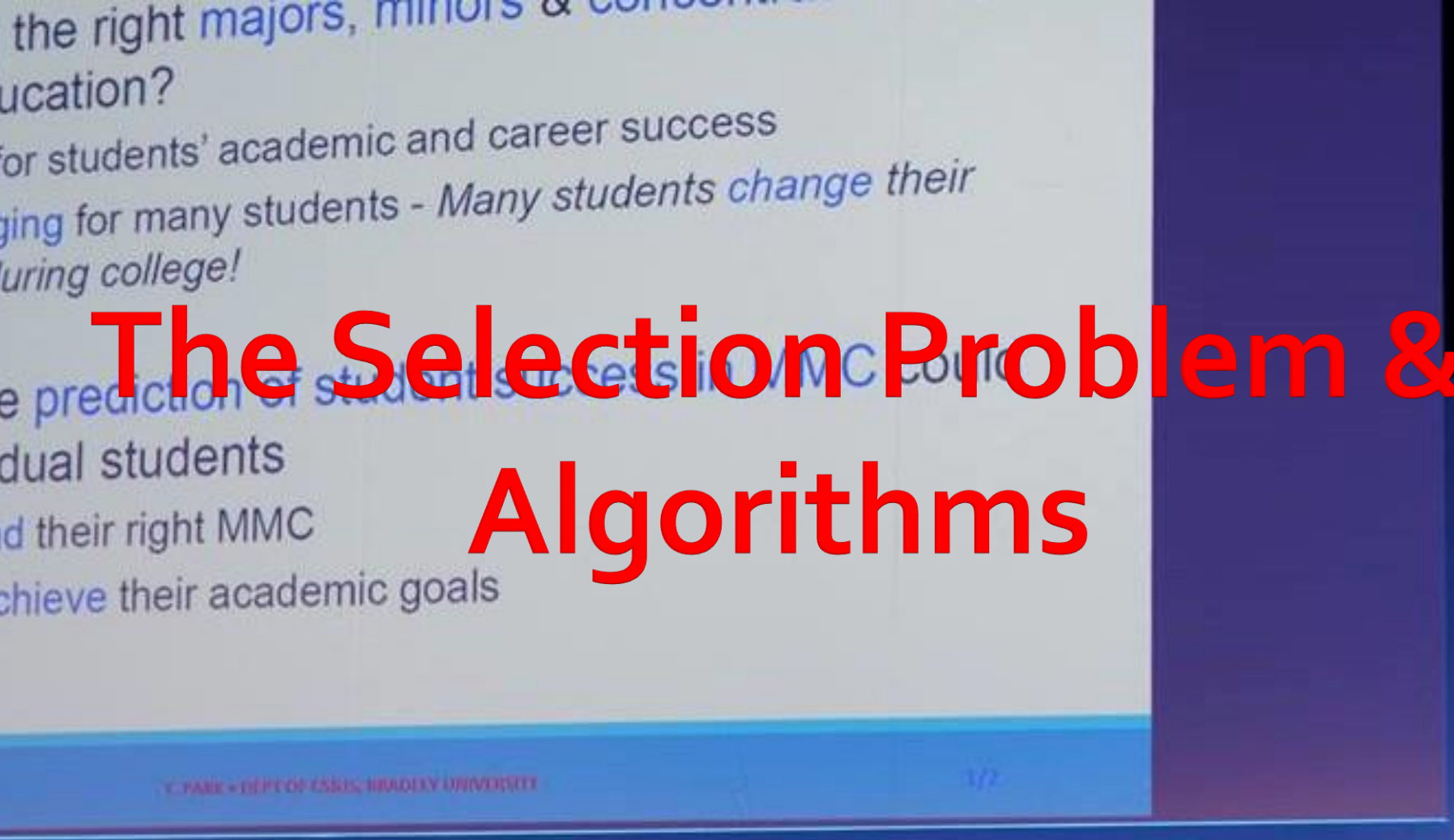
The Selection Problem & Algorithms



Prof. Young Park

✓ The Selection Problems & Algorithms

- A different searching problem!

