Lecture 4: Order Statistics

Instructor: Saravanan Thirumuruganathan

Outline

1. Order Statistics
    - Min, Max
    - k^{th}-smallest and largest
    - Median
    - Mode and Majority

In-Class Quizzes

- **URL:** `http://m.socrative.com/`
- **Room Name: 4f2bb99e**

Order Statistics

- i^{th} Order Statistic of a set of n elements is the i^{th} smallest element
- **Selection** Problem
  - **Input:** A set A of n (distinct) numbers and an integer i with $1 \leq i \leq n$
  - **Output:** i^{th} smallest element in A
    - The element $x \in A$ that is larger than exactly $i - 1$ other elements of A
    - Select element with **rank** i

Popular Order Statistics

- $i = 1$
- $i = n$
- $i = \lfloor \frac{n+1}{2} \rfloor$ and $i = \lceil \frac{n+1}{2} \rceil$

Popular Order Statistics

- Minimum: $i = 1$
- Maximum: $i = n$
- Median: $i = \lfloor \frac{n+1}{2} \rfloor$ (lower) and $i = \lceil \frac{n+1}{2} \rceil$ (upper)

Selection Problem

- **Input:** A set A of n (distinct) numbers and an integer i with $1 \leq i \leq n$
- **Output:** i^{th} smallest element in A
- Naive Solution?

Selection Problem

- **Input:** A set A of n (distinct) numbers and an integer i with $1 \leq i \leq n$
- **Output:** i^{th} smallest element in A
- Naive Solution?
  - Sort A and pick $A[i]$
  - Time Complexity: $O(n \log n)$

Finding the Minimum

Finding the Minimum

```
Minimum(A):
    min = A[1]
    for i = 2 to A.length
        if min > A[i]
            min = A[i]
    return min
```

Analysis:

Finding the Minimum

```
Minimum(A):
    min = A[1]
    for i = 2 to A.length
        if min > A[i]
            min = A[i]
    return min
```

Analysis:

- Complexity Measure: Number of Comparisons

Finding the Minimum

```
Minimum(A):
    min = A[1]
    for i = 2 to A.length
        if min > A[i]
            min = A[i]
    return min
```

Analysis:

- Complexity Measure: Number of Comparisons
- Number of Comparisons: $n - 1$
- Time Complexity: $O(n)$

Finding the Maximum

```
Maximum(A):
    max = A[1]
    for i = 2 to A.length
        if max < A[i]
            max = A[i]
    return max
```

Analysis:

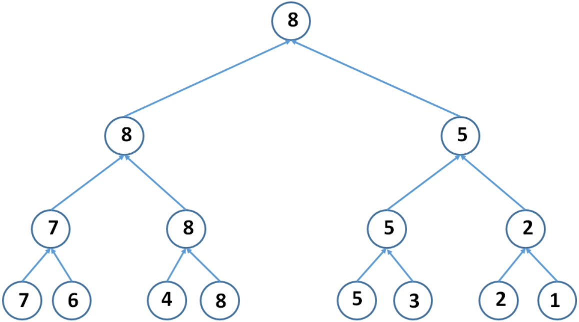
- Complexity Measure: Number of Comparisons
- Number of Comparisons: $n - 1$
- Time Complexity: $O(n)$

Recursive Maximum

Idea: Use Divide and Conquer to find Maximum

Recursive Maximum

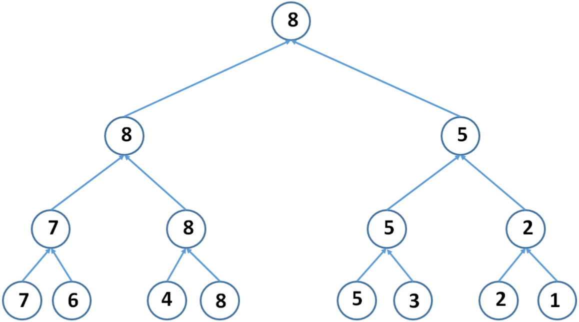
Idea: Use Divide and Conquer to find Maximum



Analysis:

Recursive Maximum

Idea: Use Divide and Conquer to find Maximum



Analysis:

- Recurrence Relation: $T(n) = 2T(\frac{n}{2}) + 1 = O(n)$
- Number of Comparisons: $n - 1$ (Intuition)

