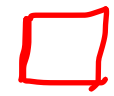
Priority Queue.

(Heap)

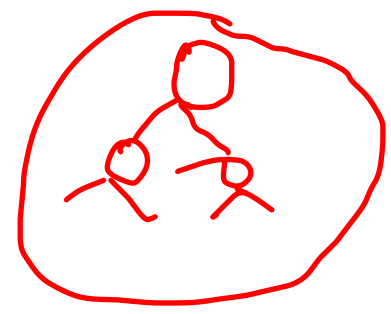


① → he

② → VIP

$\frac{1}{cx}$ $\frac{2}{cv}$

→ any element placed in PQ will always be in sorted order.



PQ.

-2
2
5
7
10
.

Java → By default PQ is ascending order

C++ → decending order.

only have access to the top element.

PQ.

-2
0
3
7
10

arr → [-2, 3, 0, 10, 7]

→ duplicates are allowed.

Syntax

```
PriorityQueue<Integer> pq = new PriorityQueue<>();
```

In-built

→ pq.add(val); // add element in PQ

→ pq.remove(); } remove & return

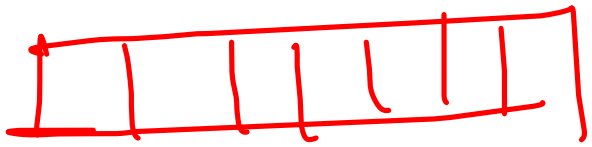
→ pq.poll();

→ pq.peek(); // top element

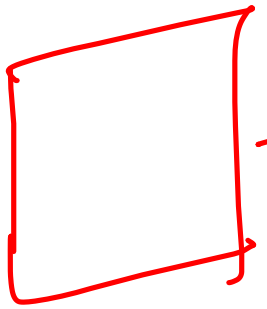
→ pq.size();

→ pq.isEmpty();

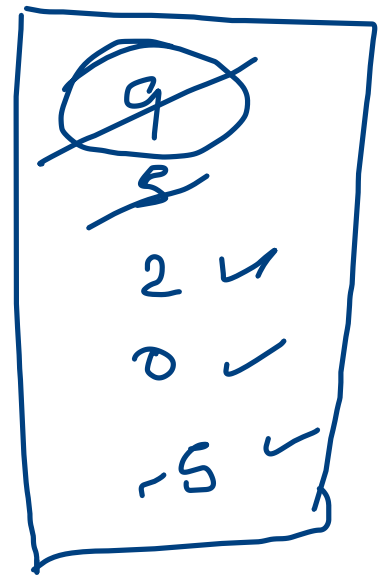
TC → O(log(n))



→ Array.sort() // n log n



→ n×log n



2 types of PQ.

