SCALER

Heaps - 2

TABLE OF CONTENTS

Notes

Sort the array

	TC	SC
Merge Sort →	$O(N \log(N))$	$O(N)$
Quick Sort →	$O(N \log(N)) \rightarrow O(N^2)$	$O(\log(N)) \rightarrow O(N)$

Heap - Sort

Approach 1 → 1) Build Heap.

2) Extract min / max & insert in answer → N times.

$TC = O(N + N\log(N)) = O(N\log(N))$

$SC = O(N)$ // Heap

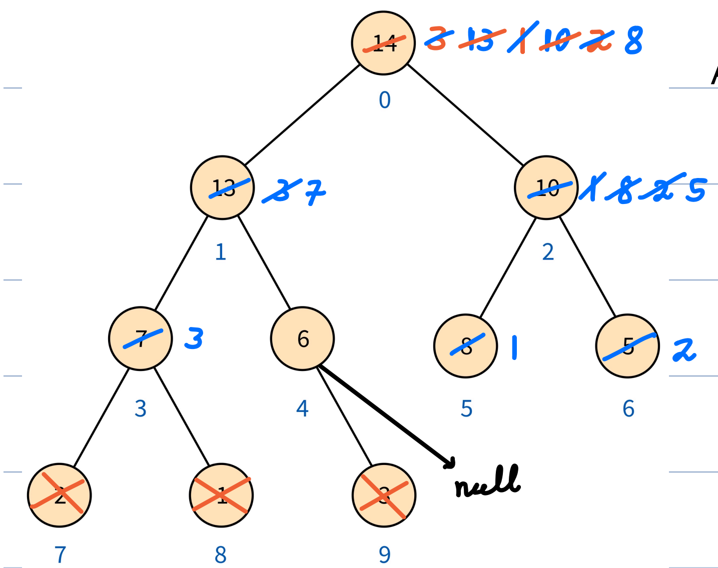
$O(1)$

Approach 2 →

Array → Heap → Sorted Array

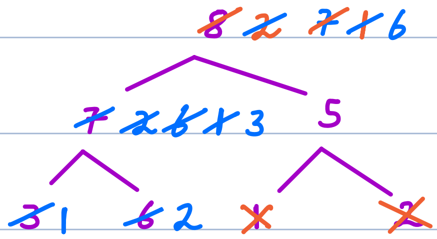
(Max Heap)

max Element



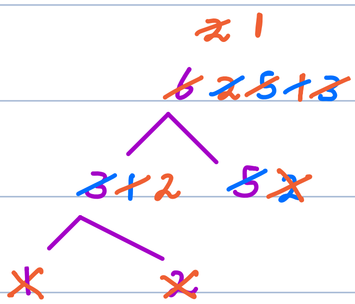
	0	1	2	3	4	5	6	7	8	9
A =	[~~14~~	~~13~~	~~10~~	~~7~~	6	~~8~~	~~5~~	~~2~~	~~1~~	~~3~~]
	~~3~~	~~3~~	~~1~~	3		1	2	10	13	14
	~~13~~	7	~~8~~							
	~~1~~		~~2~~							
	~~10~~		5							
	~~2~~									
	8									

Extract max & store it at last index.



	0	1	2	3	4	5	6	7	8	9
A = [	~~8~~	~~7~~	5	~~3~~	~~6~~	~~1~~	~~2~~	10	13	14]
	~~2~~	~~2~~		1	2	7	8			
	~~7~~	~~6~~								
	~~1~~	~~1~~								
	6	3								

↓



	0	1	2	3	4	5	6	7	8	9
A = [	~~6~~	~~3~~	~~5~~	~~1~~	~~2~~	7	8	10	13	14]
	~~2~~	~~1~~	~~2~~	5	6					
	~~5~~	2	3							
	~~1~~									
	~~3~~									
	~~2~~									
	1									

↓

	0	1	2	3	4	5	6	7	8	9
A = [	1	2	3	5	6	7	8	10	13	14]

Heap Sort

TC = $O(N \log(N))$

SC = $O(1)$

→ **Not Stable** (relative order of equal index).