Simultaneous Maximum and Minimum

Aim: Find the maximum and minimum of array A

Simultaneous Maximum and Minimum

Aim: Find the maximum and minimum of array A

```
Minimum-Maximum(A):
    min = Minimum(A)
    max = Maximum(A)
    return min, max
```

Analysis:

Simultaneous Maximum and Minimum

Aim: Find the maximum and minimum of array A

```
Minimum-Maximum(A):
    min = Minimum(A)
    max = Maximum(A)
    return min, max
```

Analysis:

- Number of Comparisons: $(n-1)+(n-1)=2n-2$

Simultaneous Maximum and Minimum

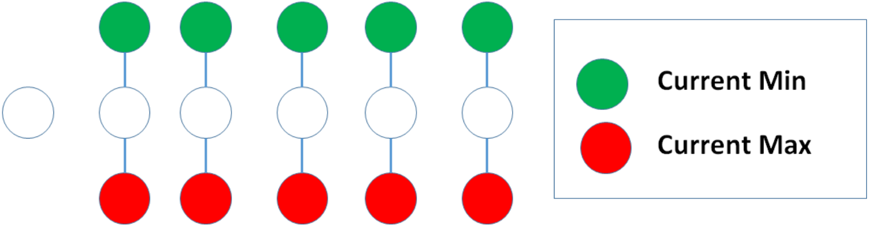
Aim: Find the maximum and minimum of array A

```
Minimum-Maximum(A):
    min = Minimum(A)
    max = Maximum(A)
    return min, max
```

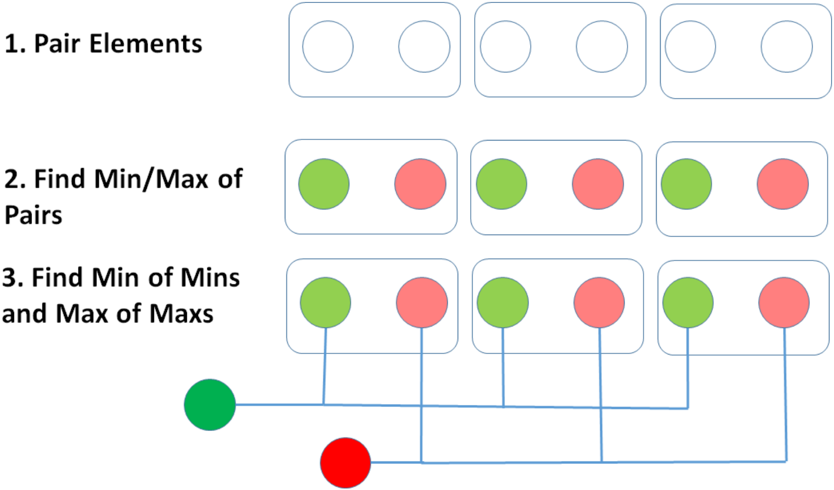
Analysis:

- Number of Comparisons: $(n-1)+(n-1)=2n-2$
- Slightly better: $(n-1)+(n-2)=2n-3$ (for e.g., by swapping min with first element of array)

Simultaneous Maximum and Minimum - Visualization



Simultaneous Maximum and Minimum - Better Algorithm



Simultaneous Maximum and Minimum

Analysis:

Simultaneous Maximum and Minimum

Analysis:

- Number of Comparisons (approximate): Pairwise + Min of Mins + Max of Maxs

$$(\frac{n}{2}) + (\frac{n}{2}) + (\frac{n}{2}) = \frac{3n}{2}$$

Finding Second Largest Element - Naive Method

Finding Second Largest Element - Naive Method

```
Find-Second-Largest(A):
    max = Maximum(A)
    Swap A[n] with max
    secondMax = Maximum(A[1:n-1])
    return secondMax
```

Analysis:

Finding Second Largest Element - Naive Method

```
Find-Second-Largest(A):
    max = Maximum(A)
    Swap A[n] with max
    secondMax = Maximum(A[1:n-1])
    return secondMax
```

Analysis:

