Searching and Sorting

- Linear and Binary Search
- Order Statistics - MIN and MAX
- Comparison Based Sorting Algorithms
  - Selection Sort
  - Bubble Sort
  - Insertion Sort
- Lower Bound on Comparison based sorting
- Non-Comparison Based Sorting - Integers Sorting
  - Counting Sort
  - Radix Sort

IMDAD ULLAH KHAN

Order Statisitcs

Order statistics are used to summarize data by a single element

The i^{th} order statistic is the i^{th} smallest element

element at index i when the data is sorted

▷ Assuming data is numerical (has a total order)

- Minimum: 1st order statistic
- Maximum: nth order statistic
- Median: $\lfloor \frac{n+1}{2} \rfloor$th order statistic ▷ ($n$: odd/even)
- quartiles, deciles, percentiles are all order statistics

Order statistics have many applications and are closely related to sorting

Find Minimum Element

Input: An array A of n distinct numbers
Output: The smallest number $x \in A$ and its index

Algorithm FINDMIN(A)

```
min ← A[1]                          ▷ A[1] is minimum of A[1···1]
for i = 2 to n do
    if A[i] < min then
        min ← A[i]                  ▷ Update min if A[i] is smaller
```

- Correctness is proved using *'loop invariant'*

  Prove by induction on i that

  After i^{th} iteration $min =$ **minimum of** $A[1 \dots i]$

- Runtime is $n - 1$ comparisons
- This is the best we can do

Lower Bound on Finding Min Element

This is the best we can do

Any comparison based FINDMIN algorithm needs $\Omega(n)$ comparisons

- Initially every element of A is a candidate to be *min*
- A comparison between two elements makes a winner and a loser
- Except for min every element must have won at least one comparison

  otherwise it may still be the *min*
- One comparison produces at most one winner

  $\triangleright$ reduces the number of candidates by at most one
- For $n-1$ candidates elimination at least $n-1$ comparisons are needed

Find Min and Max

Input: An array A of n distinct numbers
Output: The smallest and largest numbers x and y in A and their indices

- First run $\textsc{FindMin}(A)$, then run $\textsc{FindMax}(A)$

  ▷ Takes $2n - 2$ comparisons ($O(n)$)

- Cannot do asymptotically better, but can improve the constant

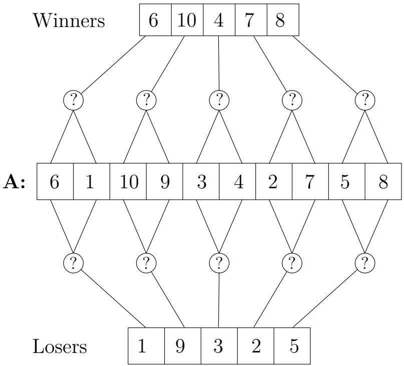
Find Min and Max: A better algorithm

Compare successive pairs in A and make two new arrays

- Losers: smaller in each comparison
- Winners: bigger in each comparison

Find Min in the Losers array

Find Max in the Winners array



Runtime: $n/2 + n/2 - 1 + n/2 - 1 = 3n/2 - 2$ (matches lower bound)

▷ called the tournament style algorithm

Can easility be done without the auxiliary arrays

Find Max and Second Max

Input: An array A of n distinct numbers
Output: The largest and the second largest numbers x and y in A

Algorithm $\textsc{FindMax2ndMax}(A)$

$x \leftarrow \textsc{FindMax}(A)$ ▷ $n - 1$ comparisons
$A \leftarrow A \setminus \{x\}$ ▷ 0 comparisons
$y \leftarrow \textsc{FindMax}(A)$ ▷ $n - 2$ comparisons

Total runtime: $(2n - 3)$ comparisons

Find Max and Second Max

Input: An array A of n distinct numbers
Output: The largest and the second largest numbers x and y in A

