DAY 124 DSA

Intro-duction to Priority Queue.

Date → 28 Jan, 22.
Day → Friday

Date / /
Page No.

Module → HashMap and Heap

Agenda → 1) Introduction to Priority Queues (Heap)
2) K Largest Elements

PRIORITY QUEUE

PriorityQueue के predefinid functions मिलते हैं (similar to Queue), इन्हें का func का working नहीं जानना है।

It is based on abstraction.

1. → Priority Queue is an abstract data type
2. Priority Queue is very similar to Queue,
3. however, In Priority Queue, every element has some priority.

हर element की कुछ Priority होती है।

4. The priority of the elements in a priority queue determines the order in which elements are removed from the Priority Queue
5. Therefore, all the elements are either arranged in ascending or descending order.

So, a Priority Queue is an extension of the Queue with the following properties:-

1) → Every queue has a priority associated with it
2) → An element with higher priority is dequeued before an element with lower priority

मतलब जिस element की ज्यादा priority होगी, तो Priority Queue में से पहले dequeue (remove) होगा

3) If two elements have the same priority, they are served according to their order in the queue.

अगर 2 elements की priority same है, तो जो element Queue में पहले आता है occurance में, तो पहले dequeue होगा

4).* The elements from Priority Queue are retrieved in sorted order.

Priority Queue में से सभी elements को जब भी हम access करेंगे (elements को जब PQ से बाहर निकालेंगे) तो हमें सभी elements एक sorted order में मिलेंगे

sorted

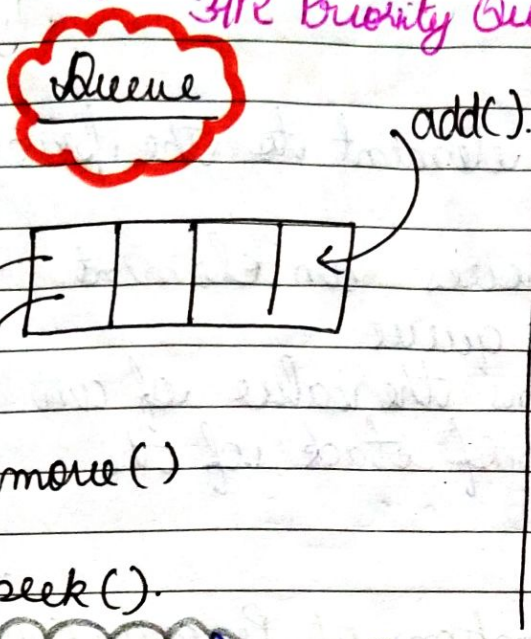
5). It is important to note that the elements in a priority queue may not be sorted. However elements are always retrieved in an sorted order.

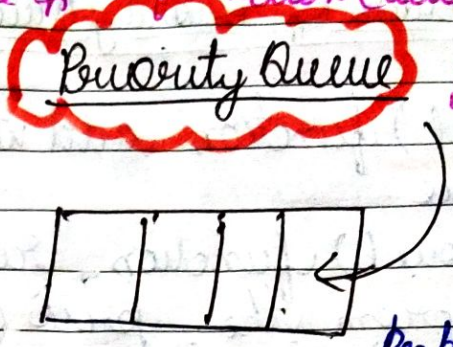
Priority Queue में से जब सभी element remove होकर बाहर निकलेंगे तो वो sorted order में ही होते हैं।

The Priority Queue class has the same structure and function as that of Queue.

Priority Queue का structure बिल्कुल Queue जैसा होता है और Priority Queue के function (add, remove, peek) भी Queue जैसे ही है।

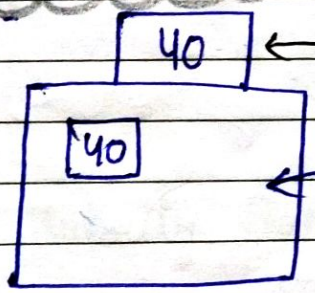
6.



