Q1 K-sorted array. Sort the array

each element is atmost k distance from sorted position

	0	1	2	3	4	5	6	
Eg	6	5	3	2	8	10	9	3-sorted
sorted	2	3	5	6	8	9	10	
dist	3	1	1	3	0	1	1	

Soln → Simply sort the array.

TC: N log N

Idea Do you agree that min elem has to be between idx 0 to k

yes

Because correct pos is 0 & it can be at max distance k.

Similarly 2nd min 1 — k+1

3rd min 2 — k+2

Soln Use heap

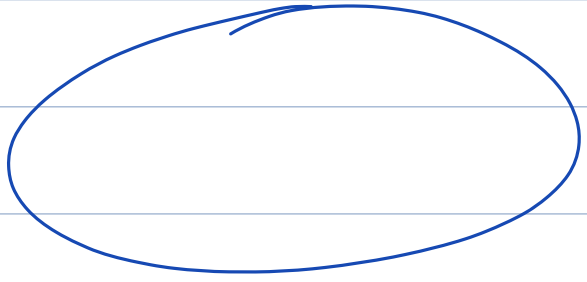
1) Build min heap 0-k

2) for (i=k+1 ; i<n ; i++) {
    extractmin()
    insert (arr[i])
}

At the end of loop, keep extracting min till empty heap.