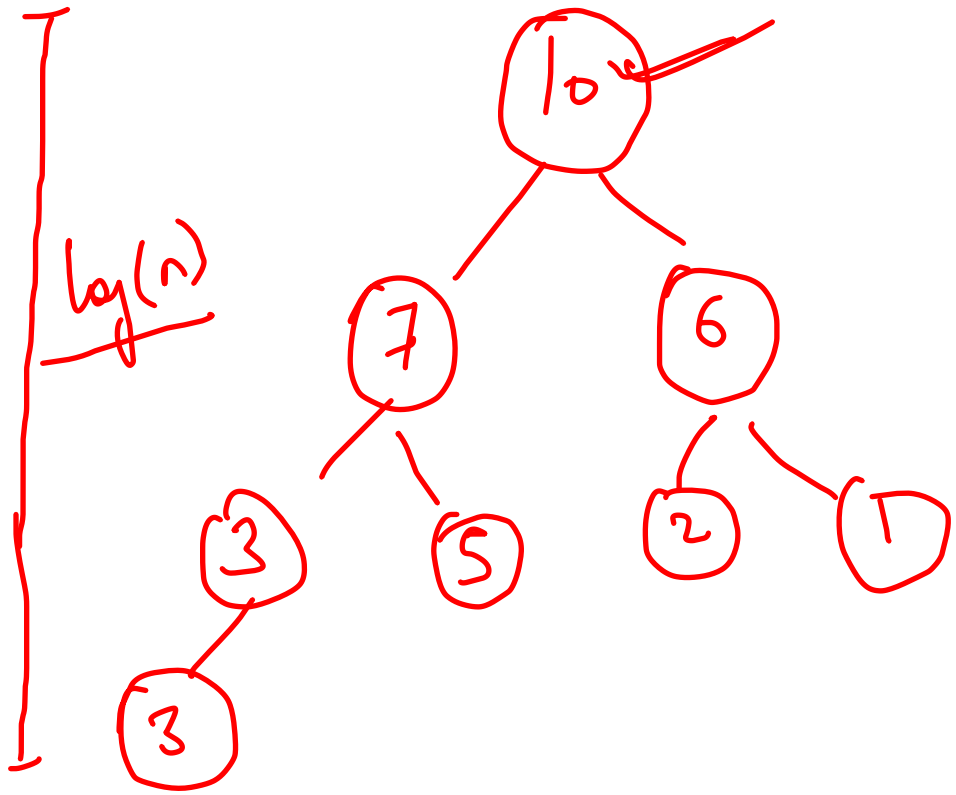
→ minHeap : ascending order sort

→ maxHeap : decending order sort

Lambda function in PQ.

```
Arrays.sort(arr, (a,b) -> {
    return b-a;
});
```

```
PriorityQueue<Integer> pq = new PriorityQueue<>((a,b) -> {
    return b-a;
});
```



n elements in tree

height : log(n)

HW_Take Gifts From the Richest Pile 2

25	64	9	4	~~100~~	
25	~~64~~	9	4	10	— 1
~~25~~	8	9	4	10	— 2
5	8	9	4	~~10~~	— 3
5	8	9	4	3	— 4 → 29

```java
Scanner s = new Scanner(System.in);
int n = s.nextInt();
int gifts[] = new int[n];
for(int i = 0; i < n; i++){
    gifts[i] = s.nextInt();
}
int k = s.nextInt();

// max heap
PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
for(int gift : gifts){
    pq.add(gift);
}

for(int i = 0; i < k; i++){
    int maxGift = pq.poll(); // 100
    int remaining = (int) Math.floor(Math.sqrt(maxGift)); // 10
    pq.add(remaining);
}

int total = 0;
while(!pq.isEmpty()){
    total += pq.poll();
}
System.out.println(total);
}
```

$O(n \log n)$

```java
public static int smallest(int arr[], int k){

    PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
    for(int i = 0; i < k; i++){
        pq.add(arr[i]);
    }

    for(int i = k; i < arr.length; i++){
        if(arr[i] < pq.peek()){
            pq.poll();
            pq.add(arr[i]);
        }
    }
    return pq.peek();
}
```

```
6
7 10 4 3 20 15
3
```

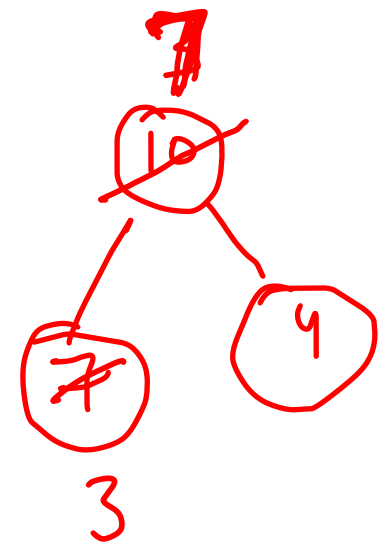
1 2 3 4 5

i=0, 7

i=1, 10

i=2, 4

i=3<3 F



1 3 7

k=3

arr[3] = 3<10 T

arr[4] = 20<7 F

arr[5] = 15<7 F

6 | P → 7

min

[3, 4, 7, 10, 15, 20]

max

[20, 15, 10, 7, 4, 3]