The Selection Problems

- Keys are in an **unsorted** array:

1. Find the largest or smallest key.
2. Find the smallest and largest keys.
3. Find the second-largest or second-smallest key.
4. **Find the kth largest or smallest key.**

1. Finding the Largest/Smallest Key

- **Algorithm 8.2** Find Largest Key
  - The number of comparisons of keys T(n)= `n - 1`
  - **O(n)** comparisons worst-case

2. Finding Both the Smallest and Largest Keys

- **Algorithm 8.3** Find Smallest and Largest Keys
  - Exactly the number of comparisons done if the smallest and largest keys are found separately.
  - W (n) = **2 (n – 1)** comparisons worst-case
  - **O(n)** comparisons worst-case

Finding Both the Smallest and Largest Keys

- Can we improve ?
  - Compare the keys **in pairs** and find which key in each pair is smaller.
    - **n/2** comparisons.
  - Find the smallest of all the **smaller** keys
    - **n/2** comparisons.
  - Find the largest of all the **larger** keys
    - **n/2** comparisons.
  - W (n) = **3n/2** total comparisons.

Finding Both the Smallest and Largest Keys

- **Algorithm 8.4** Find Smallest and Largest Keys by Pairing Keys
  - $T(n) = 3n/2 - 2$ for even n
    $3n/2 - 3/2$ for odd n
  - **O(n)** comparisons worst-case

3. Finding the Second-Largest/Smallest Key

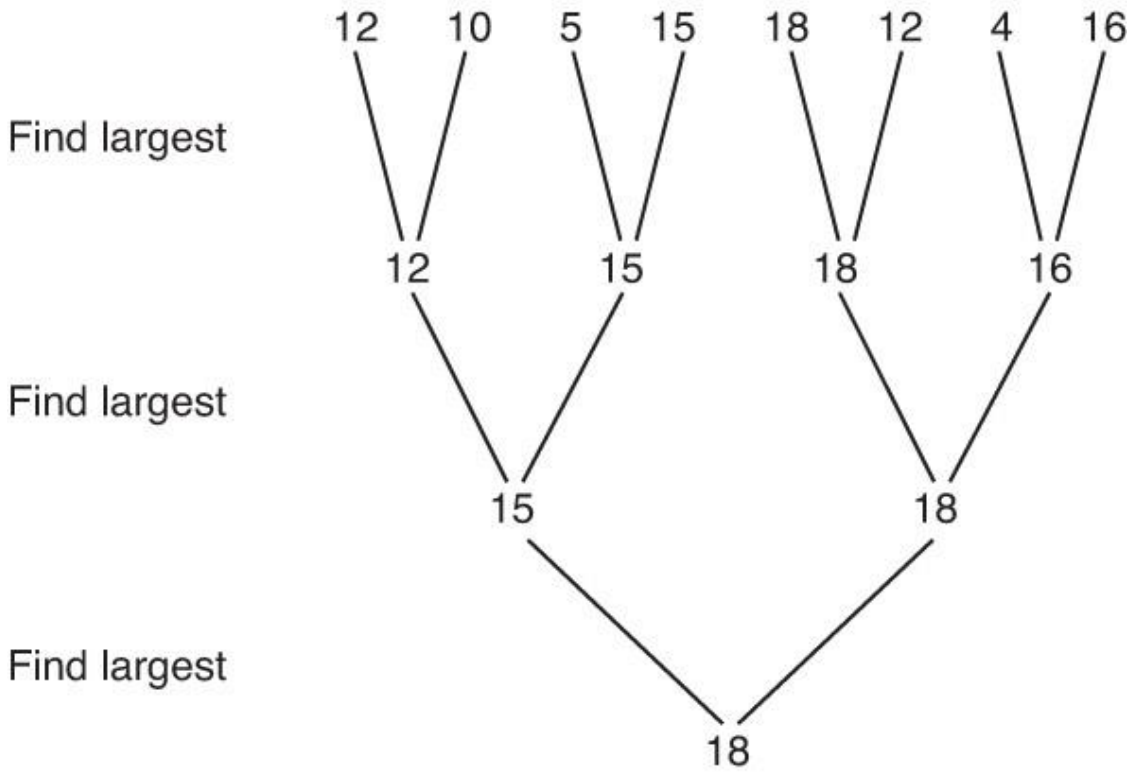
- An approach:
  - Use Algorithm 8.2 Find Largest Key to find the largest key
    - **n – 1** comparisons
  - Eliminate that key!
  - And then use Algorithm 8.2 again to find the largest remaining key
    - **n – 2** comparisons.
  - T(n)= n-1 + n-2 = **2n - 3** comparisons

Finding the Second-Largest Key

- Can we improve?
  - Many of the comparisons done when finding the largest key can be used to eliminate keys from contention for the second largest.
  - **Idea?**
  - Any key that loses to a key other than the largest cannot be the second largest!
  - The **Tournament method** uses this fact.

