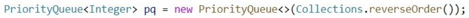
Heap → (Priority Queue)

Queue ×
↳ FIFO ×

→ processing → priority

→ priority → (min priority) (rank list)
→ (max priority) (total score)

→ By Default = (min - priority) → Integer



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

→ Inverse Nature ⇒ Collections.reverseOrder()

operations

add ()] logn
remove ()

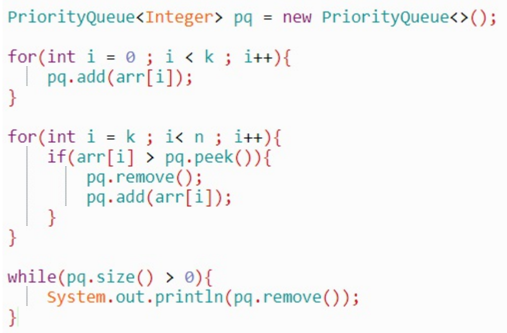
peek ()] ⇒ o(1)
size ()

PriorityQueue <> Pq = new PriorityQueue();

Example

```
PriorityQueue<Integer> PQ = new PriorityQueue()
PQ.add(10);
PQ.add(5);
PQ.add(20);
PQ.add(-1);
// -1 5 10 20
```

" 24071998 " → 24071998 % 10000 ⇒ 1998



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

for(int i = 0 ; i < k ; i++){
    pq.add(arr[i]);
}

for(int i = k ; i< n ; i++){
    if(arr[i] > pq.peek()){
        pq.remove();
        pq.add(arr[i]);
    }
}

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

K = 4

Min Priority

100

100

4

12	62	22	15	37	99	11	37	98	67	31	84	99

4

~~12~~ 37

62

22

15

logn   logk

O(nlogk)

S.C. → O(k)

k <= n

84

98

99

99

logk

n log k

n. (2 logk)

⇓

2n logk ⇒ O(nlogk)

