C → 2.

a → 2.

b → 2

Heap ⇒ (Priority Queue)

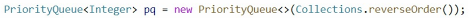
Queue ×
↳ FIFO ×

processing → priority

priority → (min priority) (ranklist)

priority → (max priority) (total score)

By Default = (min - priority) → Integer



```
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);

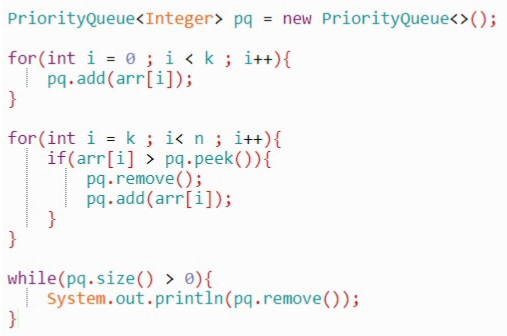
PQ.add(5);

PQ.add(20);

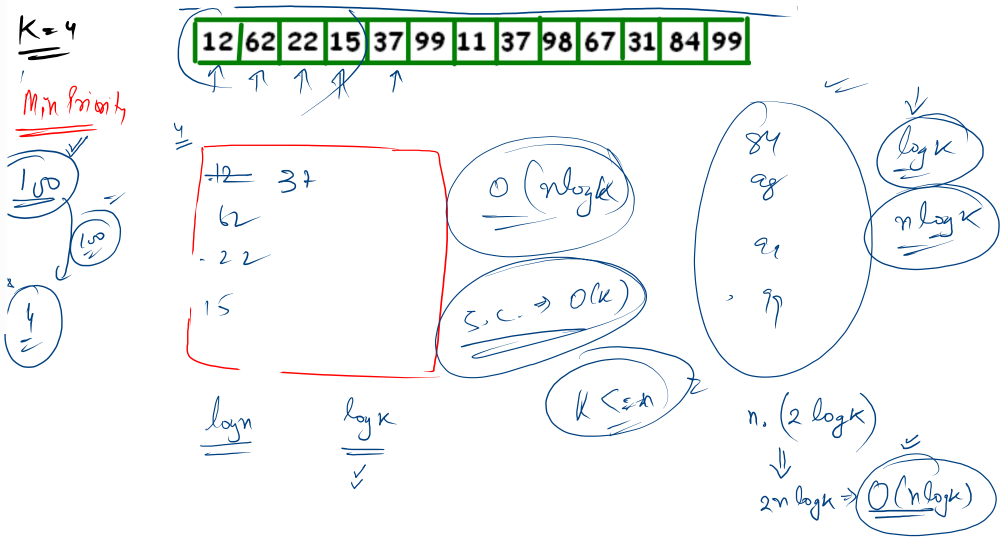
PQ.add(-1);

// -1 5 10 20

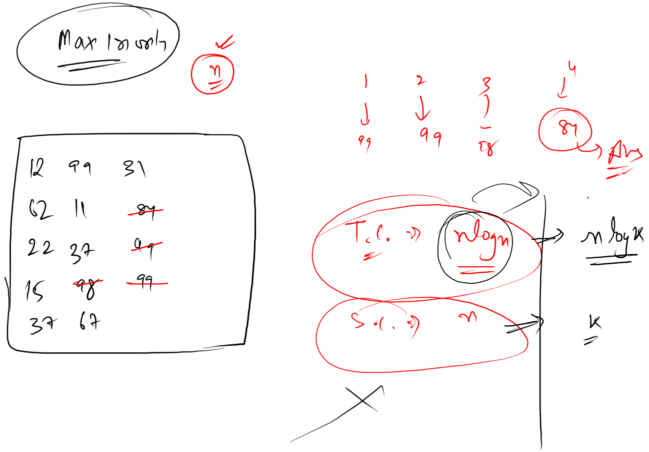
"24071998" → 24071998 % 10000 ⇒ 1998



