≥ 3/10ths of our 50-element input
```

```
So, after restructuring, pivots must have at least 3n/10 elements before and after!
```

```
We recurse on d_i or d_j, and both have size at most 7n/10
```

```
We want to pick a pivot
```

```
We choose the pivot
```

```
Imagine sorting each row
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```
**How much does the problem shrink?**

- Shrinks by at least $3(r+1)$
- Problem size $= n = 10r + 5$
- Subproblem size $\leq n - \text{Shrink} = n - 3(r + 1)$
  - $= 10r + 5 - 3r - 3 = 7r + 2$
- Express in terms of $n$ using $r = \frac{n-5}{10}$
  - Subproblem size $\leq 7 \left(\frac{n-5}{10}\right) + 2$  
  - $= \frac{7n}{10} - \frac{7}{2} + 2 = \frac{7n}{10} + \frac{1}{2}$

**Time complexity**

- $T(n) \in O(n) + T(n/5) + T(7n/10)$ if $n \geq 15$
- $T(n) \in O(1)$ if $n \leq 14$

**Guess & check**

- Let $T(n) = cn + T\left(\frac{n}{2}\right) + T\left(\frac{n}{10}\right)$ where $c' > 0$
- Want to prove: $T(n) = cn$ for some $c > 0$
- Note $c$ and $c'$ are independent constants
  - $c' = 0$ 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 $3c > 0$ if $T(n') = cn'$ for $15 \leq n' < n$
  - $T(n) = cn + c\frac{n}{2} + c\frac{7n}{10}$ (by inductive hypoth.)
  - $T(n) = cn$ (want this to be true)
  - $\Rightarrow c' + c\frac{1}{2} + c\frac{7}{10} = c \Rightarrow c = 10c'$ (by algebra)