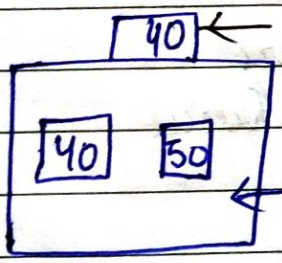


Peek() function and Remove() function works according to the priority

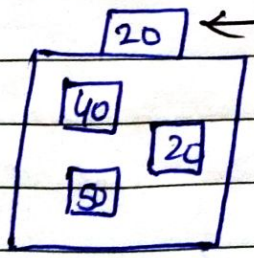
Example 1) The Priority will be given to smaller elements



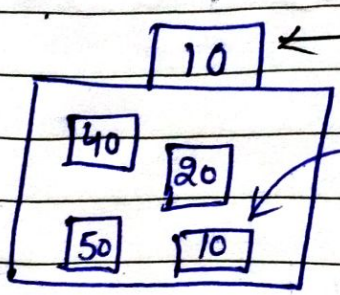
40 को Priority Queue में डाला तो Peek होगी 40.



अब 50 को Priority Queue में डाला तो अभी भी Peek पर होगा 40



अब 20 को Priority Queue में डाला तो अब peek पर होगा 20.



अब 10 को priority queue में डाला तो अब 10 peek पर होगा.

(7) Functions of Priority Queue

→ add() function add an element to the priority queue.
→ remove() function removes an element from the priority queue.
→ peek() function returns the value of an element at the top of stack of the priority queue.

(8) Time Complexity of functions of Priority Queue

add() → O(log n)
remove() → O(log n)
peek() → O(1)

(9) Heap

↳ is an advance Data Structure.
Heap is also known as Priority Queue.

SIMILARITY IN QUEUE AND PRIORITY QUEUE (PQ).

Date / /
Page No.

① Queue और Priority Queue के functions के नाम भी same हैं और काम भी same हैं।

add() → element को add करता है Queue/PQ में।
remove() → Element को remove करता है Queue/PQ से।
peek() → peek() function element return करता है।

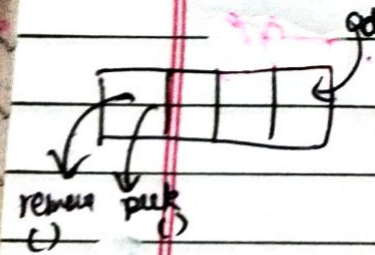
② Queue और Priority Queue का structure same है।

* Priority Queue is just an extention of queue.

DIFFERENCES BETWEEN QUEUE AND PRIORITY QUEUE

① When an element is removed from the priority queue, a certain priority is followed. We can set this priority according to our need.

जब हम queue से कोई element remove करते हैं तो head of queue element remove हो जाता है और peek() function भी head of the queue ही return करता है।



लेकिन जब हम priority queue में से कोई element remove करते हैं तो highest priority वाला element remove होता है। और peek() function भी highest priority वाला element ही remove करता है।

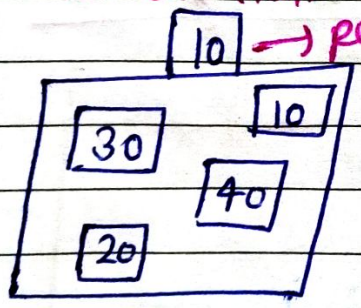
② हम Priority अपनी need की according set कर सकते हैं।

↳ हम 2 types of Priority के रख सकते हैं Priority Queue में

a) ⟶ Highest Rank Order (Smaller number को ज्यादा Priority की जाती है)

↳ छोटी Rank (Rank 1/2/3 --- की ज्यादा Priority होती है as compared to Rank 150/160/ etc.)

अगर Priority Queue की priority Highest Rank Order है, मतलब अगर smaller elements की priority ज्यादा है तो जब हम priority queue से elements retrieve करेंगे तो हमें सभी elements ascending order में मिलेंगे।



Output after we retrieve all elements

→ 10, 20, 30, 40

↳ pq.remove();
10

→ pq.remove();
10 20

→ pq.remove();
10 20 30

→ pq.remove();
10 20 30 40.

b) Highest Score Order (greater number को smaller number से ज्यादा priority मिलेगी)

↳ अगर किसी के score ज्यादा है (400/500) तो उसको ज्यादा priority मिलेगी when compared to someone जिसके score कम है (40-50).