Max Priority

n

12 99 31

62 11 ~~84~~

22 37 ~~99~~

15 ~~98~~ 99

37 67

1 → 99

2 → 99

3 → 98

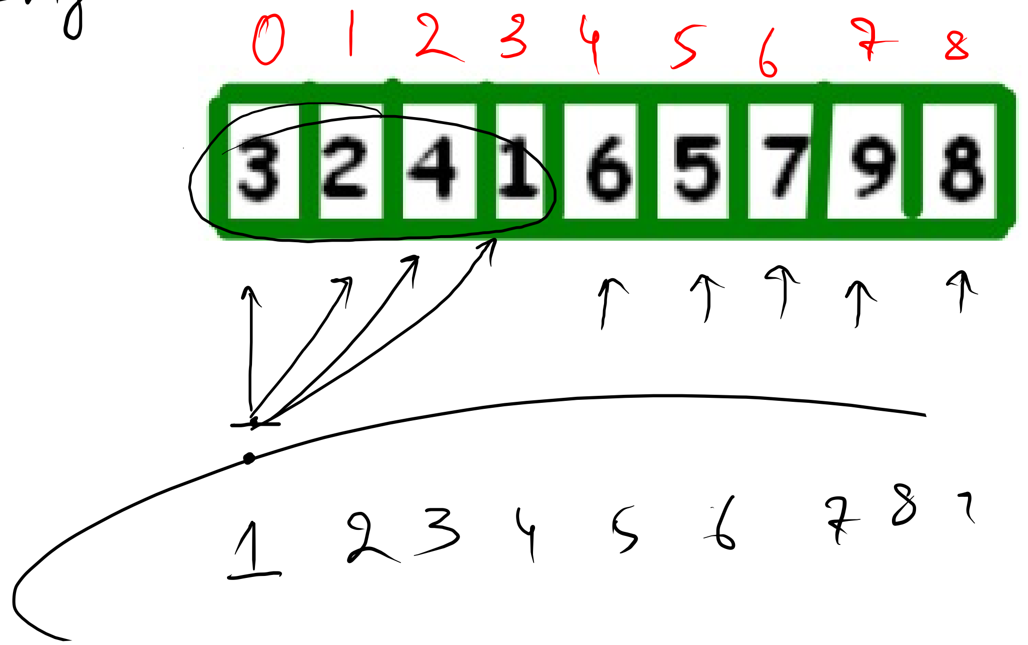
4 → 84 → Ans

T.C. → nlogn → n logk

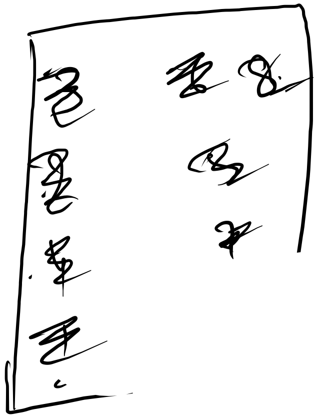
S.C. → n → k

ele => atmost k idx shifted from sorted position

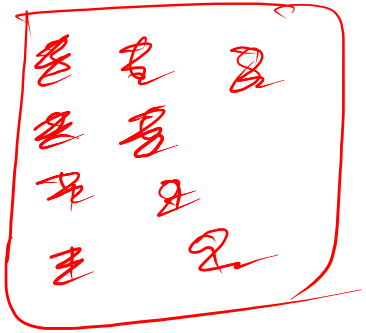
K=3

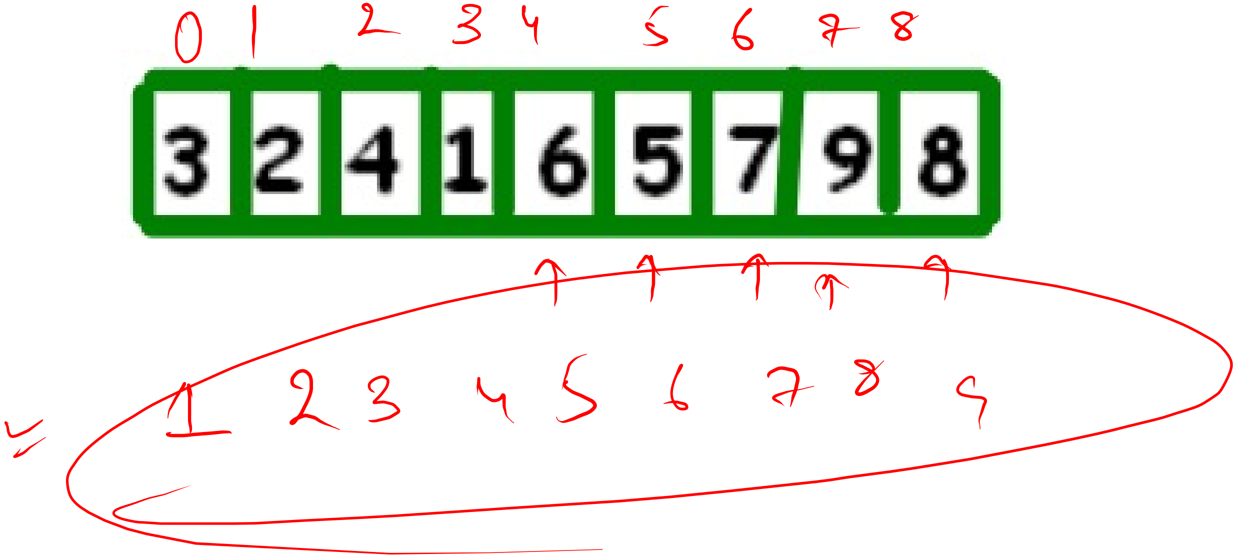


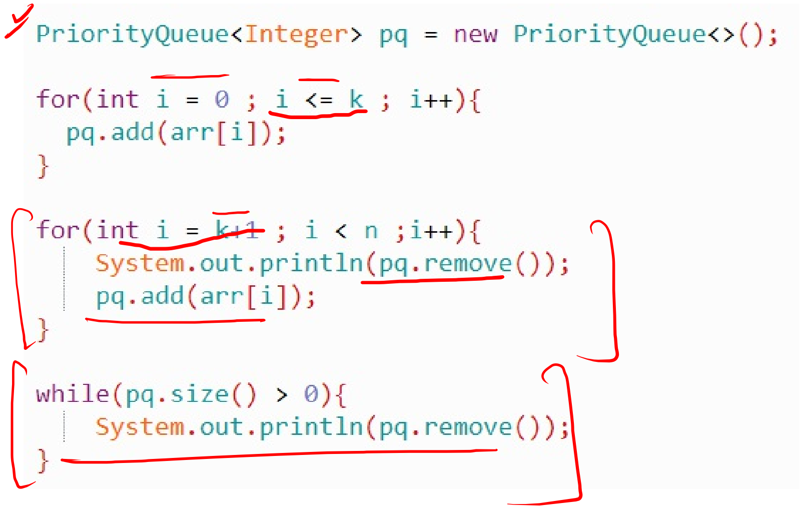
0	1	2	3	4	5	6	7	8
3	2	4	1	6	5	7	9	8



k=3



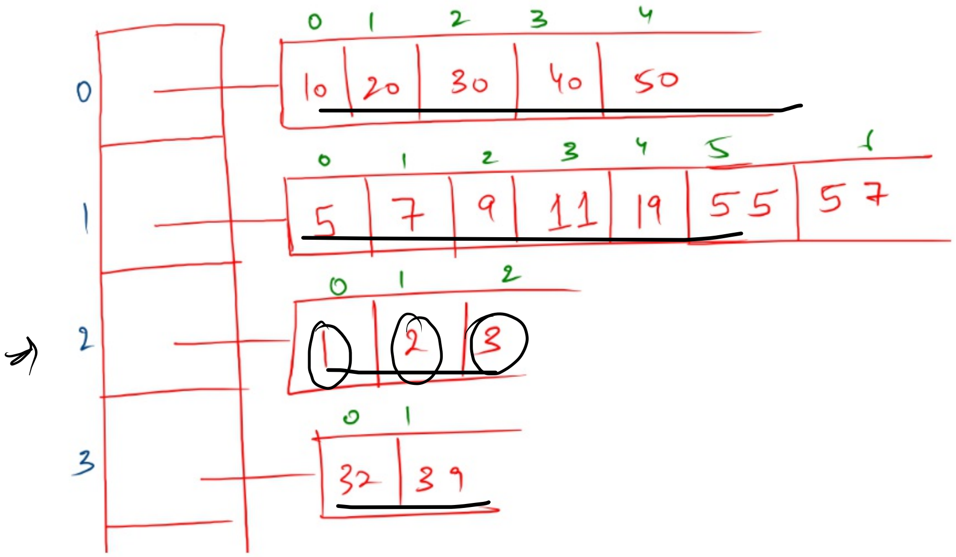




```
PriorityQueue<Integer> pq = new PriorityQueue<>();

for(int i = 0 ; i <= k ; i++){
  pq.add(arr[i]);
}

for(int i = k+1 ; i < n ;i++){
    System.out.println(pq.remove());
    pq.add(arr[i]);