```
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());
}
```

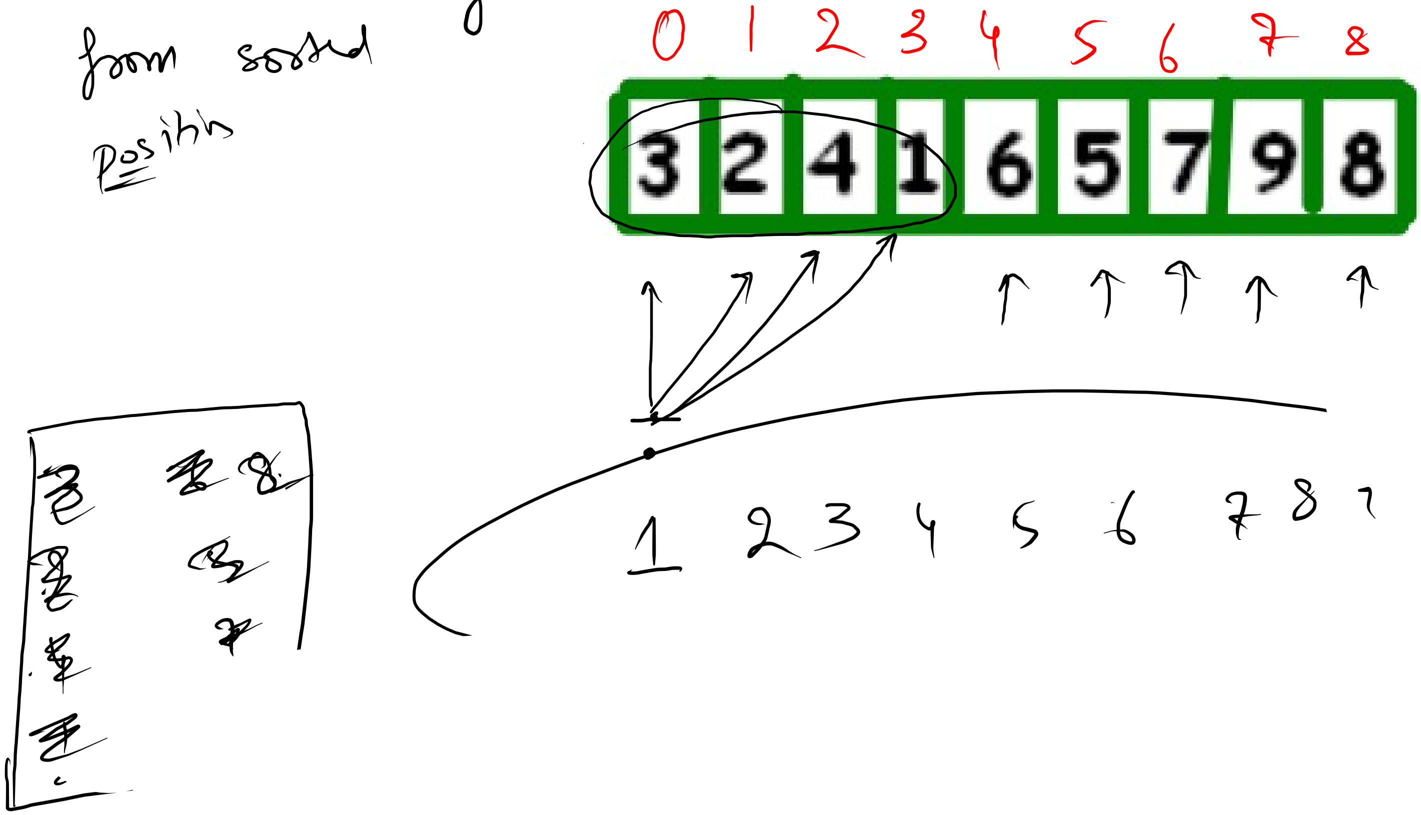


12 62 22 15 37 99 11 37 98 67 31 84 99

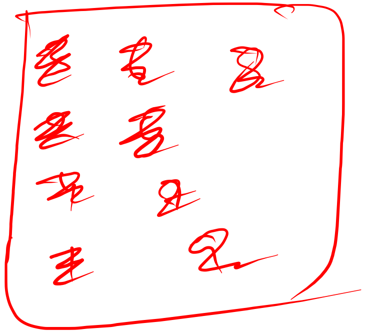