A Tournament Style Algorithm

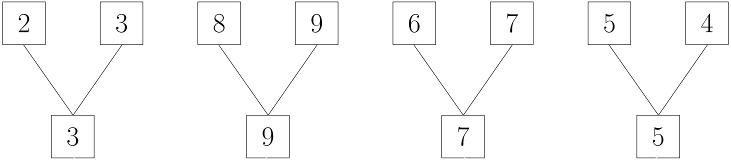
- Make Losers and Winners array ▷ $n/2$ comp
  ▷ x will be in Winners array
  ▷ y would win against every element except x
  ▷ y could be in the Losers or Winners array
- Find max and second max of the Winners array ▷ $(2n/2 - 3)$ comp
- Find max of Losers array ▷ $(n/2 - 1)$ comp
- y is the larger of second max of Winners and max of Losers

Total number of comparisons is still $\frac{n}{2} + \frac{2n}{2} - 3 + \frac{n}{2} - 1 \simeq 2n - 3$

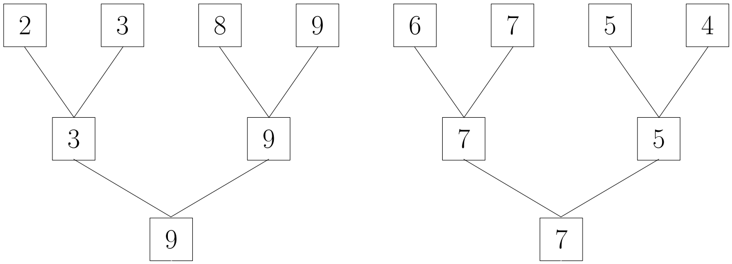
Find Max and Second Max: A better algorithm

2	3	8	9	6	7	5	4

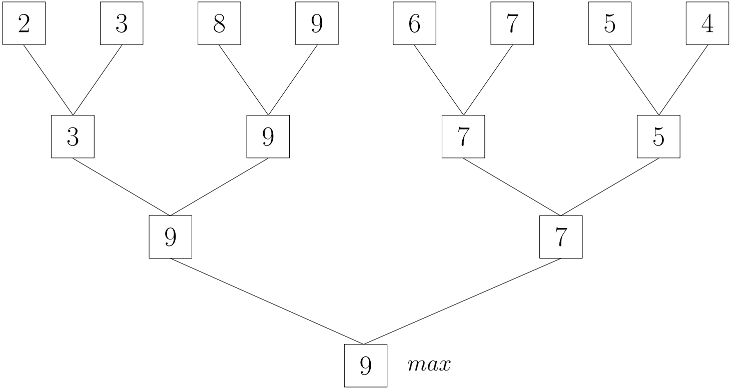
Find Max and Second Max: A better algorithm



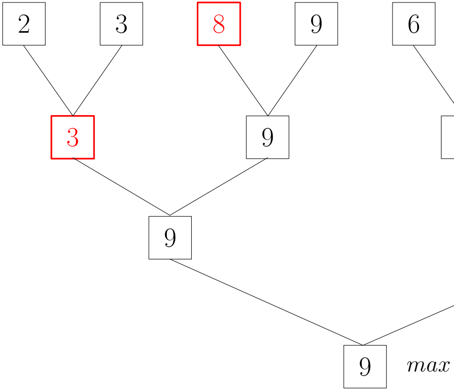
Find Max and Second Max: A better algorithm



Find Max and Second Max: A better algorithm



Find Max and Second Max: A better algorithm



- Follow the comparison trail of the max
- Leads to an algorithm taking $n + \log n$ comparisons
- This is also the lower bound

Order Statistics: Summary

- Order statistics are important summaries of data, related to sorting
- Minimum (and analogously maximum) can be found in linear time
- Tournament style algorithm finds maximum and minimum in $3n/2$
- Find maximum and second maximum can be found in $O(n + \log n)$ using the comparison trail