TC: $k + (n-k)\log k$

0	1	2	3	4	5	6	
6	5	3	2	8	10	9	3-sorted
2	3	5	6	8	9	10	



02 Median of running stream of integers. Find median after every integer comes in.

Eg- 10 30 20 40 50 60 70

median ⇒ 30

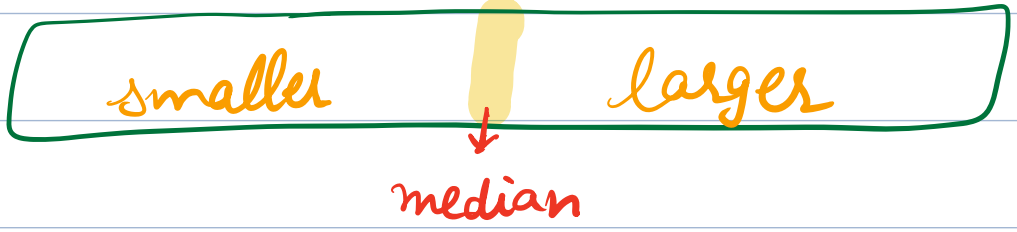
2+3 / 2 = 2.5

1 2 3 4

Median is the middle elem of the sorted version of the array

Brute force: Sort after each number comes in

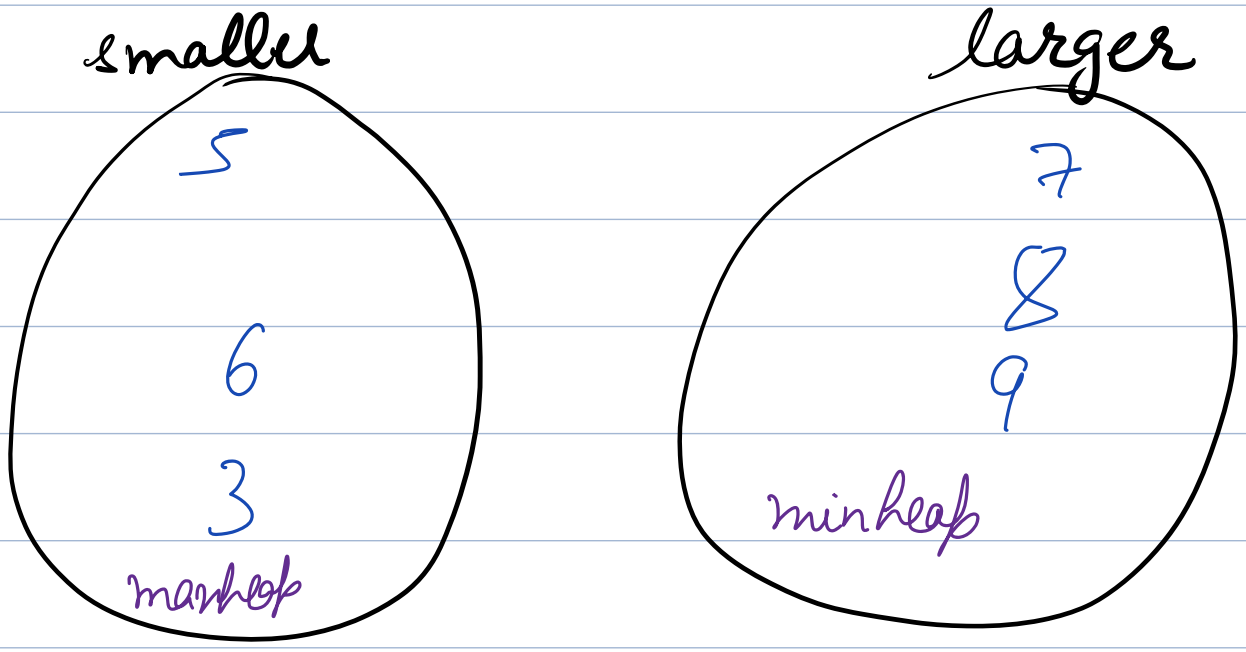
size(small) − size(large) ≤ 1



median divides in almost half.

9 8 7 3 6 5

med 9 8.5 8 7.5 7 6.5



when n = even

$$\text{median} = \frac{\max(A) + \min(B)}{2}$$

n = odd

$$\text{median} = \begin{cases} \max(A) & \text{if } A \text{ larger} \\ \min(B) & \text{if } B \text{ larger} \end{cases}$$

Code

```
for(i=0; i<n; i++) {
  // elem = x comes
  if( x < max of A) {
    insert x in A
    if( size(A) - size(B) > 1) {
        extractMax from A
        insert in B
    }
  }

  else {
    insert x in B
    if( size(B) - size(A) > 1) {
      extract min from B
      insert in A
    }
  }
}
```

{done}

TC : O(nlogn)

SC : O(N)

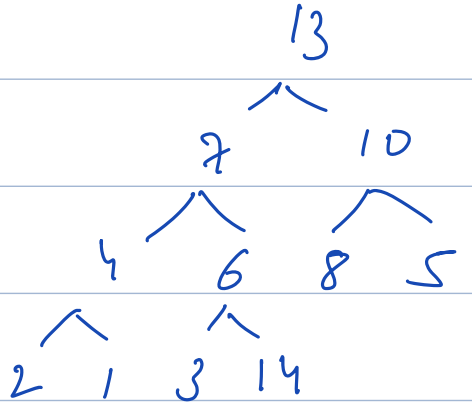
Q1 Sort the given array using heaps

Heap sort

array → (TC: O(N)) → heap → (TC: O(n log n)) → extract min n times & put in new array

TC: O(n log n) SC: O(N)

0	1	2	3	4	5	6	7	8	9	10
~~14~~	13	10	7	6	8	5	2	1	3	~~4~~
4										14



1) Build max heap in array

```
i = n-1
while (i > 0) {
    swap(a[0], a[i])
    i--
    heapify(0, arr, i)
}
```

SC: O(1)

Q2 Find K^{th} largest element in array.

Eg - 8 5 1 2 3 6 9 7

k=3 ans = 7

Solutions - 1) sort & return arr[n-k]

2) sort & Binary search

3) Max heap & extract-max k times

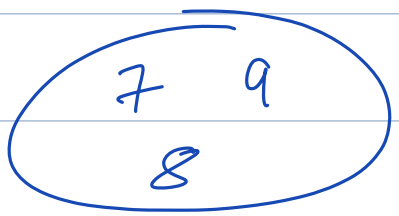
How to do with heaps

1) Build max heap.
2) Extract max k times

TC: O(N) + k log N SC: O(1)

