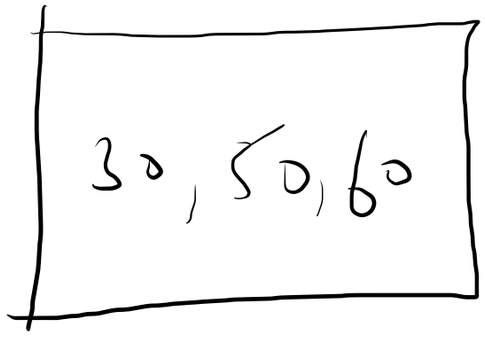
28 Jan 2022 — Priority Queue Introduction

add (50)
add (60)
add (30)



peek() → O(1)

add() → O(log n)
remove()

remove() → remove most priority element i.e. 30 here as min element gets most priority

peek() → returns most priority element

```java
import java.util.*;
import java.io.*;

public class Main {


    public static void main(String[] args) throws Exception {
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        pa.add(20);
        pa.add(40);
        pq.add(33);
        pq.add(57);
        pq.add(10);
        pq.add(67);

        System.out.println(pq.peek()); //
    }
}
```

→ 10 is o/p

after this if we do pq.remove(), 10 is removed & then if we do pq.remove() 20 is removed as new element is taken as next highest Element

Sort an array

```java
int[] arr = {10, 57, 33, 44, 56, 11, 98};
for(int val: arr){
    pq.add(val);
}

while(pq.size()> 0){
    System.out.println(pq.peek());
    pq.remove();
}
```

→ n log n

→ n log n

→ log n

$\Rightarrow 2 n \log n \Rightarrow O(n \log n)$

so it sort in $O(n \log n)$

Priority or rank basis [choose to most priority]

```
public static void main(String[] args) throws Exception {
    PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
    pq.add(20);
    pq.add(40);
```

To get priority of max element as 1st

Q K-largest element

[10, 19, 3, 74, 86, 57, 24, 5, 11] K=3

O/P → 86, 74, 57 [Top 3 largest Element]

k <= N (N is arr.length)

Sol → Make Priority Queue with Max element as Most priority ones & then do k-removal

TC → $O(n \log n) + O(k \log n)$

↓ → n elements Pqueue mei bharne ke lie

↳ → k removal

SC → $O(n)$

We want TC → $O(n \log k)$ & SC → $O(k)$

P_Queue mei k se jyada element nhi have chahie

[10, 19, 3, 74, 86, 57, 24, 5, 11]

10, 19, 3

Initially

Now at 74
Min value se check hua & compare both
so 74 > 3 so remove 3 & put 74

at 86, 86 > 10 (min se nikala tha)
& put 86 there

10, 19, 74

86, 19, 74

57 > 19 so put 57 there instead of 19

85,74,57

at 24 $24 < 57$ so i++

$5 < 57$ so i++

$11 < 57$ so i++

for K-Map Minimum nikalte jao & Maximum

karte jao

Its like small nikalte jao & boye karte jao

$O(n \log k)$ TC ($\log k$) as k elements in p-queue

& done n times insertion SC → $O(k)$

```java
import java.io.*;
import java.util.*;

public class Main {

   public static void main(String[] args) throws Exception {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int n = Integer.parseInt(br.readLine());
      int[] arr = new int[n];

      for (int i = 0; i < n; i++) {
         arr[i] = Integer.parseInt(br.readLine());
      }

      int k = Integer.parseInt(br.readLine());

      PriorityQueue<Integer>pq=new PriorityQueue<>();
      for(int i=0;i<k;i++){
          pq.add(arr[i]);
      }
      for(int i=k;i<arr.length;i++){
          int val=pq.peek();
          if(val<arr[i])
          {
           pq.remove();
          pq.add(arr[i]);
          }
      }
      while(pq.size()>0){
          System.out.println(pq.peek());
         pq.remove();
       }

}
}
```

k log k

(n-k) log k

k log k

$\Rightarrow (n+k)\log k$

$k << n$ so

$n \log k$

```java
    // This function takes as input an array,n length of array
    // It should print required output.
     public static void solve(int n,int[] arr,int k){
     PriorityQueue<Integer>pq=new PriorityQueue<>();
      for(int i=0;i<k;i++){
           pq.add(arr[i]);
      }
      for(int i=k;i<arr.length;i++){
          int val=pq.peek();
          if(val<arr[i])
          {
           pq.remove();
          pq.add(arr[i]);
          }
      }
      ArrayList <Integer>al=new ArrayList<>();
      while(pq.size()>0){
          al.add(pq.peek());
          pq.remove();
      }
      for(int i=al.size()-1;i>=0;i--){
          System.out.print(al.get(i)+" ");
      }

    }
```

decreasing order bde se chote o/p mang rhe the isliye Array list mei likh ke usko ulta display kar do.

Example 2:

```
Input: nums = [3,2,3,1,2,4,5,5,6], k = 4
Output: 4
```

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook | 125 Amazon | 17 LinkedIn | 14 Microsoft | 11 Google | 4
ServiceNow | 2 Goldman Sachs | 2 Uber | 2 Adobe | 2 Oracle | 2 Cisco | 2
Walmart Labs | 2 Shopee | 2

215. Kth Largest Element in an Array

Medium 8069 446 Add to List Share

Given an integer array `nums` and an integer `k`, return *the* k^{th} *largest element in the array*.

Note that it is the k^{th} largest element in the sorted order, not the k^{th} distinct element.

Example 1:

```
Input: nums = [3,2,1,5,6,4], k = 2
Output: 5
```

```java
class Solution {
    public int findKthLargest(int[] arr, int k) {
        PriorityQueue<Integer>pq=new PriorityQueue<>();
        for(int i=0;i<k;i++){
            pq.add(arr[i]);
        }
        for(int i=k;i<arr.length;i++){
            int val=pq.peek();
            if(val<arr[i])
            {
             pq.remove();
            pq.add(arr[i]);
            }
        }
       return pq.peek();
    }
}
```

Success Details ›

Runtime: 10 ms, faster than 33.16% of Java online submissions for Kth Largest Element in an Array.

Memory Usage: 45 MB, less than 7.29% of Java online submissions for Kth Largest Element in an Array.

Leet code soln

① Quick Select Algo (Kth largest) → O(N) on Avg But O(N²) in WC

② Other way → O(Nlgk) with O(k) Extra space

There is another method called as poll() in priority queue pq.poll(); Returns the next priority element & then deletes it

Rank list is main use case of PriorityQueue

Q Merge k-sorted list

We need to learn comparable to do it