Heaps 2

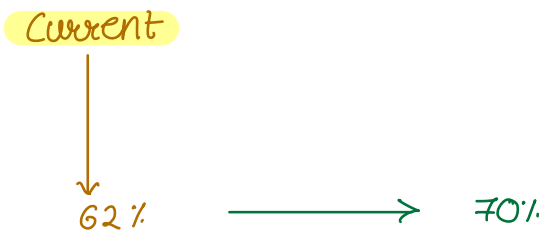
Madhan Kumar M S
Abhishek Sharma
Akansh Nirmal
amit khandelwal
Balaji S K
Bhaveshkumar
Burhan
Gagan Kumar S
Hemant Kumar
KULDEEP PATIDAR
Nikhil Nagrale
Nikhil Pandey
Purusharth A
Rajat Sharma
Rajendra
Rathna
Sanket Giri
Saurabh Ruikar
Shani Jaiswal
sharath r
Shradha Srivastava
Shreya Gupta
Sneha L
Sridhar Hissaria
Subhranil Kundu
Sumit Adwani
Suyash Gupta
Vasanth
Vetrivel H M
Vimal Kumar

Sort the array {Heap Sort}

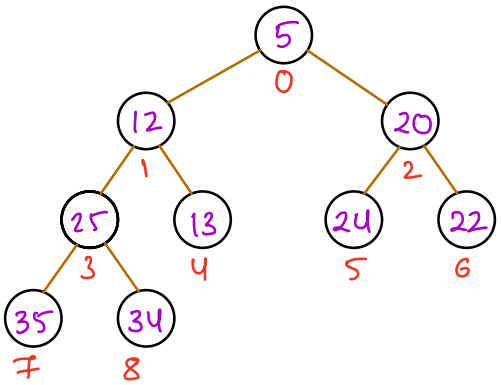
k^{th} largest no.

K places apart

Running Median *



Sort the array



Sort the array in increasing order using heap

	0	1	2	3	4	5	6	7	8
A =	5	12	20	25	13	24	22	35	34

Idea 1 ⟶ Insert all in min heap ⟶ TC: O(N log N)

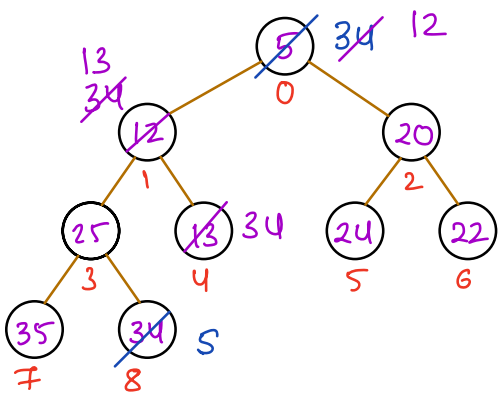
⟶ Extract min one by one ⟶ TC: O(N log N)

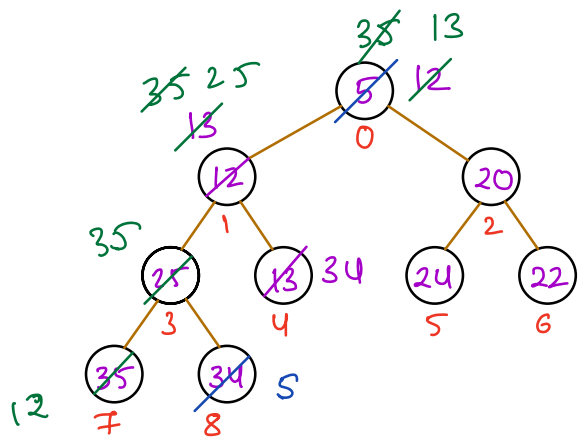
overall space ⟶ O(N)

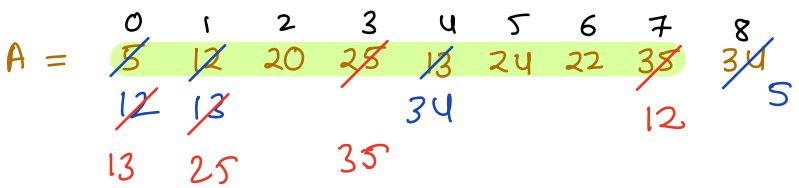
Idea 2 ⟶ Use the given array as heap.

