CS4311
Design and Analysis of Algorithms

Lecture 8: Order Statistics
About this lecture

• Finding max, min in an unsorted array (upper bound and lower bound)

• Finding both max and min (upper bound)

• Selecting the $k^{th}$ smallest element

$k^{th}$ smallest element $\equiv k^{th}$ order statistics
Finding Maximum

in unsorted array
Finding Maximum (Method I)

• Let $S$ denote the input set of $n$ items
• To find the maximum of $S$, we can:
  Step 1: Set $\text{max} = \text{item 1}$
  Step 2: for $k = 2, 3, \ldots, n$
      | if (item $k$ is larger than $\text{max}$)
      |    Update $\text{max} = \text{item } k$
  Step 3: return $\text{max}$;

  # comparisons = $n - 1$
Finding Maximum (Method II)

- Define a function `Find-Max` as follows:

\[ \text{Find-Max}(R, k) \quad /* \text{R is a set with } k \text{ items */} \]

1. if \( k \leq 2 \) return maximum of \( R \);
2. Partition items of \( R \) into \( \lceil k/2 \rceil \) pairs;
3. Delete smaller item from \( R \) in each pair;
4. return \( \text{Find-Max}(R, k - \lfloor k/2 \rfloor) \);

Calling \( \text{Find-Max}(S, n) \) gives the maximum of \( S \)
Finding Maximum (Method II)

Let $T(n) = \# \text{ comparisons for Find-Max with problem size } n$

So, $T(n) = T(n - \lfloor n/2 \rfloor) + \lceil n/2 \rceil$ for $n \geq 3$

$T(2) = 1$

Solving the recurrence (by substitution), we get $T(n) = n - 1$
Lower Bound

Question: Can we find the maximum using fewer than $n - 1$ comparisons?

Answer: No! To ensure that an item $x$ is not the maximum, there must be at least one comparison in which $x$ is the smaller of the compared items.

So, we need to ensure $n-1$ items not max

$\Rightarrow$ at least $n - 1$ comparisons are needed
Finding Both Max and Min in unsorted array
Finding Both Max and Min

Can we find both max and min quickly?

Solution 1:

First, find max with \( n - 1 \) comparisons

Then, find min with \( n - 1 \) comparisons

\[ \Rightarrow \text{Total} = 2n - 2 \text{ comparisons} \]

Is there a better solution??
Finding Both Max and Min

Better Solution: (Case 1: if \( n \) is even)

First, partition items into \( n/2 \) pairs;

Next, compare items within each pair;

\( \bullet = \text{larger} \quad \circ = \text{smaller} \)
Finding Both Max and Min

Then, $\text{max} = \text{Find-Max in larger items}$
$\text{min} = \text{Find-Min in smaller items}$

$\# \text{ comparisons} = \frac{3n}{2} - 2$
Finding Both Max and Min

Better Solution: (Case 2: if \( n \) is odd)

We find \( \text{max} \) and \( \text{min} \) of first \( n - 1 \) items;
if (last item is larger than \( \text{max} \))
Update \( \text{max} = \) last item;
if (last item is smaller than \( \text{min} \))
Update \( \text{min} = \) last item;

\[ \# \text{ comparisons} = \frac{3(n-1)}{2} \]
Finding Both Max and Min

Conclusion:
To find both $\text{max}$ and $\text{min}$:

- if $n$ is odd: $\frac{3(n-1)}{2}$ comparisons
- if $n$ is even: $\frac{3n}{2} - 2$ comparisons

Combining: at most $3\left\lfloor \frac{n}{2} \right\rfloor$ comparisons

$\Rightarrow$ better than finding $\text{max}$ and $\text{min}$ separately
Lower Bound

Textbook Ex 9.1-2 (Very challenging):

• Show that we need at least 
  \[\lceil \frac{3n}{2} \rceil - 2\] comparisons
  to find both \textbf{max} and \textbf{min} in worst-case

Hint: Consider how many numbers may be max or min (or both). Investigate how a comparison affects these counts
Selecting $k^{th}$ smallest item in unsorted array
Selection in Linear Time