< Question > : Given arr[N]. Find Kth largest element.

(Ans)

8	5	1	2	4	9	7
0	1	2	3	4	5	6

arr[ ] → [K = 3]

(Ans)

4 3 2 1

Quiz : arr[ ] → [K = 5]

1	2	3	4	5
0	1	2	3	4

Sol 1 → sort the array & ans = A[N-K]

TC = O(N log(N))

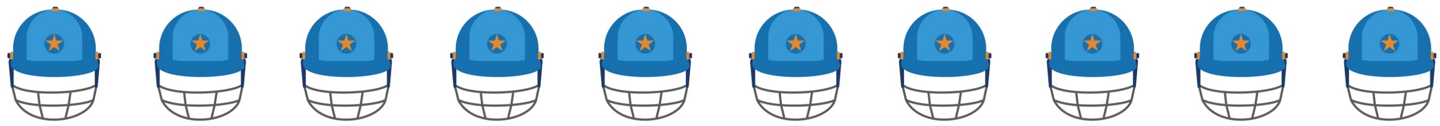
Sol 2 → 1) Build max heap.

2) Extract max K times.

TC = O(N + K log(N)) SC = O(1)

Team Selection

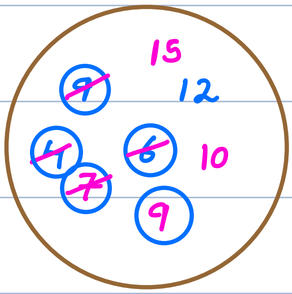
- Select 4 best batsman



B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
9	12	4	6	7	5	10	9	8	15
✓	✓	✓	✓	↑	↑	↑	↑	↑ x	↑

ignore

Keep track of top 4 players in a min heap.



if (cur > root of min heap) {
    extractMin()
    insert (cur)
} else → Nothing to do

minHeap size K

Ans = <u>root of minHeap of size K.</u>

TC = $\underline{O(K + (N-K)\log(K))}$

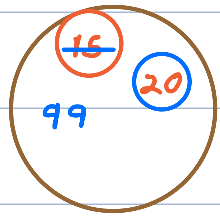
SC = $\underline{O(K)}$

< Question > : Kth largest element for all the windows starting from 0th idx.

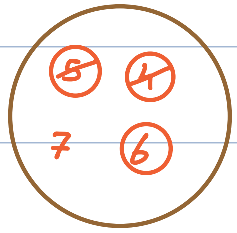
min Heap

	0	1	2	3
A =	15	20	99	1
ans →	-1	15	20	20

K = 2



min Heap



	0	1	2	3	4
Example →	5	4	1	6	7
ans →	-1	4	4	5	6

K = 2

```
if (cur > root of min heap) {
    extractMin()
    insert (cur)
} else → Nothing to do
ans = root of min Heap
```

$TC = \underline{O(K + (N-K) \log(K))}$

$SC = \underline{O(K)}$

Flipkart

Flipkart is currently dealing with the difficulty of precisely estimating and displaying the expected delivery time for orders to a specific pin code.
The existing method relies on historical delivery time data for that pin code, using the median value as the expected delivery time.

As the order history expands with new entries, Flipkart aims to enhance this process by dynamically updating the expected delivery time whenever a new delivery time is added.

The objective is to find the expected delivery time after each new element is incorporated into the list of delivery times.

End Goal

With every addition of new delivery time, requirement is to find the median value.

Why Median ?

The median is calculated because it provides a more robust measure of the expected delivery time.

The median is less sensitive to outliers or extreme values than the mean. In the context of delivery times, this is crucial because occasional delays or unusually fast deliveries (outliers) can skew the mean significantly, leading to inaccurate estimations.

