Today's agenda

→ PriorityQueue

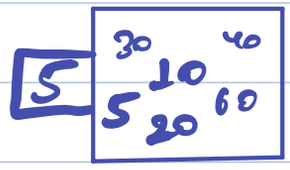
- Introduction to Heap / PQ.
- K smallest element. +1
- Median of an array. → Leetcode Hard

// Introduction

	insert (n)	getmin()	deletemin()
LinkedList	O(1) | O(N)	O(N)	O(N)
Queue	O(1)	O(N)	O(N)
Hashmap	O(1)	O(N)	O(N)
PQ	O(logN) (size of PQ)	O(1)	O(logN)

Min PQ

```
PriorityQueue<Integer> PQ = new PriorityQueue<>();
```
(PQ: name)




```
PQ.peek();   // get the min element
```
2

```
PQ.remove();   // remove and return the min ele
```
2

```
PQ.add(5);
```

Max PQ

```
PriorityQueue<Integer> PQ = new PriorityQueue<>(Collections.reverseOrder());
```
(PQ: name)

```
PQ.add(5);   // add in PriorityQueue.
PQ.add(10);
PQ.add(20);
PQ.add(2);
```

```
PQ.size();
```
↳ no. of elements in PQ.

Q) Kth Smallest Element

↳ Given N distinct elements, Print K Smallest elements.

Ex: arr[10] = { 8 3 10 4 11 2 7 6 14 1 }
(indices: 0 1 2 3 4 5 6 7 8 9)

K=4: 1 2 3 4

arr[8] = { -3 6 2 0 8 -3 10 4 }
(indices: 0 1 2 3 4 5 6 7)

K=3: -3 -3 0

//idea 1

↳ Sort the array and return the first K elements.

T.C: O(NlogN + K) ≈ O(NlogN)

//idea 2

↳ Add all the elements to min PQ and get the first elements.

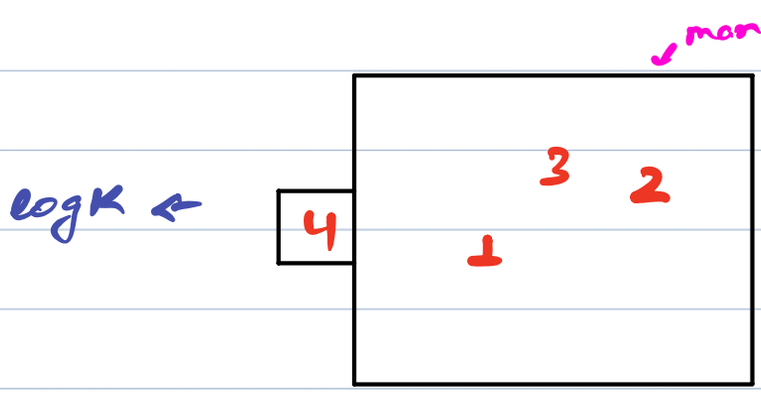
T.C: O (NlogN + KlogN) = O(NlogN)

S.C: O(N)

//idea 3

arr[10] = { 8 3 10 4 11 2 7 6 14 1 }
(0 1 2 3 4 5 6 7 8 9)

k=4; 4 3 2 1



//Pseudo code

```
void KthSmallest ( int arr[N], int k){
    maxHeap<Integer> mh;

    for(int i=0; i<k; i++){
        mh.add(arr[i]);
    }

    for (int i=k; i<N; i++){
        if (arr[i] < mh.Peek()){
            mh.remove();
            mh.add(arr[i]);
        }
    }

    while (mh.size()>0){
        Print (mh.remove());
    }
}
```

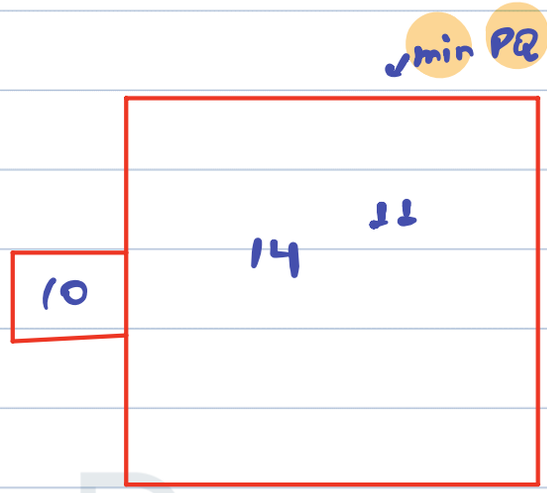
T.C: O(NlogK)
S.C: O(K)

Q) K largest elements

	0	1	2	3	4	5	6	7	8	9
arr[10]: {	8	3	10	4	11	2	7	6	14	1 }

K:3

↳ 10 11 14



//median

↳ middle element of sorted number.

arr[3] : { 2 5 3 }

↳ { 2 3 5 } → 3

arr[5] : { 4 3 6 8 5 }

↳ { 3 4 5 6 8 } → 5

arr[6] : { 4 3 9 5 12 2 }

↳ { 2 3 4 5 9 12 } → $\frac{4+5}{2}$: 4.5

arr[4] : { 4 6 10 14 }

↳ $\frac{6+10}{2} = 8$

Break till 10:17 PM

Q) Print median after each insertion.

arr[5]: 9 6 3 10 4

9 → 9 | 6 → 7.5 | 3 → 6 | 10 → 7.5 | 4 → 6

Idea 1

→ After every insertion, sort the array and return the middle one.

