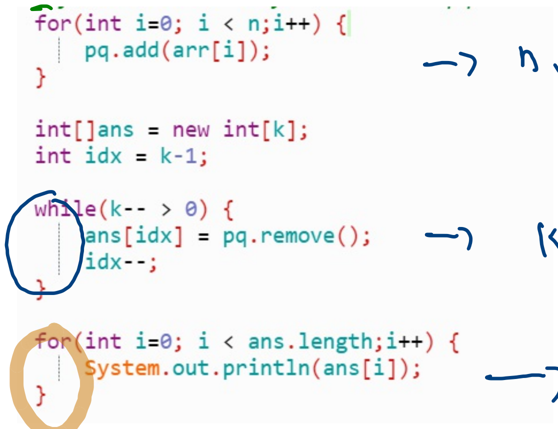
n : no. of elements
in pq.

add -> log n

remove -> logn

peek -> o(1)

25 6 19 8 7 24 15 39 11



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

int[]ans = new int[k];
int idx = k-1;

while(k-- > 0) {
    ans[idx] = pq.remove();
    idx--;
}

for(int i=0; i < ans.length;i++) {
    System.out.println(ans[i]);
}
```

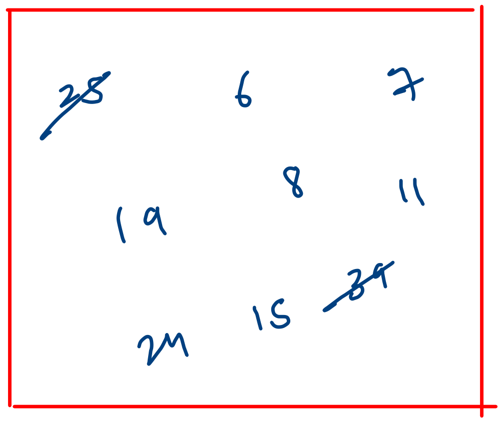
-> n logn

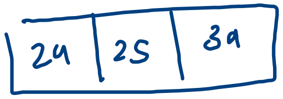
+

-> k logn

+

-> k

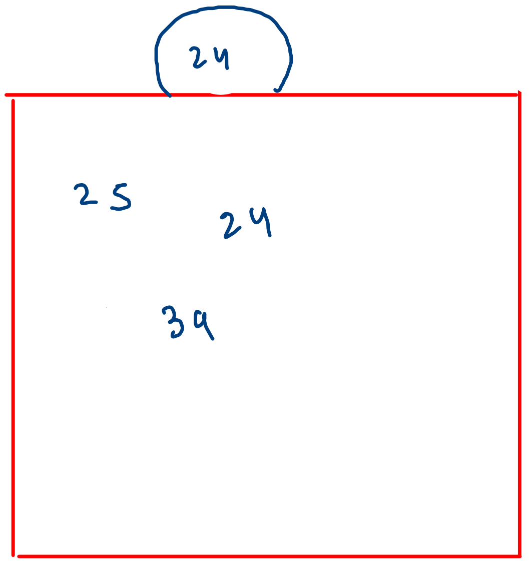




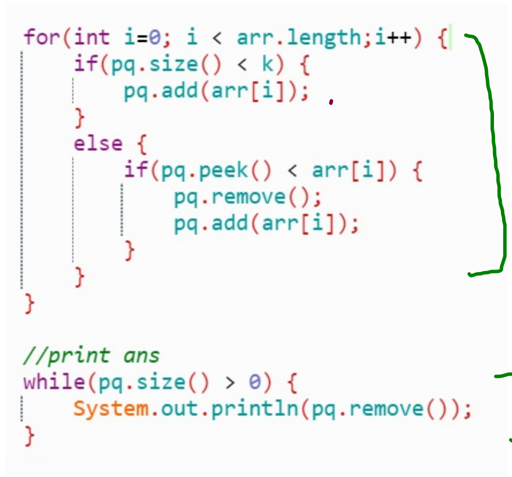
nlogk

k = 3

25 6 19 8 7 24 15 39 11



24 25 39



```
for(int i=0; i < arr.length;i++) {
    if(pq.size() < k) {
        pq.add(arr[i]);
    }
    else {
        if(pq.peek() < arr[i]) {
            pq.remove();
            pq.add(arr[i]);
        }
    }
}