• In next slides, we describe a recursive call $\text{Select}(S,k)$ which supports finding the $k^{th}$ smallest element in $S$

• Recursion is used for two purposes:
  (1) selecting a good pivot (as in Quicksort)
  (2) solving a smaller sub-problem
Select($S$, $k$)

/* First, find a good pivot */
1. Partition $S$ into $\left\lfloor |S|/5 \right\rfloor$ groups, each group has five items (one group may have fewer items);
2. Sort each group separately;
3. Collect median of each group into $S'$;
4. Find median $m$ of $S'$:
   
   $m = \text{Select}(S', \left\lfloor \left\lfloor |S|/5 \right\rfloor / 2 \right\rfloor)$;
4. Let $q = \# \text{ items of } S \text{ smaller than } m$;
5. If $(k == q + 1)$
   \[\text{return } m;\]
   /* Partition with pivot */
6. Else partition $S$ into $X$ and $Y$
   \[X = \{\text{items smaller than } m\}\]
   \[Y = \{\text{items larger than } m\}\]
   /* Next, form a sub-problem */
7. If $(k < q + 1)$
   \[\text{return Select}(X, k)\]
8. Else
   \[\text{return Select}(Y, k-(q+1));\]
Selection in Linear Time

Questions:

1. Why is the previous algorithm correct? (Prove by Induction)

2. What is its running time?
Running Time

• In our selection algorithm, we chose \( m \), which is the median of medians, to be a pivot and partition \( S \) into two sets \( X \) and \( Y \).

• In fact, if we choose any other item as the pivot, the algorithm is still correct.

• Why don’t we just pick an arbitrary pivot so that we can save some time??
Running Time

• A closer look reviews that the worst-case running time depends on \(|X|\) and \(|Y|\)

• Precisely, if \(T(|S|)\) denote the worst-case running time of the algorithm on \(S\), then

\[
T(|S|) = T(\lceil |S|/5 \rceil) + \Theta(|S|) + \max \{T(|X|), T(|Y|) \}
\]
Running Time

- Later, we show that if we choose $m$, the “median of medians”, as the pivot,
  
  both $|X|$ and $|Y|$ will be at most $3|S|/4$

- Consequently,

  $$T(n) = T\left(\left\lfloor n/5 \right\rfloor \right) + \Theta(n) + T\left(3n/4\right)$$

  $$\Rightarrow T(n) = \Theta(n) \quad \text{(obtained by substitution)}$$
Median of Medians

• Let’s begin with \( \lceil n/5 \rceil \) sorted groups, each has 5 items (one group may have fewer)
Median of Medians

- Then, we obtain the median of medians, $m$.
Median of Medians

Then, we know that all items marked with $X$ have value at most $m$.

Groups with median smaller than $m$

$\star = m$  $X = \text{“value } \leq m\text{”}$
Median of Medians

The number of items with value at most $m$ is at least

$$3\left\lceil \frac{n}{5} \right\rceil / 2 - 1 - 2$$

- each full group has 3 'crossed' items
- min # of groups
- one group may have only 1 'crossed' item

$\Rightarrow$ number of items: at least $3n/10 - 5$
Median of Medians

Previous page implies that at most

\[ \frac{7n}{10} + 5 \] items

are greater than \( m \)

\[ \Rightarrow \text{For large enough } n \text{ (say, } n \geq 100) \]

\[ \frac{7n}{10} + 5 \leq \frac{3n}{4} \]

\[ \Rightarrow |Y| \text{ is at most } \frac{3n}{4} \text{ for large enough } n \]
Median of Medians

Similarly, we can show that at most
\[ \frac{7n}{10} + 5 \text{ items are smaller than } m \]
\[ |X| \text{ is at most } \frac{3n}{4} \text{ for large enough } n \]

Conclusion:
The “median of medians” helps us control the worst-case size of the sub-problem
\[ \Rightarrow \text{ without it, the algorithm runs in } \Theta(n^2) \text{ time in the worst-case} \]