Minheap => 8 5 1 2 3 6 9 7

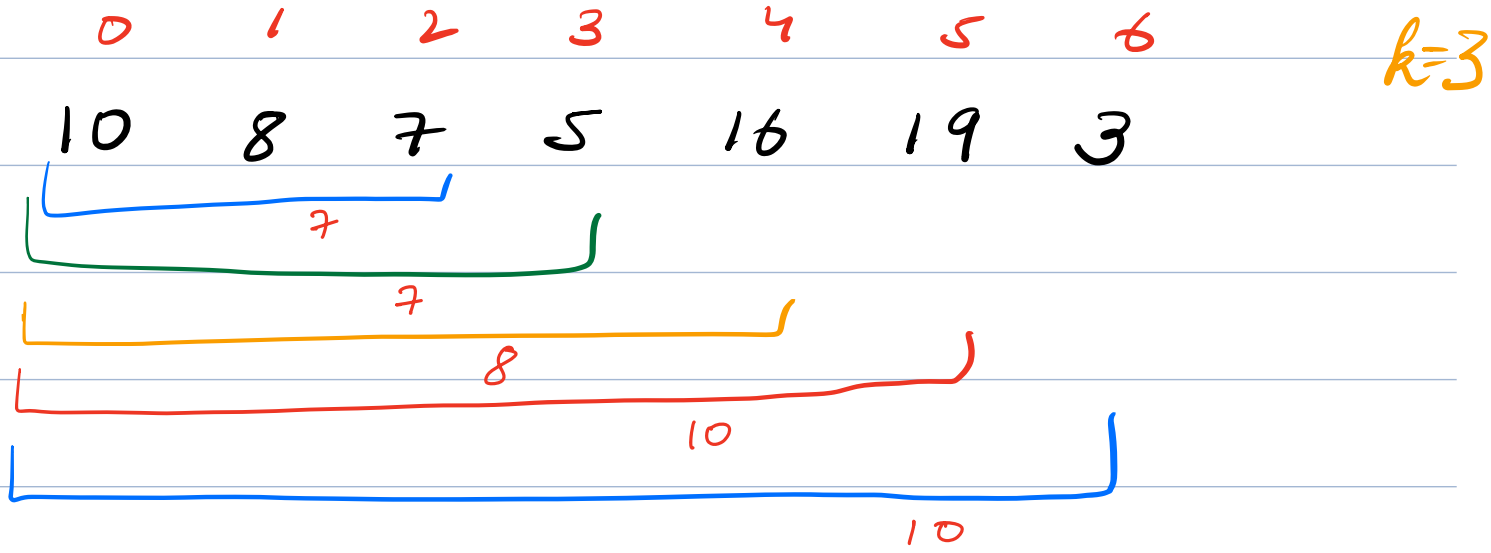
k=3



ans = mh.min()

SC: O(k)

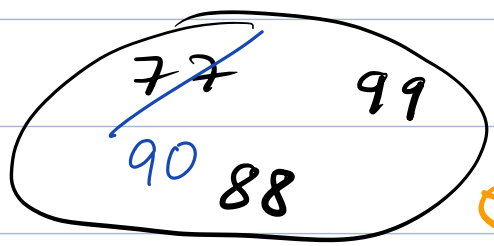
Q3 Find K^{th} largest for every prefix [0-i]



Brute force: For each window, iterate on the whole window. $O(n^2)$

Idea Maintain the k largest no.s of the present window

Eg- k=3



[0: 12]

a[13]: 90

Will you put 90 in the set

Need to compare with min of the set

⇒ minheap

remove min (77)
insert 90.

What is the k^{th} largest ?

min of heap

Dry run

0	1	2	3	4	5	6
10	8	7	5	16	19	3

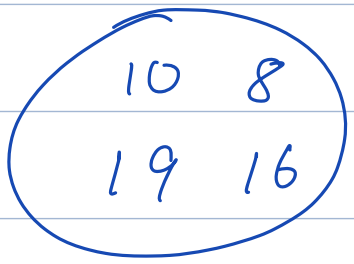
k=4

⇒ 5

⇒ 7

⇒ 8

⇒ 8



Code

```
list<int> ans
min heap mh
for(i=0;i<k;i++)
    mh.insert(a[i])
ans.insert(mh.getmin())

for(i=k;i<n;i++){
    elem = a[i]
    if(elem > mh.getmin()){
        mh.deletemin()
        mh.insert(elem)
        ans.add(mh.getmin()
    }
}
```

TC : $k \log k + (n-k) \log k$

$\Rightarrow n \log k$

SC : $O(k) + n-k+1$

$\Rightarrow O(N)$