

Ques

First Negative Sliding window

```

Queue<Integer> que = new ArrayDeque();
for(int i=0; i<k; i++){
    if(arr[i]<0){
        que.offer(arr[i]);
    }
}

if(!que.isEmpty()){
    System.out.print(que.peek()+" ");
}else{
    System.out.print("0"+" ");
}

for(int i=k; i<arr.length; i++){
    if(!que.isEmpty() && que.peek()==arr[i-k]){ // to remove element outside window
        que.poll();
    }
    if(arr[i]<0){ // add only -ve value
        que.offer(arr[i]);
    }
    if(!que.isEmpty()){ // print first -ve value of window
        System.out.print(que.peek()+" ");
    }else{
        System.out.print("0"+" ");
    }
}
/* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution */

```

Diagram illustrating the sliding window process for k=3:

Initial array: 1, -2, 3, -4, -5, 6, 7, -8, 9

Step 1: Initial window [1, -2, 3]. Negative values: -2, 3. Queue: [-2, 3].

Step 2: Slide to [-2, 3, -4]. Negative values: -2, 3, -4. Queue: [-2, 3, -4].

Step 3: Slide to [3, -4, -5]. Negative values: -4, -5. Queue: [-4, -5].

Step 4: Slide to [-4, -5, 6]. Negative values: -4, -5. Queue: [-4, -5].

Step 5: Slide to [-5, 6, 7]. Negative values: -5. Queue: [-5].

Step 6: Slide to [6, 7, -8]. Negative values: -8. Queue: [-8].

Step 7: Slide to [7, -8, 9]. Negative values: -8. Queue: [-8].

Final output sequence: -2, -2, -4, -4, -5, -8, -8

- ① push all -ve values of k slide
- ② print first -ve value for that slide
- ③ pop -ve value outside from my bucket
- ④ add value to queue if -ve.
- ⑤ print peek(); if 0;

Priority Queue / Heap

[0, -3, 5, -2, 8, 9]

1) min pq

2) max pq

g { pq.poll()
.remove()

pq



pq.poll() } -3
pq.remove()

min pq

Syntax

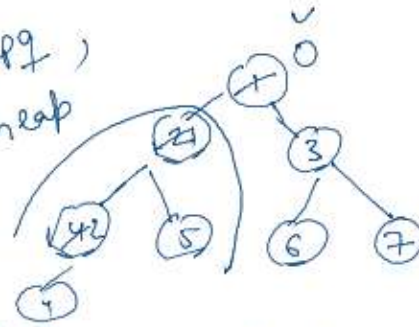
PriorityQueue<Integer> pq = new PriorityQueue<>();
PriorityQueue<Integer> pq = new PriorityQueue<>(
Collections.reverseOrder());

[[minHeapify()]]

1 2 3 4 5 6 7

$O(\log n)$

min pq,
min heap



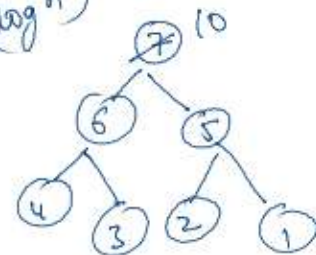
pq.peek() = 0
pq.poll() = 0
.remove()

pq.add(0)

collections.
reverseOrder();

[[maxHeapify()]]

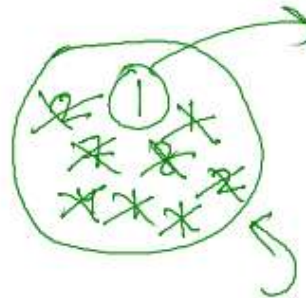
$O(\log n)$



10

Ques Stone game

[2 7 4 1 8 1]
0 1 2 3 4 5



```
public static void main(String[] args) {  
    Scanner scn = new Scanner(System.in);  
    int n = scn.nextInt();  
    int[] arr = new int[n];  
    for(int i=0; i<n; i++){  
        arr[i] = scn.nextInt();  
    }
```

[2 7 4 1 8 1]

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

```
    for(int num : arr){  
        maxPQ.add(num);  
    }
```

```
    while(maxPQ.size() > 1){  
        int a = maxPQ.poll();  
        int b = maxPQ.poll();  
  
        if(a != b){  
            int val = Math.abs(a - b);  
            maxPQ.add(val);  
        }  
    }
```

```
    System.out.println(maxPQ.size() > 0 ? maxPQ.peek() : "0");  
    /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class shc  
    }
```



a = 8 4 2 1 1
b = 7 8 1 1 1
#

Ques weak soldiers

$k=3$

①	1	1	0	0	0
②	1	1	1	0	0
③	1	1	0	0	0
④	1	1	1	1	0
⑤	1	0	0	0	0

0's = civilians
1's = soldiers