step 1 ⟶ Convert an array into heap ⟶ O(N)

	0	1	2	3	4	5	6	7	8
A =	5	12	20	25	13	24	22	35	34







Repeat the below N times

① swap index 0 with last index
last index = last index − 1

② Heapify down from index 0

TC : O (N + N log N + N)

N : convert array to min heap

N : Reverse the array

SC : O (1)

Algo for sort	TC	SC
Merge sort	O(N log N)	O(N)
Quick sort	O(N log N) — O(N²)	O(log N) — O(N)

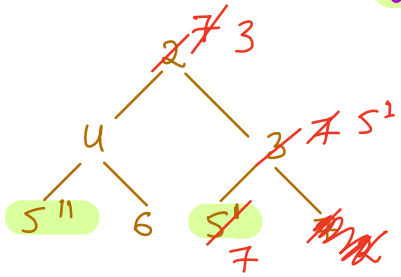
Pseudocode

```
minHeap
// build min heap from given A
j = N - 1
while j > 0
    swap (0, j)
    j--
    heapifyDown (0, minHeap, j)
// reverse the array.
```

Is heap sort in-place ? ⟶ Yes

Is heap sort stable ? ⟶ Order is not maintained.

Unstable



Given A[N], Find x^{th} largest element.

A = 8 5 1 2 4 9 7 $\quad$ K = 3

output K = 1 ⟶ 9
= 2 ⟶ 8
= 3 ⟶ 7

A = 1 2 3 4 5

K = 1 ⟶ 5
2 ⟶ 4
3 ⟶ 3
4 ⟶ 2
5 ⟶ 1

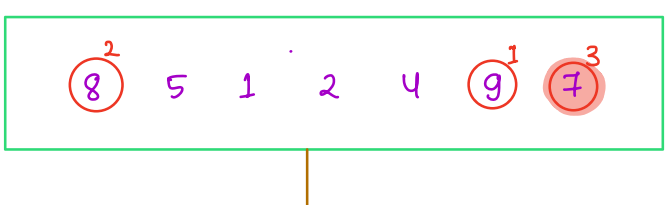
Idea 1 ⟶ Sort in ascending order, return A[N-K]
TC: O(N log N)

Idea 2 ⟶ Binary search
TC: O(N log N)

Idea 3 ⟶ Partitioning Algo {Quick sort}

Idea 4) ⟶ Heap {Max Heap}

K = 3



Ans = 7

↓

make given array into max heap ⟶ O(N)

extract max k times ⟶ K * log N

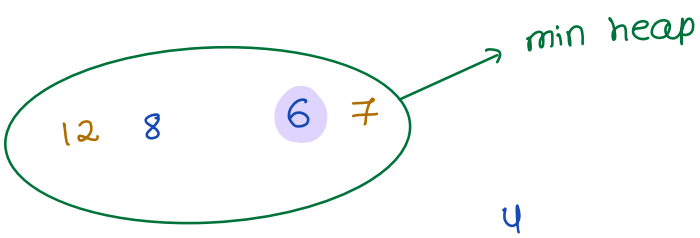
TC: O(N + KlogN)

SC: O(1)

Idea 5 > use Min heap ****

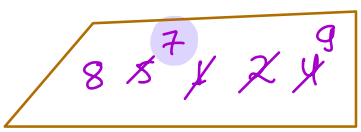
Form a team of 4 best batsmen

B_1	B_2	B_3	B_4	B_5	
12	8	4	6	7	K = 4



A = 8 5 1 2 4 9 7 k = 3

ans = 7



Pseudocode

```
minHeap //

for (val : A) {
        insert (val)
        if (size of minHeap > k) {
                extractMin (minHeap)
        }
}
print (minHeap[0])
```

TC : O (NlogK)

SC : O (K)

K^{th} Largest Element

K^{th} Largest element for all the window $\forall (0, i)$ $i >= K-1$

	0	1	2	3	4	5	6
A = {	10	18	7	5	16	19	3 }
	-1	-1	7	7	10	16	16

K=3

A = 5 4 1 6 7 K=2

4 4 5 6

22:50

Idea : Use min heap

	0	1	2	3	4	5	6
A = {	10	18	7	5	16	19	3 }
	-1	-1	7	7	10	16	16

K=3

18 19 16

TC: O(N log K)

