# Medians and Order Statistics

By
Dr. Subhashis Majumder
Professor and HoD, CSE,
Dean UG Programme, HIT

# What is ith order statistic?

- It is the ith smallest element in a set of n elements (which are comparable)
- Minimum – $1^{st}$ order statistic
- Maximum – nth order statistic
- Median – half way element
- Median is unique when n is odd, $i = (n + 1)/2$
- Two medians exist when n is even, $i = n/2$ and $i = (n)/2 + 1$
- Normally median refers to the lower median irrespective of parity, where $i = \lfloor (n + 1)/2 \rfloor$

# Selection Problem

- Statement –
- Given a set A of n (distinct) numbers and i, $1 \leq i \leq n$.
- Return $x \in A$, ∋ it is exactly larger than $i - 1$ other elements of A.
- Naïve solution – Sort in $O(n \lg n)$ time and then return the ith element.
- However, **better** solutions **exist.**

# Finding the minimum (or maximum) of n elements

- Comparisons required is $n - 1$.
- Minimum(A)

```
min ← A[1]
for i ← 2 to length(A)
    do if min > A[i]
        then min ← A[i]
return min
```

# Finding Minimum & Maximum simultaneously

- number of comparisons required $\leq 2(n - 1)$
- Is it obvious? Will you call it an upper bound?
- Can you improve the bound further, or rather can you lower the upper bound to make more tight?

# Better Strategy

- Suppose **min** is your current minimum and **max** is your current maximum
- Take a pair of elements from the input set and compare with each other.
- Now compare the smaller one with **min** to decide the new value of **min**
- Similarly compare the larger one with **max** ….
- So need 3 comparisons for every 2 elements

