Priority Queue :-

It is like a queue where the elements are stored based on the priority.

Syntax:-

```
PriorityQueue<Type> pq = new priorityQueue<>();
```

By default, element will be in ascending order.

Methods:-

(i-) add(element) → It is used to add element to the priority queue.

(ii-) peek() → It will return element at front

(iii-) poll() → It removes element from the front.

(iv-) size() → It gives the size of a priority queue.

(v-) remove(element) → It removes the given element.

Add() in priority queue has time complexity as O(log n)

To arrange the elements in descending order

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

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

03 January 2025 20:41

Priority Queue basics

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

public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print out
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        for(int i=0;i<t;i++){
            pq.add(sc.nextInt());
            System.out.println(pq.peek());
        }
    }
}
```

Maximum Product of Two elements in an array

n = 4

arr = [3, 4, 5, 2]

First largest = 5

Second largest = 4

Product = (5-1) * (4-1) = 4 x 3 = 12.

```
pq.add(arr[0]);
pq.add(arr[1]);
  :
  :
pq.add(arr[n-1]);
```

```
PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
for(int i=0; i<n; i++) {
pq.add(arr[i]);
}
int firstlargest = pq.poll();
int secondlargest = pq.poll();
int product = (firstlargest-1) * (secondlargest-1);
S.O.println(product);
```

minimum digits

n = 6

6, 8, 4, 5, 2, 3

684
523

658
423

586
324

We have to get minimum possible sum

2, 3, 4, 5, 6, 8

234
568

802

246
358

604

246 ← num1
358 ← num2

num1 = 0, num2 = 0;

num1 = num1*10 + pq.poll();

= 0*10 + 2 = 2

num2 = num2*10 + pq.poll()

= 0*10 + 3 = 3

num1 = 2*10 + 4 = 24

num2 = 3*10 + 5 = 35

num1 = 24*10 + 6 = 246

num2 = 35*10 + 8 = 358

S.O.P ln (num1 + num2);

int num1=0, num2=0;

while(!pq.isEmpty()) {

num1 = num1*10 + pq.poll();

if(!pq.isEmpty()) {

num2 = num2*10 + pq.poll();

}

Code :-

```
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
for(int i=0; i<n; i++) {
  pq.add(sc.nextInt());
  }
int num1=0, num2=0;
while(!pq.isEmpty()) {
num1 = num1*10 + pq.poll();
if(!pq.isEmpty()) {
num2 = num2*10 + pq.poll();
  }
}
S.o.pln(num1+num2);
```

Optimize while loop

```
int eo=0;
while(!pq.isEmpty()){
    if(eo%2==0){
    num1=num1*10+pq.poll();
    }else{
     num2=num2*10+pq.poll();
    }
    eo++;
}
```

0007

00+07=7

0145=

04

15

Substract numbers1

n = 5

arr = [1, 5, 0, 3, 5]

1. Create a priority queue and add all the non-zero elements.

pq.add( ) ← 1, 5, 3, 5

[1, 3, 5, 5] →

↳ ① [1, 1, 3, 5, 5]

↓ ↓

0 0

no of operation ←

↓

1.

[3, 5, 5]

↓

3

[0, 5, 5]

operation = 2

[5, 5]

↓

5

[5, 5]

↓ ↓

0 0

operation = 3

5
operation-3 0 0

[0, 1, 1, 3, 5, 5] → (1, 1, 3, 5, 5)

1. 1st operation: 1 2 3

[0, 0, 0, 3, 5, 5]

3 2nd operation

[0, 0, 0, 0, 5, 5]

5 3rd operation.

[0, 0, 0, 0, 0, 0].

1, 1, 0, 3, 5, 5

pq → [1, 1, 3, 5, 5]

1 3 5

1st 2nd 3rd.

Using HashSet —

HashSet<Integer> hs = new HashSet<>();

```
HashSet<Integer> hs = new HashSet<>();
for(int i=0; i<n; i++) {
    if(arr[i] > 0) {
        hs.add(arr[i]);
    }
}
S.o.pln(hs.size());
```

Using PriorityQueue.

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

for(int i=0; i<n; i++) {
    if(arr[i] > 0) {
        pq.add(arr[i]);
    }
}
int op = 0;
while(!pq.isEmpty()) {
int value = pq.poll();
while(!pq.isEmpty() && pq.peek() == value) {
    pq.poll();
```

1, 1, 0, 5, 3, 5

pq → [1, 1, 3, 5, 5]

→ 1

pq.poll();
}
op++;
}

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

public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print out
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        for(int i =0;i<n;i++){
            int t = sc.nextInt();
            if(t>0){
                pq.add(t);
            }
        }

        int op =0;
        while(!pq.isEmpty()){
            int val = pq.poll();
            while(!pq.isEmpty()&& pq.peek()==val){
                pq.poll();
            }
            op++;
        }
        System.out.println(op);
    }
}
```