T.C: $O(N \log N * N) \simeq O(N^2 \log N)$

Idea 2

3 5 11 23 20 25 17 7

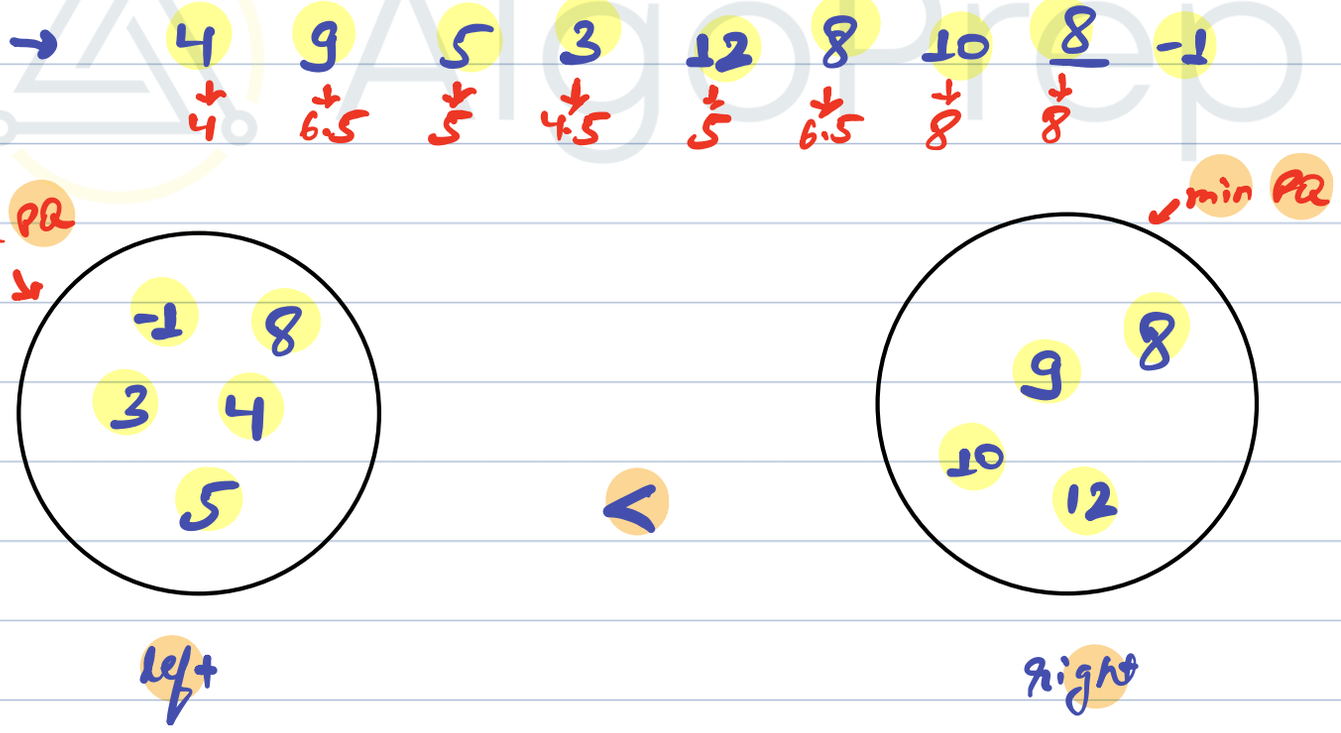
even element

median: $\frac{\text{max of left bucket} + \text{min of right bucket}}{2}$

odd element

3 5 11 23 20 25 17

median : max of left



if (left.size() == right.size()) { new element is (odd)th element }

IIPSuedo Code

```
Class MedianFinder {
    maxHeap<Integer> left;
    minHeap<Integer> right;
    Public MedianFinder() {
    }
    Public void addnum(int num){
        if (left.size() == right.size()){
            right.add(num);
            left.add(right.remove());
        }
        else {
            left.add(num);
            right.add(left.remove());
        }
    }
    Public double findmedian(){
        if (left.size() == right.size()){
            return (left.peek() + right.peek()) / 2.0;
        }
        else{
            return (left.peek() * 1.0);
        }
    }
}
```

T.C: O(N*logN)

S.C: O(N)

3*logN

O(1)