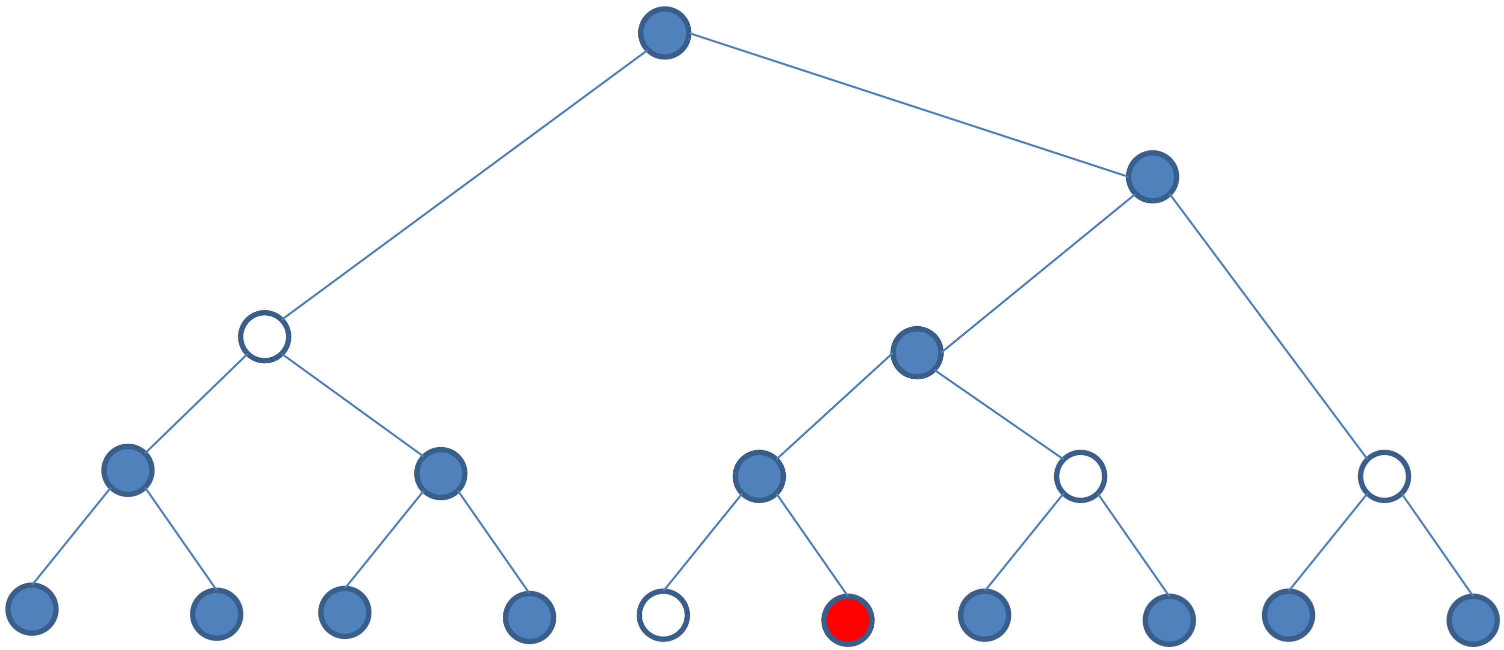
# Details

- Case 1 – n is odd
  - Assign the 1st element to both **min** and **max**
  - Then we need $3 \lfloor n/2 \rfloor$ comparisons to process the rest of the elements
- Case 2 – n is even
  - Compare the first pair and assign the **min** & **max**
  - Then perform $3 (n - 2)/2$ comparisons to process the rest
  - Total is $1 + 3 n/2 - 3 = 3 n/2 - 2$
- So we can say total number of comparisons in either case is **at most** $3 \lfloor n/2 \rfloor$

# Find 2nd smallest element

- Naïve solution: $(n - 1) + (n - 2)$ comparisons
- Show that the 2nd smallest of n elements can be found with $\mathbf{n + \lceil lgn \rceil - 2}$ comparisons in the worst-case.
- For determining the smallest element we obviosly need $n - 1$ comparisons
- Consider this as a knock-out tournament of some game, where the smaller element always wins.
- Question – Who are the possible candidates for becoming the runner-up (i.e. 2nd smallest)?
- Hint – The runner-up can lose only to the winner

- Answer – The candidates are those elements who lost a direct game against the winner.
- 2nd question – What is the number of such candidates?



- Answer – $\lceil \lg n \rceil$ direct losers in the worst-case
- In the figure, n = 10, $\lceil \lg n \rceil = 4$.
- Now, to determine the winner out of them will take $\lceil \lg n \rceil - 1$ comparisons.
- Hence total number of comparisons = $(n - 1) + (\lceil \lg n \rceil - 1) = n + \lceil \lg n \rceil - 2$

# Lower Bound Result

- Problem – Show that $\lceil 3n/2 \rceil - 2$ comparisons are necessary for finding the minimum and maximum of n numbers in the worst-case.
- Answer –
  - Observe that initially all the n elements are in contention each for the maximum or minimum position and we have to eliminate n – 1 elements from each of these two knock-out tournaments.
  - Now a comparison between two 'fresh' elements (which did not take part in any such comparison before) will eliminate 1 element each from the 2 tournaments.
  - However if any of the elements is not fresh, then a comparison either eliminates one element from the max-tournament or one from the min-tournament.

- Note that if you are lucky, you may strike two elements out – eg. If a fresh element is found smaller than a used element which is already out of the max-tournament
- However if the fresh element turns out to be greater than the used element in the above comparison, then only job done is – fresh element out of the min-tournament
- If both elements are used, then either both of them will be in max-tour. or both in min-tour. and hence only one will be eliminated
- Hence in the worst-case if both elements are not fresh, number of eliminations will be only 1.

- Now, maximum number of comparisons possible where both elements are fresh is $\lfloor n/2 \rfloor$, which will contribute to exactly 2 $\lfloor n/2 \rfloor$ eliminations
- Hence number of eliminations still required is $2(n – 1) - 2 \lfloor n/2 \rfloor$
- Hence in the worst-case number of comparisons that is necessary to finish the job is $\lfloor n/2 \rfloor + 2(n – 1) - 2 \lfloor n/2 \rfloor = 2(n - \lfloor n/2 \rfloor) + \lfloor n/2 \rfloor - 2 = 2 \lceil n/2 \rceil + \lfloor n/2 \rfloor - 2 = n + \lceil n/2 \rceil - 2 = \lceil 3n/2 \rceil - 2$ (Since $\lfloor n/2 \rfloor + \lceil n/2 \rceil = n$)

# Selection in Expected Linear Time

- Randomized_Select works on only one side of the partition. It returns the ith smallest element from the array A[p…r]
- Procedure Randomized_Select(A, p, r, i)

```
if p = r
    then return A[p]
q ← Randomized_Partition(A, p, r)
K ← q – p + 1
if i = k   // lucky – pivot value is the answer
    then return A[q]
elseif i < k
    then return Randomized_Select(A, p, q - 1, i)
else
    return Randomized_Select(A, q + 1, r, i - k)
```