अगर priority queue की priority set है Highest Score Order मतलब अगर elements को ज्यादा priority दी जाएगी तो जब हम उस priority queue से सभी elements को retrieve करेंगे तो हमें सभी elements descending order में मिलेंगे।

a) PRIORITY SET TO SMALLER VALUES

Date / /
Page No.

Creating a Priority Queue

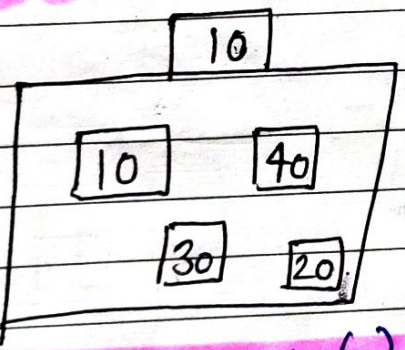
name of priorityQueue

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

datatype of elements

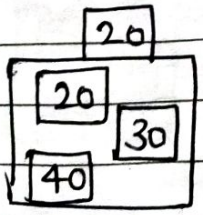
By default, the priority will be set of smaller values (Higher Rank Order)

```
pq.add(40);
pq.add(20);
pq.add(30);
pq.add(10);
```



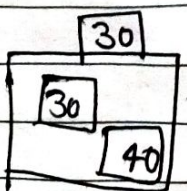
`pq.peek();` → 10.

`pq.remove();` → 10 removed from pq.



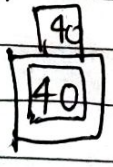
`pq.peek()` → 20

`pq.remove();` → 20 removed from pq.



`pq.peek();` → 30.

`pq.remove();` → 30 removed from pq



`pq.peek();` → 40

`pq.remove();` → 40 removed from pq.

So, we retrieved all the elements from pq.

The Output → 10 20 30 40

We got the output in Sorted Order (ASCENDING)

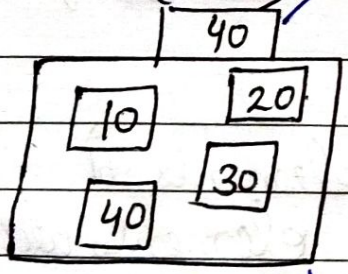
b) PRIORITY SET TO GREATER VALUES

```
PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
```

↳ reverse order में elements remove होंगे।

greater elements will be given more priority
(बड़े digit को ज्यादा priority मिलेगी)
तो वो पहले queue में से remove होंगे।

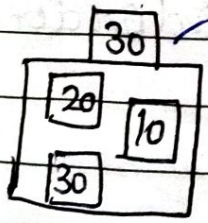
```
pq.add(20);
pq.add(40);
pq.add(30);
pq.add(10);
```



→ peek

pq.peek(); → 40

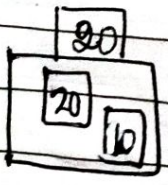
pq.remove(); 40 will be removed



→ peek

pq.peek(); → 30

pq.remove(); → 30 will be removed



pq.peek(); → 20

pq.remove(); → 20 will be removed

pq.peek(); → 10

pq.remove(); → 10 will be removed.

HEAP SORT

Date
Page No.

Ques- 1

हमें एक array दिया होगा. This array will be unsorted. How can we sort the array using priority queue in ascending and descending order.

a) Approach for getting all elements in Ascending Order

10	57	33	44	56	11	98

Make a priority queue, priority will be set as default

पहले loop लगाके इस array के सभी elements को priority queue में add करो

और priority को default ही रहने दो तो smaller elements will be given more priority

priority queue से अब जब हम elements को remove करेंगे, तो now the elements will be retrieved (received) in a sorted order

10	11	33	44	56	57	98

Smaller element is given more priority

b) Approach for getting all elements in Descending Order.

Date / /
Page No.

10	57	33	44	56	11	98

Make a priority queue, priority will be given more to greater elements

PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
↳ priority is set in reverseOrder.

अब array पर loop लगाके सब elements को pq में add करो

→ Now, elements will be retrieved in reverse Order.