Finding the Second-Largest Key

Figure 8.10



Finding the Second-Largest Key

- **Observation:**
  - The winner in the last round is the largest key.
  - But, the loser in that round is not necessarily the second-largest key.

Finding the Second-Largest Key

- **Idea?**
- Keep track of all the keys that lose to the largest key and then use Algorithm 8.2 to find the largest of those.
  - We have to maintain linked lists, one for each key.
  - After a key loses a match, it is added to the winning key's linked list.

Finding the Second-Largest Key: Tournament method

- **T(n) =** = **n + log n -2** comparisons
- **O(n)** comparisons worst-case

4. Finding the *k*th-Smallest Key

- Special case of k = n/2: the median

Finding the *k*th-Smallest Key

- **Idea?**
- By sorting:
  - Sort (e.g., mergesort) the keys and return the *k*th key.
  - **O(n log n)** comparisons worst-case

Finding the *k*th-Smallest Key via D&C

- By sorting: **O(n log n)** comparisons worst-case
- Can we do better?
  - Yes! **Apply D&C**!
  - **Linear time O(n) average (expected)-case** time algorithm: QuickSelect
  - **Linear time O(n) worst-case** time algorithm: MedianofMediansSelect

✓ Average-case Linear Time Algorithm for Finding the *k*th-Smallest Key

Average-case Linear Time Algorithm for Finding the *k*th-Smallest Key via D&C

- An **O(n) average (expected)-case time** algorithm for Finding the kth-Smallest Key?
  - **D&C-based!**

Finding the *k*th-Smallest Key by Partitioning

- **Idea?**
- By partitioning:
  - The procedure partition in **Quicksort** partitions an array so that all keys smaller than some pivot item come before it in the array and all keys larger than that pivot item come after it.
- We can solve this kth-smallest General Selection Problem by partitioning until the pivot item is at the *k*th slot.
  - **QuickSelect**
  - **Randomized QuickSelect - A random pivot!**

Good average-case time complexity!

QuickSelect: Finding the *k*th-Smallest Key

- **QuickSelect(S, k):**
  - Pick a pivot element P at random from S.
  - Partition S into subarrays LESS and GREATER as in Quicksort, i.e., LESS P GREATER.

  - If k = p, then output P.
  - If k < p, output **QuickSelect(LESS, k)**
  - If k > p, output **QuickSelect(GREATER, k – p)**

QuickSelect: Finding the *k*th-Smallest Key

- **Algorithm 8.5 General Selection**
  - Divide-and-Conquer
  - **Partition-based**
- **QuickSelect**

▶QUIZ? QuickSelect

- Finding the kth-Smallest Key using **QuickSelect**?
  - k= 4?
  - k= 8?
  - k= 12?

  - S= **[2, 3, 5, 8, 12, 1, 7, 10, 13, 30, 6, 14, 15, 18, 22]**

▶QUIZ?

- Describe the **average-case linear time** algorithm for Finding the kth-Smallest Key?

QuickSelect: Finding the *k*th-Smallest Key **Best-Case**

- **S:** 
- **k**= 1, 2, 3, …, n-1, n
- **p**= 1, 2, 3, …, n-1, n

- **O(n)** Best-case

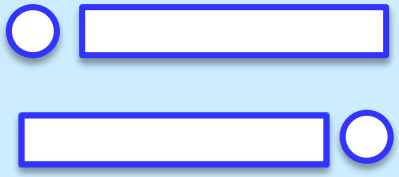
▶QUIZ?

- What is the **best-case** time complexity of QuickSelect?