< Question > : Given an infinite stream of integers. Find the median of current set of elements.

middle element in sorted order

[1 5 10 12 15]

[2 8 10 20 60 62]

smaller mid

6 → 6

6, 3 → 3

6, 3, 8 → [3 6 8] → 6

6, 3, 8, 11 → [3 6 8 11] → 6

[1 2 4 3] → [1 2 3 4]

2.5 (Ans)

4

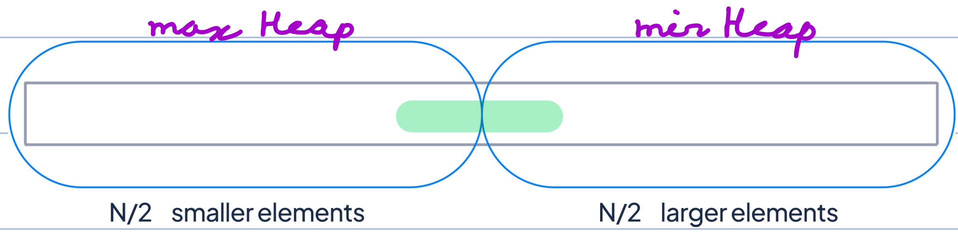
6 3 8 11 → 3 6 8 11

3 $\boxed{4 \quad 6 \quad 8}$ 11

___ . . . . 80 | 100 . . . . ___

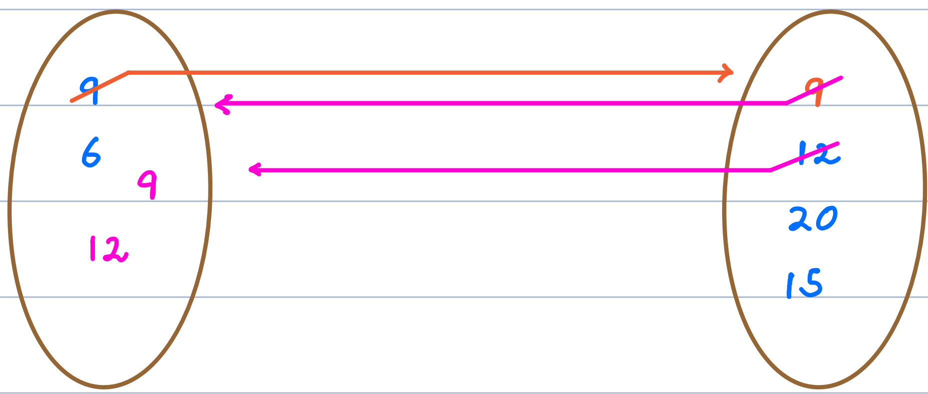
small elements — large element

max Heap — min Heap



Example : 9 6 12 20 15 . . .

ans → 9 6 9 9 12 . . .



max Heap

min Heap

Ans = root of max Heap

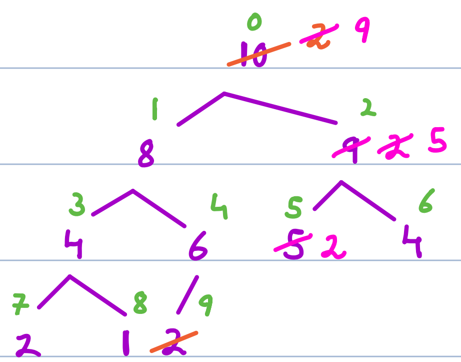
$\boxed{\text{(size of maxHeap - size of minHeap)} = \{0, 1\}}$

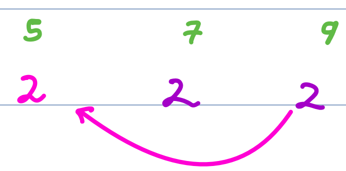
```
∀ intake x,
if ( maxHeap.isEmpty() || x <= root of maxHeap) {
    insertMaxHeap(x)
    if ( maxHeapSize - minHeapSize > 1)
        insertMinHeap( extractMaxHeap())
} else {
    insertMinHeap (x)
    if ( maxHeapSize - minHeapSize < 0)
        insertMaxHeap ( extractMinHeap())
}

ans = rootMaxHeap
```

TC = $\underline{O(N \log(N))}$

SC = $\underline{O(N)}$





<u>unstable</u>