98	57	56	44	33	11	10

SORT THE ARRAY IN ASCENDING ORDER

```
import java.util.*;
public class FirstPriorityQueue
{
  public static void main(String args)
  {
    int[] arr = {40,90,20,14,33};
    PriorityQueue<Integer> pq = new
                PriorityQueue<>();

    for(int val : arr)
    {
      pq.add(val);
    }

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

O(nlogn)

→ One for loop has O(n) time complexity

→ add() function of pq is having O(logn) time complexity

O(nlogn)

one while loop take O(n) time complexity & each element of loop use remove() function whose time complexity is O(logn)

So, we get a sorted array using this way, as we can do sorting in this way, it is also called Heap sort and the time complexity of this sorting is O(nlogn). as the time complexity of each loop is O(nlogn) which sum up to 2O(nlogn) which is O(nlogn) only.

14	20	33	40	90

Heap Sort is of 2 types → the one with space complexity O(n)
→ Another one where constant space k is used.

SORT THE ARRAY IN DESCENDING ORDER

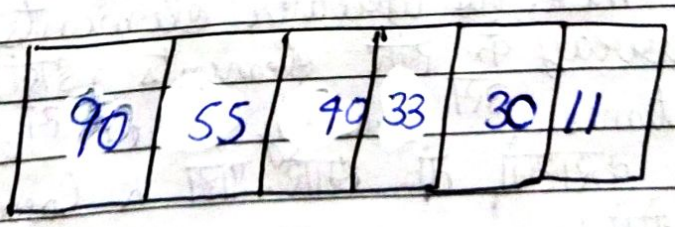
```
import java.util.*;
public class FirstPriorityQueue
{
  public static void main(String args)
  {
    int [] arr = {40,30,33,90,55,11};
    PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());

    for(int val : arr)
    {
      pq.add(val);
    }

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

O(nlogn) — highest priority वाले number को दी जाएगी

O(n)



Ques 2 K-Largest Elements

हमे एक array given होगा और एक value k given होगी। हमे array मे से k no. of largest elements print करने होंगे।

10	19	3	74	86	57	24	5	11

K = 3

74, 86, 57

These are the 3 largest elements in the given array.

Approach 1

So, we have 1 Approach where the
Time Complexity → O(n log n)
Space Complexity → O(n).

→ Here Time Complexity → O(n log n) Space Complexity → O(n)
→ हम पहले एक priority queue बनाएंगे where the priority will be given more to greater elements. उस priority queue मे array के सारे elements डाल देंगे
→ फिर k times उस Priority Queue मे से elements को remove कराएंगे तो हमारे पास k largest element retrieve होंगे।

→ Here Time Complexity → O(K(log n)) (no. of element removed → remove() function)
→ O(log n).

* So overall Time Complexity → O(n log n) + O(k log n)
→ O((n+k) log n))
→ O(n log n)
* overall Space Complexity → O(n) {n elements priority queue में store}

```
import java.util.*;
public class Main {

public static void main (String args[])
  {
   Scanner scan = new Scanner (System.in);
   int n= scan.nextInt();

   int arr[] = new int [n];
   for(int i=0;i<n;i++)
     {
      arr[i] =scan.nextInt();
     }

   int K= scan.nextInt();
PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
   for(int val:arr)
     {
      pq.add(val);
     }
   for(int i=0; i<K;i++)
     {
      int val= pq.peek();
      pq.remove();
      System.out.println(val);
     }
  }
}
```

new array बनाती n size की

Array में elements को input करती

K की value input की

pq की PriorityQueue, Priority queue to greater elements

Array के सारे elements को priority queue में डालती

O(n log n) → Time Complexity
O(n) → Space Complexity

K times pq से element remove करती और उसे peek value print करती

O(K log n) → Time Complexity

So, we will get our K largest elements.

Approach 2

Date / Page No.

↳ Here the Time Complexity allowed is → O(n log n)
↳ Space complexity allowed is → O(k).

Here k is the number of elements to be removed.