- $n-1$: for finding maximum
- $n-2$: for finding 2nd maximum
- $2n-3$: total

Finding Second Largest Element - Tournament Method

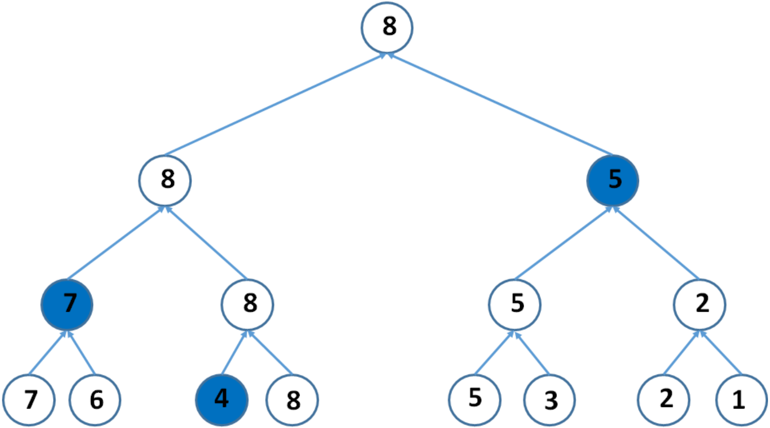
Finding Second Largest Element - Tournament Method

Observation:

- In a tournament, second best person could have only be defeated by the best person.
- It is not necessarily the other element in the final "match"

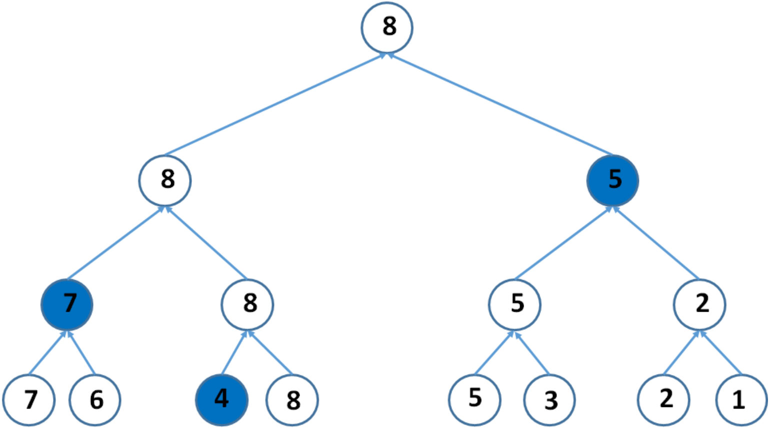
```
Find-Second-Largest(A):
    max = Recursive-Maximum(A)
    candidates = list of all elements of A that were
                    directly compared with max
    secondMax = Maximum(candidates)
    return secondMax
```

Finding Second Largest Element - Tournament Method



Analysis:

Finding Second Largest Element - Tournament Method



Analysis:
Number of Comparisons: $(n-1) + (\lceil \lg n \rceil - 1) = n + \lceil \lg n \rceil - 2$

Selection Problem

- **Input:** A set A of n (distinct) numbers and an integer i with $1 \leq i \leq n$
- **Output:** i^{th} smallest element in A
- Naive Solution?
  - Sort A and pick $A[i]$
  - Time Complexity: $O(n \log n)$
- **Surprising Result**: Can be solved in $O(n)$ time!

QuickSelect

9	3	6	7	10	8	2	5	1	4

Partition

3	6	2	5	1	4	7	9	10	8

QuickSelect

9	3	6	7	10	8	2	5	1	4

$\Downarrow$ **Partition**

3	6	7	8	2	5	1	4	9	10

QuickSelect

9	3	6	7	10	8	2	5	1	4

Partition

2	1	3	9	6	7	10	8	5	4

QuickSelect PseudoCode

```
Randomized-Select(A, p, r, i)
    if p == r:
        return A[p]
    q = Randomized-Partition(A, p, r)
    k = q - p + 1
    if i == k
        return A[q]
    elseif i < k
        return Randomized-Select(A, p, q-1, i)
    else
        return Randomized-Select(A, q+1, r, i-k)
```

Radix Sort - LSD Idea

GiG

Summary

Major Concepts:

- Concept of Lower bounds
- Lower bounds for Comparison based Sorting Algorithms
- Decision tree model for Complexity Analysis
- Linear Time Sorting Algorithms