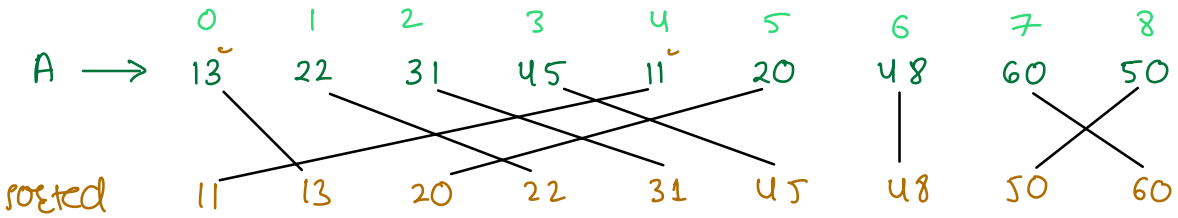
SC: O(K)

minHeap , ans

```
for val : A {
    insert in min Heap
    if (size of heap > K) extract min from heap
    if (size == K) ans.add(min from heap)
    else ans.add(-1)
}
```

Q> Given a nearly sorted array. You need to sort the array

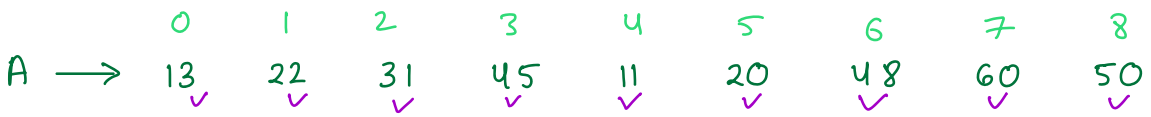
Every element is shifted away from its correct position by atmost **k steps**. k = 4



Idea 1 Sort the array TC: O(N log N) SC: O(1)

Idea 2 Min heap of size k

k = 4



11 13 20 22 31 45 48 50 60

TC : O(N log k)

SC : O(k)

Flipkart's Delivery Estimation Challenge

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 ⟶ Identify new median time for insertion

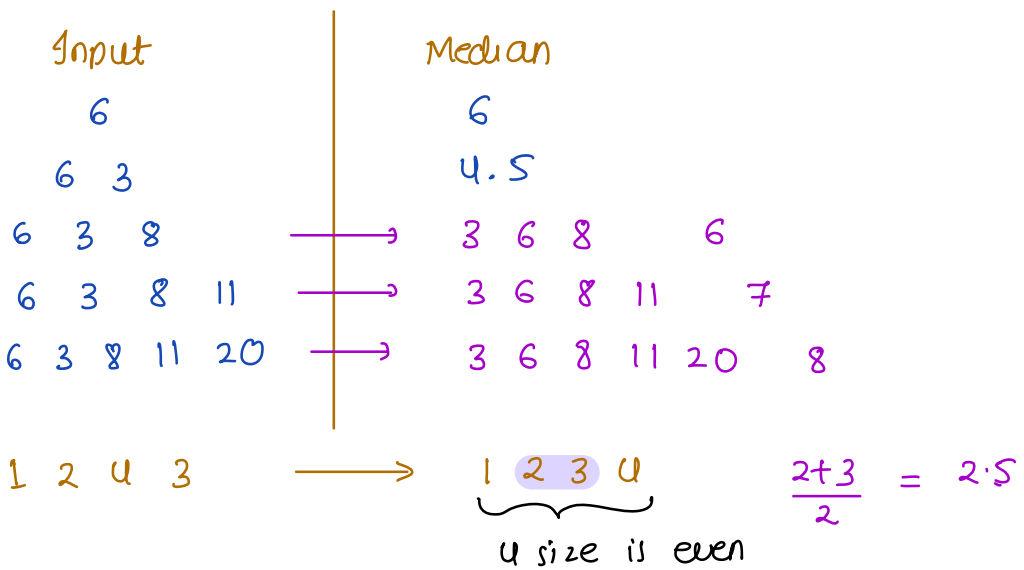
3 1 2 5 6 4

1 2 3 4 5 6

3.5

Running Median *

Given an infinite stream of integers. Find the median of current set of elements.