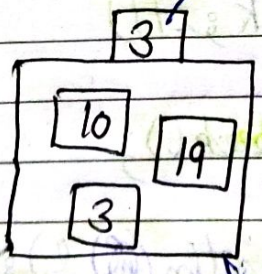
Step 1) हम priorityqueue बनाएँगे where the priority is set as default (छोटे elements को ज्यादा priority दी जाएगी)

Step 2). pq (priority queue) में array के k elements ही add किए जाएँगे

example →

10	19	3	74	86	54	24	5	11
0	1	2	3	4	5	6	7	8

Let's say k = 3



peek पर है 3

priority is given to smaller elements

Step 3). अब array पर एक loop चलाते हैं
Kth index वाले element से array के last element तक

10	19	3	74	86	54	29	5	11

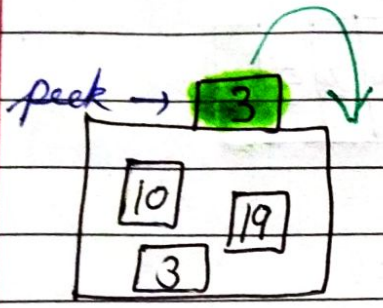
we will compare all these elements with peek

a) → If the element in the array is less than peek, then do nothing

b) But if the element in the array is greater than peek of pq,

↳ Then, we will remove that element from pq

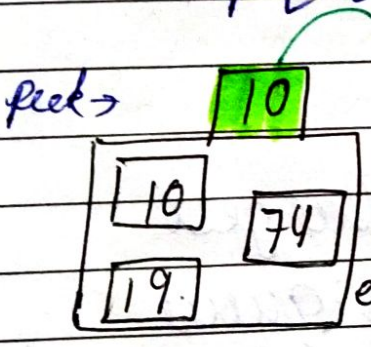
↳ Add the element in the array in the pq

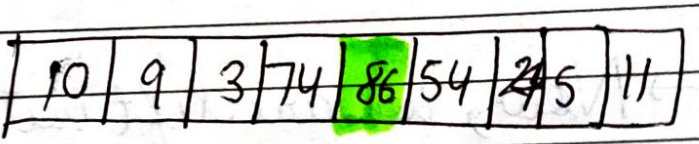


These elements will be compared with peek

element > peek

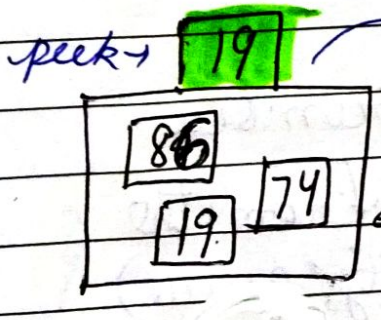
So, 74 > 3, ∴ we will remove the element from pq & add element in the array in pq

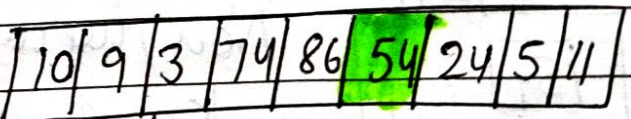




element > peek

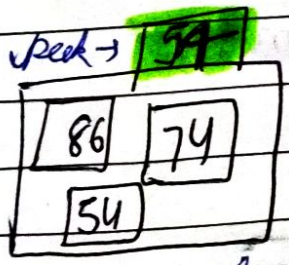
86 > 10 ∴ remove 10 from pq.

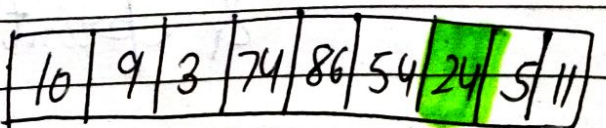




element > peek

54 > 19 ∴ remove 19 from pq.