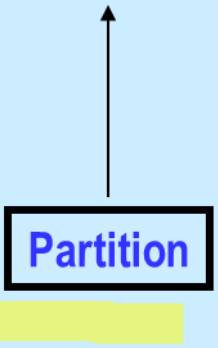
O(n)

QuickSelect: Finding the *k*th-Smallest Key **Worst-case**

- Analysis of Algorithm 8.5 General Selection
- Worst-case?
  - p=1 & k= 2, 3, …, n?
  - p=n & k= 1, 2, …, n-1?



$$W(n) = O(1) \quad \text{if } n=1$$
$$W(n) = W(n-1) + O(n) \quad \text{if } n>1$$



QuickSelect: Finding the *k*th-Smallest Key **Worst-case**

- Analysis of **Algorithm 8.5** General Selection
  - Like Quicksort, the performance is sensitive to the pivot!

$$W(n) = O(1) \quad \text{if } n=1$$
$$W(n) = W(n-1) + O(n) \quad \text{if } n>1$$

- **$O(n^2)$** comparisons worst-case

▶QUIZ?

- What is the **worst-case** time complexity of QuickSelect?

$$W(n) = O(1) \quad \text{if } n=1$$
$$W(n) = W(n-1) + O(n) \quad \text{if } n>1$$

$O(n^2)$

QuickSelect: Finding the *k*th-Smallest Key **Average-Case**

- **S:**

- **k**= 1, 2, 3, …, n-1, n
- **p**= 1, 2, 3, …, n-1, n

QuickSelect: Finding the *k*th-Smallest Key **Average-case**

- Analysis of Algorithm 8.5 General Selection
- Average-case?

  - p=k & k= 1, 2, 3, …, n? n A(0)
  - p=2 & k= 1 | p=n-1 & k=n? 2 *(1)*A(1)
  - p=3 & k= 1,2 | p=n-2 & k=n-1,n? 2*(2)*A(2)
  - p=4 & k= 1,2,3 | p=n-3 & k=n-2,n-1,n? 2*(3)*A(3)
  - …
  - p=i+1 & k= 1,2,3,…,i | p=n-i & k=n-i-1, …,n-2,n-1,n? 2*(i)*A(i)
  - …
  - p=n & k= 1,2,3,…,n-1 | p=1 & k=2, …,n-2,n-1,n? 2*(n-1)*A(n-1)

QuickSelect: Finding the *k*th-Smallest Key **Average-case**

$$A(n) = \frac{n*A(0)+ 2\,[\ 1*A(1)+2*A(2)+\ \ldots\ + i*A(i)+\ \ldots+(n-1)A(n-1)\ ]}{n+2\,(1+2+3+\ldots+i+\ldots+n-1)} + (n-1)$$

QuickSelect: Finding the *k*th-Smallest Key **Average-case**

$$A(n) = \frac{2\,[\,A(1)+2*A(2)+\ \ldots\ +\ i*A(i)+\ \ldots+(n-1)A(n-1)\,]}{n^2} + (n-1)$$

- **A(n) = O(n)**

QuickSelect: Finding the *k*th-Smallest Key **Average-case**

- Analysis of Algorithm 8.5 General Selection
  - Like Quicksort, the performance is sensitive to the pivot.
  - **O(n)** comparisons average (expected)-case

Good average-case time complexity!

▶QUIZ?

- What is the **average (expected)-case** time complexity of QuickSelect?

$$A(n) = \frac{2\,[\,A(1)+2*A(2)+\ldots+i*A(i)+\ldots+(n-1)A(n-1)\,]}{n^2} + (n-1)$$

$$A(n) = O(n)$$

▶QUIZ?

- **QuickSort** vs **QuickSelect**?

✓ Worst-case Linear Time Algorithm for Finding the *k*th-Smallest Key

Worst-case Linear Time Algorithm for Finding the *k*th-Smallest Key

- An **O(n) worst-case time** algorithm for Finding the kth-Smallest Key?
- **Possible?**

- **"Time Bounds for Selection", Blum, Floyd, Pratt, Rivest, Tarjan, JOURNAL OF COMPUTER AND SYSTEM SCIENCES, 7, 448-461 (1973) .**