Idea 1 — Sort and find median

→ store the new element inside AL al.add(num)

→ sort al

→ find median

Assume N elements in al

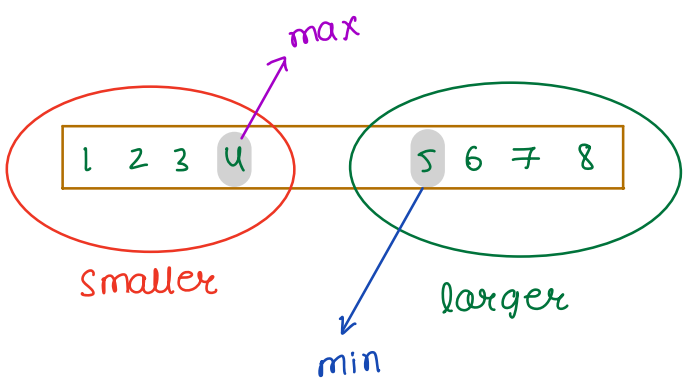
TC: per insertion = O(NlogN)

Idea 2 — Insertion sort

Use insertion sort to place newly added element at its right position

TC: per insertion = O(N)

Idea 3



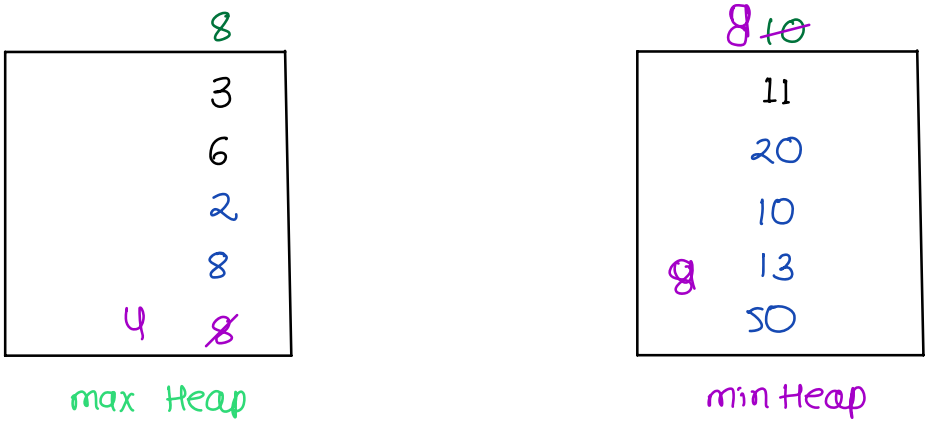
max Heap minHeap

—— Extra value goes to min Heap } Assumption

size of max heap <= size of min Heap

A = 6 3 8 11 20 2 10 8 13 50 4 .....

2 3 6 8 8 10 11 13 20 50



→ Always insert in min Heap

→ if minHeap size > max Heap size
move minimum to max Heap

→ if maxHeap size > minHeap size
move maximum to min Heap

n = size of minHeap + size of max Heap

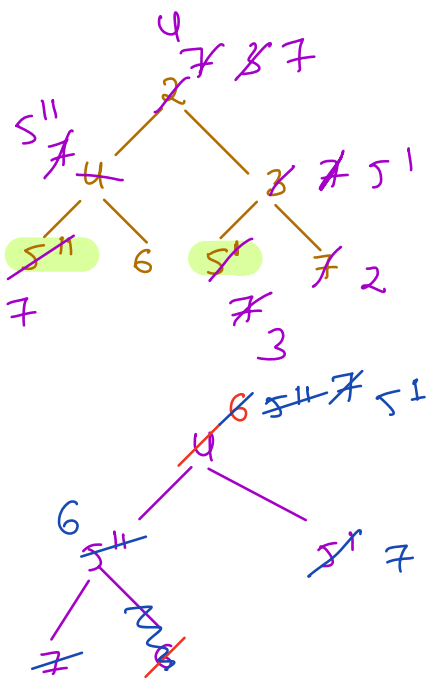
if n is odd { min in minHeap is median }

else { $\frac{min + max}{2}$ }

Assume N elements are currently present in heap.

TC: per insertion O(log N)

Doubt session



2 4 3 5'' 6 5' 7

2 3 4 5'' 5' 6 7

6 5'' 7 5' 7 6 5' 5'' 4 3 2