# Complexity Analysis

- Time required by Randomized_Select on an input array of n elements is a random variable that we denote by $T(n)$. Let us calculate an upper bound on $E(T(n))$.
- Now, Randomized_Partition is equally likely to return any element as pivot. So, for each k, $1 \leq k \leq n$, the subarray A[p…q] has k elements ($\leq$ pivot) with probability 1/n.
- For k = 1, 2, …, n, we define indicator random variables $x_k$ , where $x_k = 1$ if the subarray A[p…q] has exactly k elements, so $E(x_k) = 1/n$

- $\mathrm{T(n)} \leq \sum_{k=1}^{n} \mathrm{x_k T}(\max(k-1, n-k)) + O(n)$
- $\mathrm{E(T}(n)) \leq E(\sum_{k=1}^{n} \mathrm{x_k T}(\max(k-1, n-k)) + O(n))$

  $= \sum_{k=1}^{n} \mathrm{E}(\mathrm{x_k T}(\max(k-1, n-k))) + O(n)$

  (by linearity of expectation)

  $= \sum_{k=1}^{n} \mathrm{E}(\mathrm{x_k})(\mathrm{E}(\mathrm{T}(\max(k-1, n-k)))) + O(n)$

  (by independence)

  $= \sum_{k=1}^{n} \ 1/\mathrm{n}\ (\mathrm{E}(\mathrm{T}(\max(k-1, n-k)))) + O(n)$
- max(k – 1, n – k) = k – 1 if k > $\lceil n/2 \rceil$

  n – k if k ≤ $\lceil n/2 \rceil$
- If n is even, each term from from T($\lceil n/2 \rceil$ ) to T(n-1) appears exactly twice
- If n is odd, each of those terms appear twice and additionally T($\lfloor n/2 \rfloor$) appears once

- $\mathrm{E}(\mathrm{T}(n)) \leq 2/n \sum_{k=\lfloor n/2 \rfloor}^{n-1} \mathrm{E}(\mathrm{T}(\mathrm{k})) + O(n)$
- We solve the above recurrence relation using the method of substitution.
- Assume T(n) ≤ cn for some constant c and T(n) = O(1) for n less than some constant
- $\mathrm{E}(\mathrm{T}(n)) \leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} \mathrm{ck} + an$
- $\mathrm{E}(\mathrm{T}(n)) \leq \frac{2c}{n} (\sum_{k=1}^{n-1} \mathrm{k} - \sum_{k=1}^{\lfloor n/2 \rfloor -1} \mathrm{k}) + an$

$\leq \frac{2c}{n}$ [(n(n-1)/2) – n/4(n/2 – 1) + an = c(n-1) – c(n-2)/4

(Wrong logic)

= cn – c – cn/4 + c/2 + an = cn – (cn/4 + c/2 – an) ≤ cn

choose c and a such that (cn/4 + c/2 – an) > 0

- Hence linear.

# Correct Analysis

- $\sum_{k=1}^{\lfloor n/2 \rfloor - 1} k = \frac{1}{2} (\lfloor n/2 \rfloor - 1) (\lfloor n/2 \rfloor)$

Now, $\lfloor n/2 \rfloor > (n/2) - 1$, so $(\lfloor n/2 \rfloor - 1) > (n/2) - 2$

So $\frac{1}{2} (\lfloor n/2 \rfloor - 1) (\lfloor n/2 \rfloor) > \frac{1}{2} (n/2 - 2) (n/2 - 1)$

Hence, we can write

$- \sum_{k=1}^{\lfloor n/2 \rfloor - 1} k = - \frac{1}{2} (\lfloor n/2 \rfloor - 1) (\lfloor n/2 \rfloor)$

$< - \frac{1}{2} (n/2 - 2) (n/2 - 1)$

# Selection of any order statistic (in particular median) is indeed linear

- $\mathrm{E}(\mathrm{T}(n)) \leq \frac{2c}{n}(\sum_{k=1}^{n-1} \mathrm{k} - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} \mathrm{k}) + an$

$\leq \frac{2c}{n}$ [(n(n-1)/2) - ½ (n/2 – 2) (n/2 – 1)] + an

$= \frac{2c}{n}$ [½($n^2$ - n) - ½ ($n^2$ /4 - 3n/2 + 2)] + an

$= \frac{2c}{n}[\frac{3n^2}{8} + \frac{n}{4} - 1] + an = c(\frac{3n}{4} + \frac{1}{2} - \frac{2}{n}) + an$

$\leq$ 3cn/4 + c/2 + an = cn – (cn/4 – c/2 – an) $\leq$ cn

if we can choose c large enough so that (cn/4 – c/2 – an) ≥ 0. Choose c > 4a & it holds for n > 2c/(c–4a)