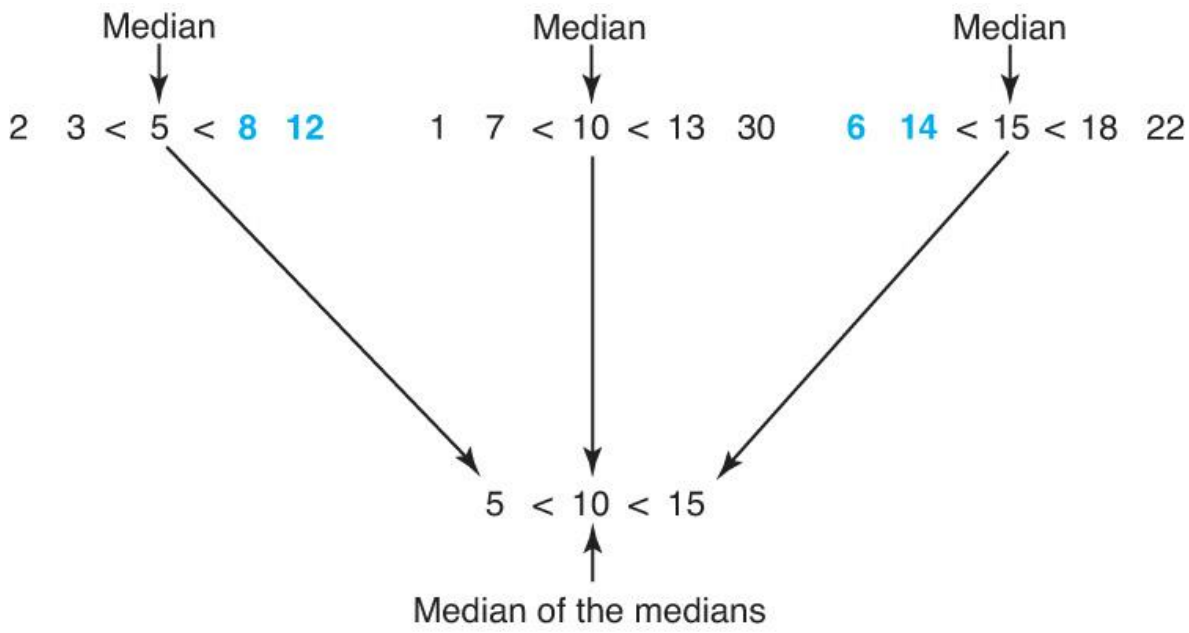
Worst-case Linear Time Algorithm for Finding the *k*th-Smallest Key via D&C

- An **O(n) worst-case time** algorithm for Finding the kth-Smallest Key?
- **Idea?**
  - **D&C-based!**
  - **Select a good pivot!**

Median of Medians: Finding the *k*th-Smallest Key

- **Idea?**
- **Use a good pivot!**
  - Divide n keys into n/5 **groups of 5** and find the **median** of each group.
  - **Recursively** determine the **median** of the n/5 **medians**.
  - **Partition using the median-of-medians as an approximate median!**

Median of the Medians of Five



Median-of-Medians: Finding the *k*th-Smallest Key

- Divide-and-Conquer
- Partition-based
- Similar to **QuickSelect**
- **A good pivot!**
- **QuickSelect with the median of medians pivot!**

- **Median of the Medians Select (MMSelect)**

Median-of-Medians: Finding the *k*th-Smallest Key

- **MMSelect(S, k):**
  - Group the array S into n/5 groups of size 5 and find the median of each group of size 5.
  - Recursively, find the true median of the n/5 medians. Call this P.
  - Use P as a pivot element.

A good pivot!

Median-of-Medians: Finding the *k*th-Smallest Key

- **MMSelect(S, k):**
  - Use a good pivot element P.
  - Partition S into subarrays LESS and GREATER as in Quicksort, i.e., LESS P GREATER.

  - If k = p, then output P.
  - If k < p, output **MMSelect(LESS, k)**
  - If k > p, output **MMSelect(GREATER, k – p)**

Median-of-Medians: Finding the *k*th-Smallest Key

- **Algorithm 8.6 General Selection Using the Median**
  - Divide-and-Conquer
  - Partition-based
  - Similar to **QuickSelect**
  - **A good pivot!**
  - **Median of the Medians Select algorithm**
  - **QuickSelect with the median of medians pivot!**

▶QUIZ? MMSelect

- Finding the kth-Smallest Key using **MMSelect**?
  - k= 4?
  - k= 8?
  - k= 12?