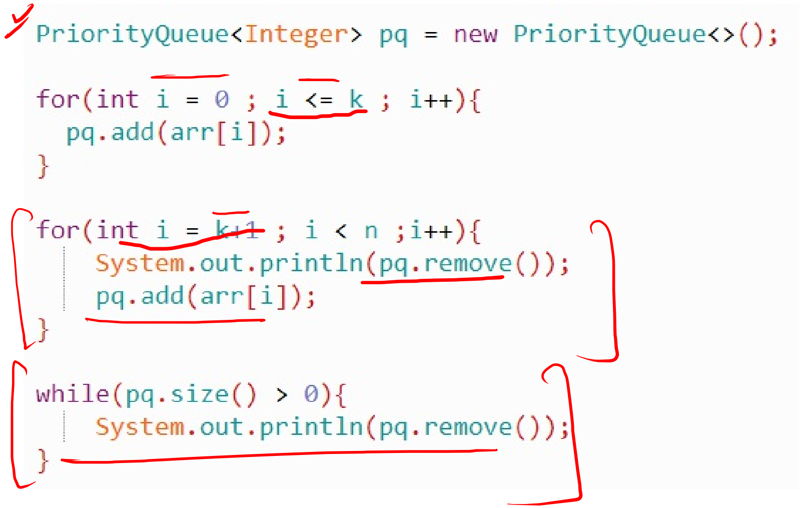
ele => atmost k idx shifted from sorted position

K=3

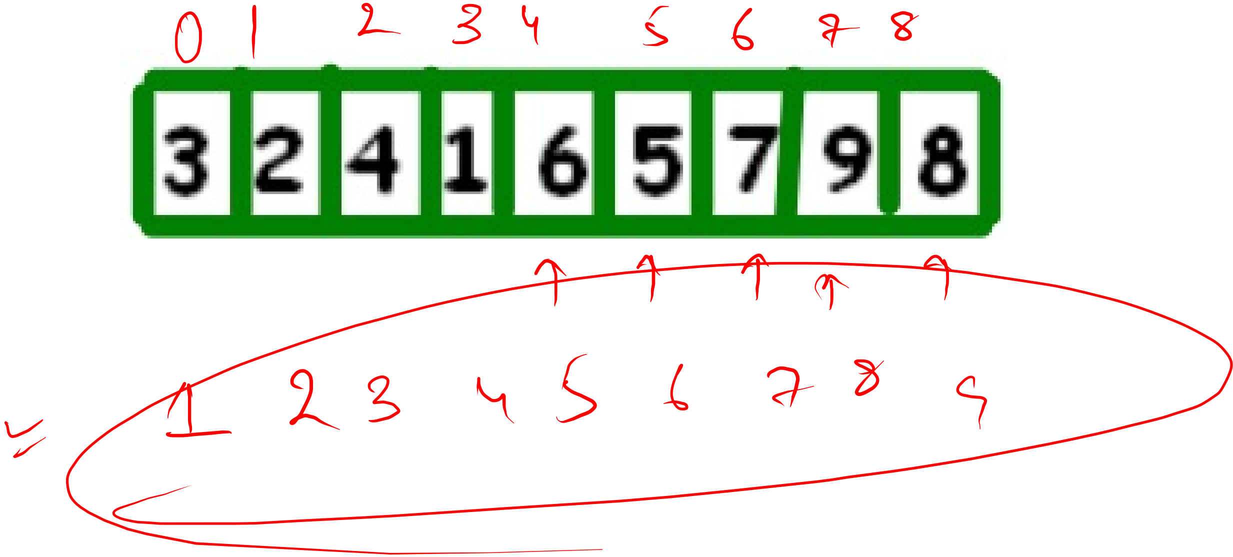


k=3

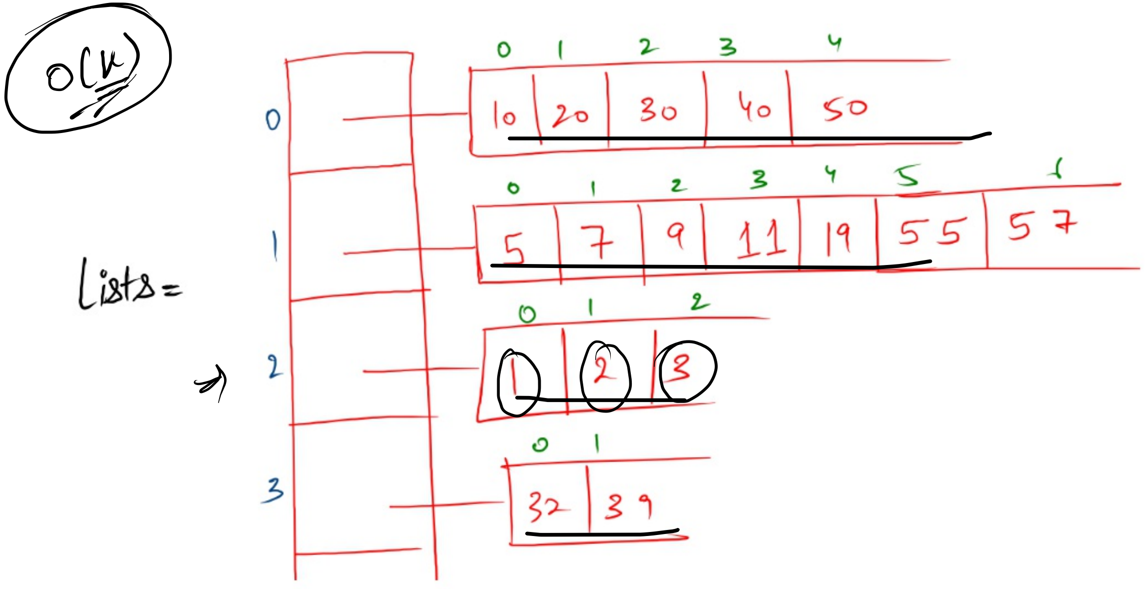


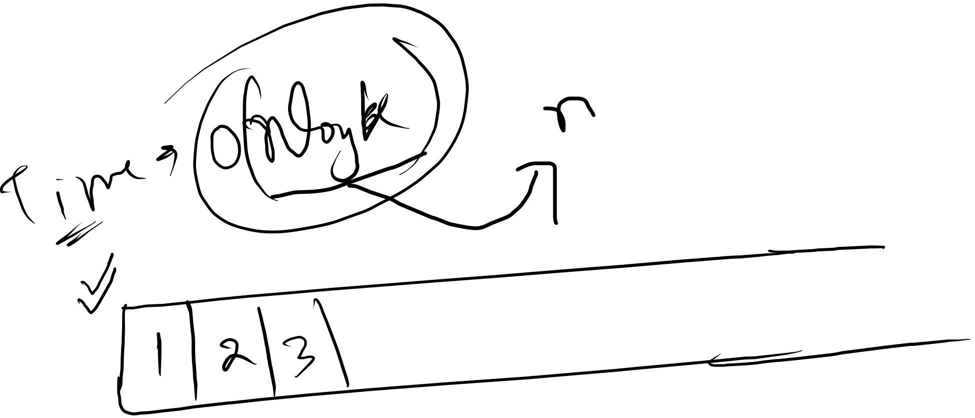