}

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

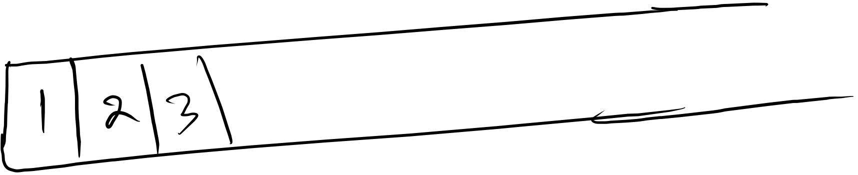
O(k)

Lists =

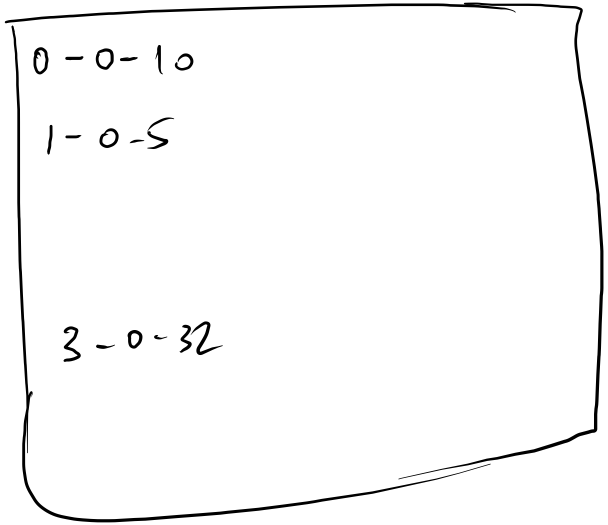


Time → O(N log k)

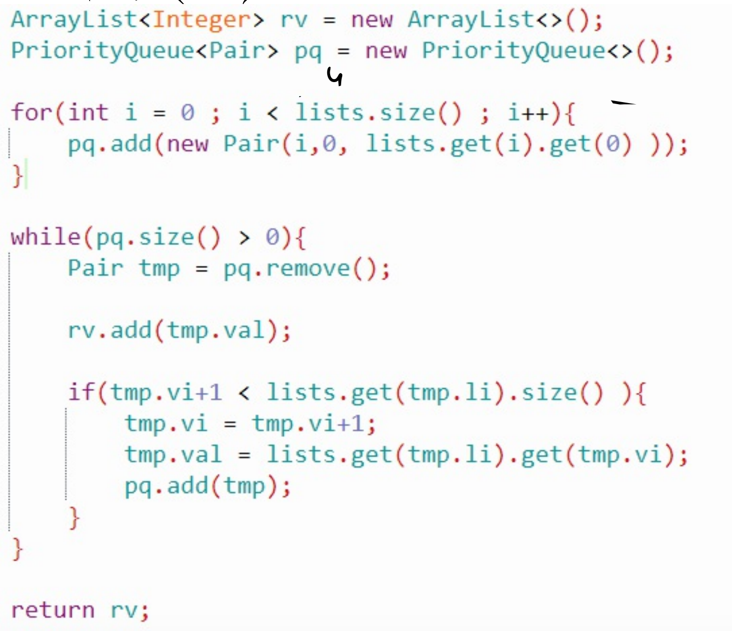
n



tmp = 2-2-3



Pair<Li, vi, val>



```
ArrayList<Integer> rv = new ArrayList<>();
PriorityQueue<Pair> pq = new PriorityQueue<>();

for(int i = 0 ; i < lists.size() ; i++){
    pq.add(new Pair(i,0, lists.get(i).get(0) ));
}

while(pq.size() > 0){
    Pair tmp = pq.remove();

    rv.add(tmp.val);

    if(tmp.vi+1 < lists.get(tmp.li).size() ){
        tmp.vi = tmp.vi+1;
        tmp.val = lists.get(tmp.li).get(tmp.vi);
        pq.add(tmp);
    }
}

return rv;
```