  - S= **[2, 3, 5, 8, 12, 1, 7, 10, 13, 30, 6, 14, 15, 18, 22]**

Median of Medians: Finding the *k*th-Smallest Key

- **S:** 
- **k**= 1, 2, …, n-1, n
- **p**= median of medians of five

Median of Medians: Finding the *k*th-Smallest Key

- Analysis of **Algorithm 8.6** General Selection Using the Median

T(n) = ???????? **?**

- **T(n)= O(n)** comparisons worst-case**!**

Median of Medians: Finding the *k*th-Smallest Key Worst-case

$(n - 1 - 2(n/5 - 1))/2$

$n/5 - 1$

$n/5 - 1$

$(n - 1 - 2(n/5 - 1))/2$

Median of Medians: Finding the *k*th-Smallest Key **Worst-case**

- 2**(n/5-1)** keys could be on either side of the median of medians!
- **½ (n-1-2(n/5-1))** keys are on one side of the median of medians!

- So, **at most ½ (n-1-2(n/5-1)) + 2(n/5-1)** = **7n/10-3/2** keys could be on one side of the median of medians!

Median of Medians: Finding the *k*th-Smallest Key

$$T(n) = T(7n/10 - 3/2) + T(n/5) + 6n/5 + n$$

Median of Medians: Finding the *k*th-Smallest Key

- $T(n) = T(7n/10 - 3/2) + T(n/5) + 6n/5 + n$
- $T(n) \approx T(7n/10) + T(n/5) + 11\,n/5$
- $T(n) = O(n)$
- **O(n)** comparisons **worst-case!**

Median of Medians: Finding the *k*th-Smallest Key

- **Why group size 5?**
  - 5 is the smallest odd number for a worst-case linear time selection algorithm.
  - **Any odd number >= 5** leads to a worst-case linear time selection algorithm!
  - When the group size is 3, O(n log n)!

▶QUIZ?

- Describe the **worst-case linear time** algorithm for Finding the kth-Smallest Key?

▶QUIZ?

- **Explain** that the worst-case time complexity algorithm for Finding the kth-Smallest Key is O(n)?

$$T(n) = T(7n/10 - 3/2) + T(n/5) + 6n/5 + n$$

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

▶QUIZ?

- Compare **QuickSelect** vs **Median-of-Medians-of-Five Select** Algorithm?

▶QUIZ?

- **NOW we have a O(n) time algorithm that finds median of an unsorted array.** Now consider a QuickSort implementation where we first find median using the above algorithm, then use median as pivot. What will be the **worst case time complexity** of this modified **QuickSort**?
  - O(n log n)

$$T(n) = O(1) \quad \text{if } n=1$$
$$T(n) = 2T(n/2) + O(n) \quad \text{if } n>1$$

▶QUIZ?

- Find the k-th smallest in an unsorted array of n items.
  - Using **Min-Heap**?
  - Using **Max-Heap**?

✓ Lower Bound of The Selection Problem

- Lower Bounds for the Selection Problems Only by Comparison of Keys?

Lower Bounds for Selection Only by Comparisons of Keys

- We prove a lower bound, **without going through all possible** selection algorithms?
- The lower bound for selection on an **unsorted** array is **Ω(n).**
  - **Proof by contradiction**

✓The Selection Problems & Algorithms Summary

- Keys are in an **unsorted** array: **Find the kth largest or smallest key?**
  - **Linear time O(n) average (expected)-case time** algorithm: QuickSelect
  - **Linear time O(n) worst-case time** algorithm: Median of Medians Select

- The **lower bound for selection** on an **unsorted** array is **Ω(n).**

Homework Assignment

Homework Assignment?

- What is the average-case time complexity of QuickSelect? Explain.
- What is the worst-case time complexity of the median-of-medians algorithm? Explain.

Homework Assignment?

- *Divide-and-Conquer*: Chapter 8 (Average-Case Linear Time Divide-and-Conquer-based Algorithm, i.e. QuickSelect Algorithm 8.5 for the k-th Smallest Selection Problem) & QuickSort using the median as the pivot (Chapter 2: Algorithm 2.6).

Test Cases

- S= [2, 3, 5, 8, 12, 1, 7, 10, 13, 30, 6, 14, 15, 18, 22]
  - k= 4?
  - k= 8?
  - k= 12?
- S= [2, 3, 5, 8, 12, 1, 7, 10, 13, 30, 6, 14, 15, 18, 22]

✓ Textbook Readings

- Chapter 8:
  - 8.5 (8.5.1, 8.5.2, 8.5.3 & 8.5.4 only)

the right majors, minors &

ucation?

or students' academic and career success

ging for many students - *Many students change their during college!*

e prediction of student success in MMC could

dual students

d their right MMC

chieve their academic goals

END

Prof. Young Park