```
PriorityQueue<Integer> pq = new PriorityQueue<>();

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());
}
```

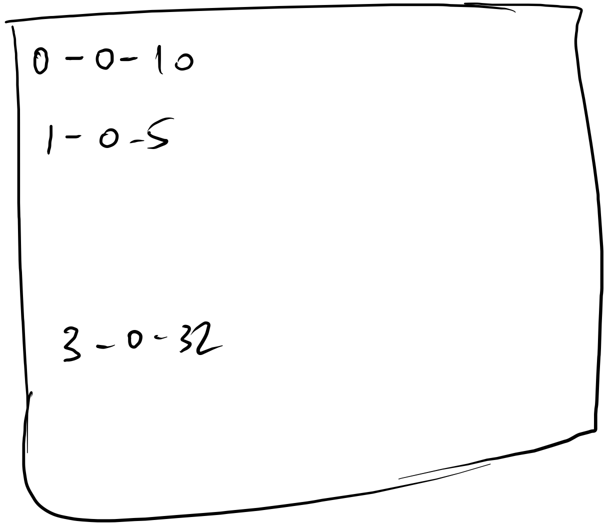


O(k)

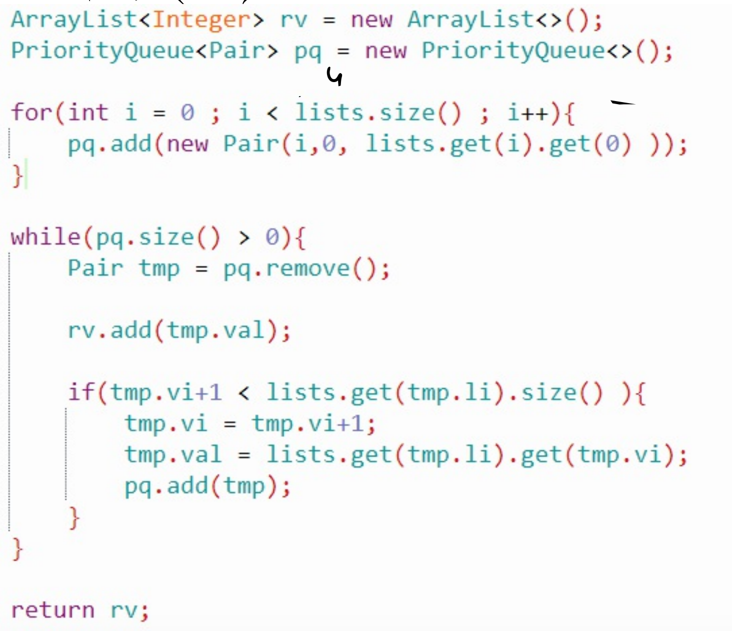




tmp = 2-2-3



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;
```