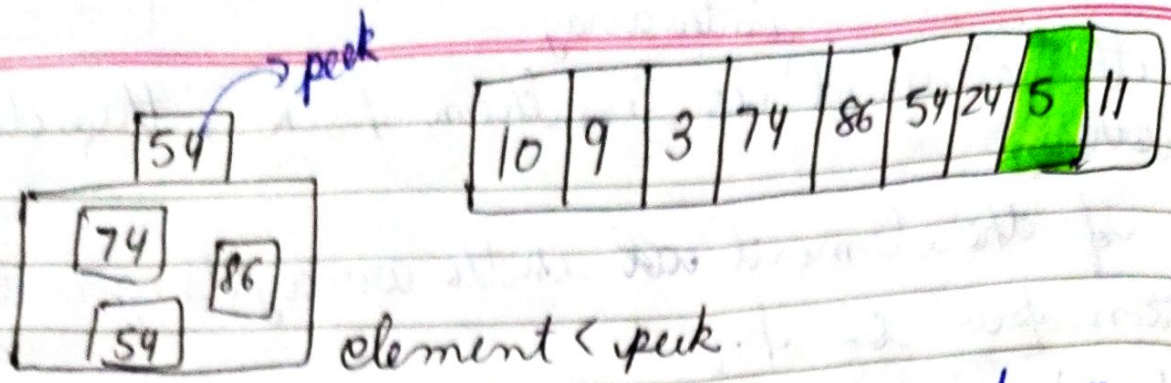




24 < 54

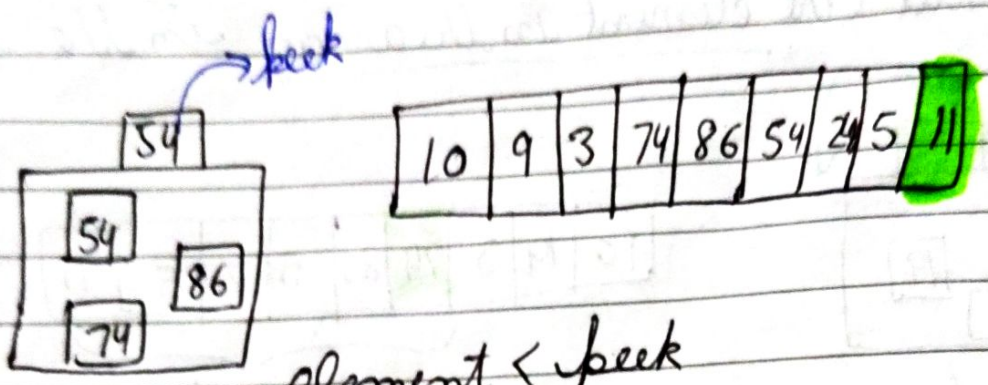
element < peek

So, we will do nothing & move to next element



element < peek

So, we will do nothing and move to next element

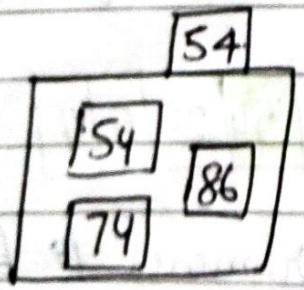


element < peek

So, we will do nothing.

Now, we have traversed on complete array

* → Now, we are left with 3 largest elements in the priority queue.



Now, we will remove K elements from the pq (priority queue) & print them.

→ 54
→ 74
→ 86

```
import java.util.*;
public class Main{
    public static void main(String[] args)
    {
        Scanner scan= new Scanner(System.in);
        int n = scan.nextInt();
        int arr[]= new int[n];
        for(int i=0;i<n;i++)
        {
            arr[i]= scan.nextInt();
        }
        int k = scan.nextInt();
        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 = arr[i];
            if(val > pq.peek())
            {
                pq.remove();
                pq.add(val);
            }
        }
        while(pq.size()>0)
        {
            System.out.println(pq.peek());
            pq.remove();
        }
    }
}
```

Date / /
Page No.

एक array बनाया n size का

array के element हमने input किए

k की value input ली

हमने एक PriorityQueue बनाया where the priority is default (छोटे element को ज्यादा priority दी जाएगी)

pq में k elements add किए

Array के kth index से last index तक सभी elements को pq के peek से compare करेंगे

अगर element की value peek से ज्यादा हुई तो हम peek को pq से remove करदेंगे और element को pq में add करदेंगे

pq में जो भी k elements बचे हैं, उन्हें हम print करेंगे