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

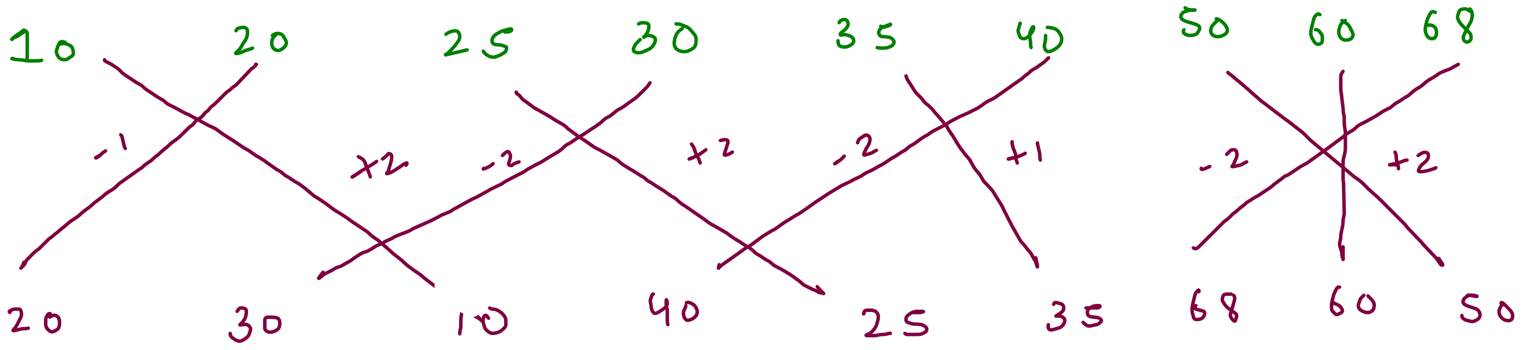
nlogk

klogk

K-sorted array

k = 2

k = 2

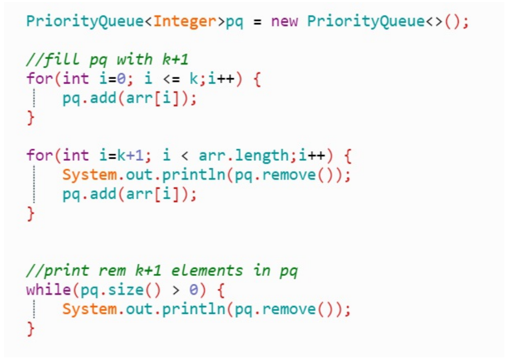


Input ; 20 30 10 40 25 35 68 60 50

input : 20 30 10 40 25 35 50 68 60

~~50~~, ~~60~~
~~68~~

10 20 25 30 35 40 50 60 68



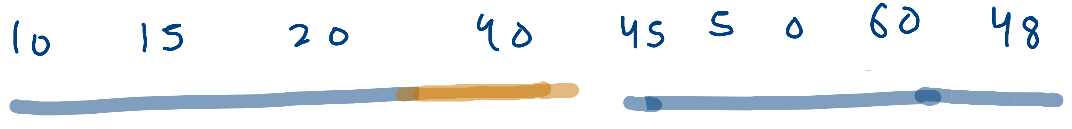
```java
PriorityQueue<Integer>pq = new PriorityQueue<>();

//fill pq with k+1
for(int i=0; i <= k;i++) {
    pq.add(arr[i]);
}

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


//print rem k+1 elements in pq
while(pq.size() > 0) {
    System.out.println(pq.remove());
}
```

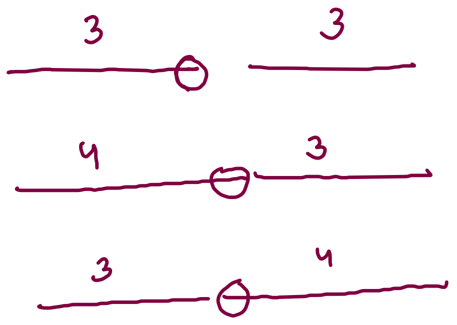
5 9 2 10 1 12 6

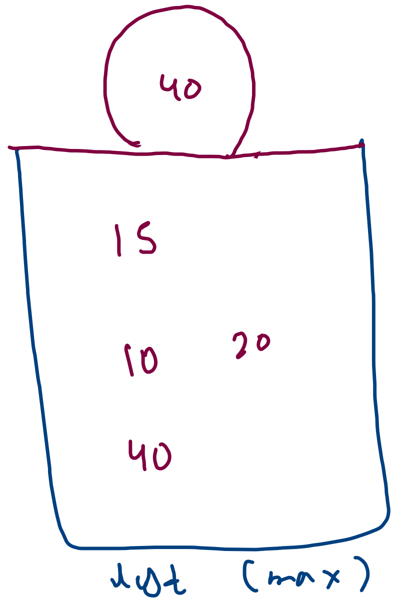
mpq. peek() → median

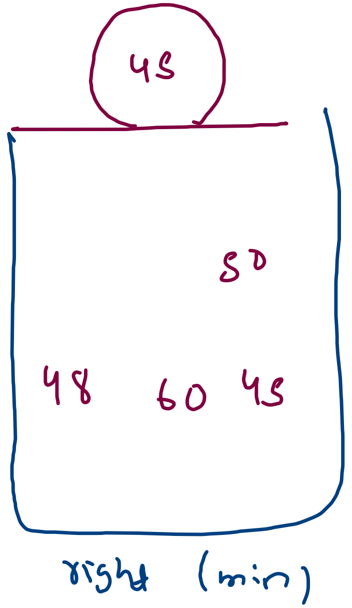
mpq. add() → addition

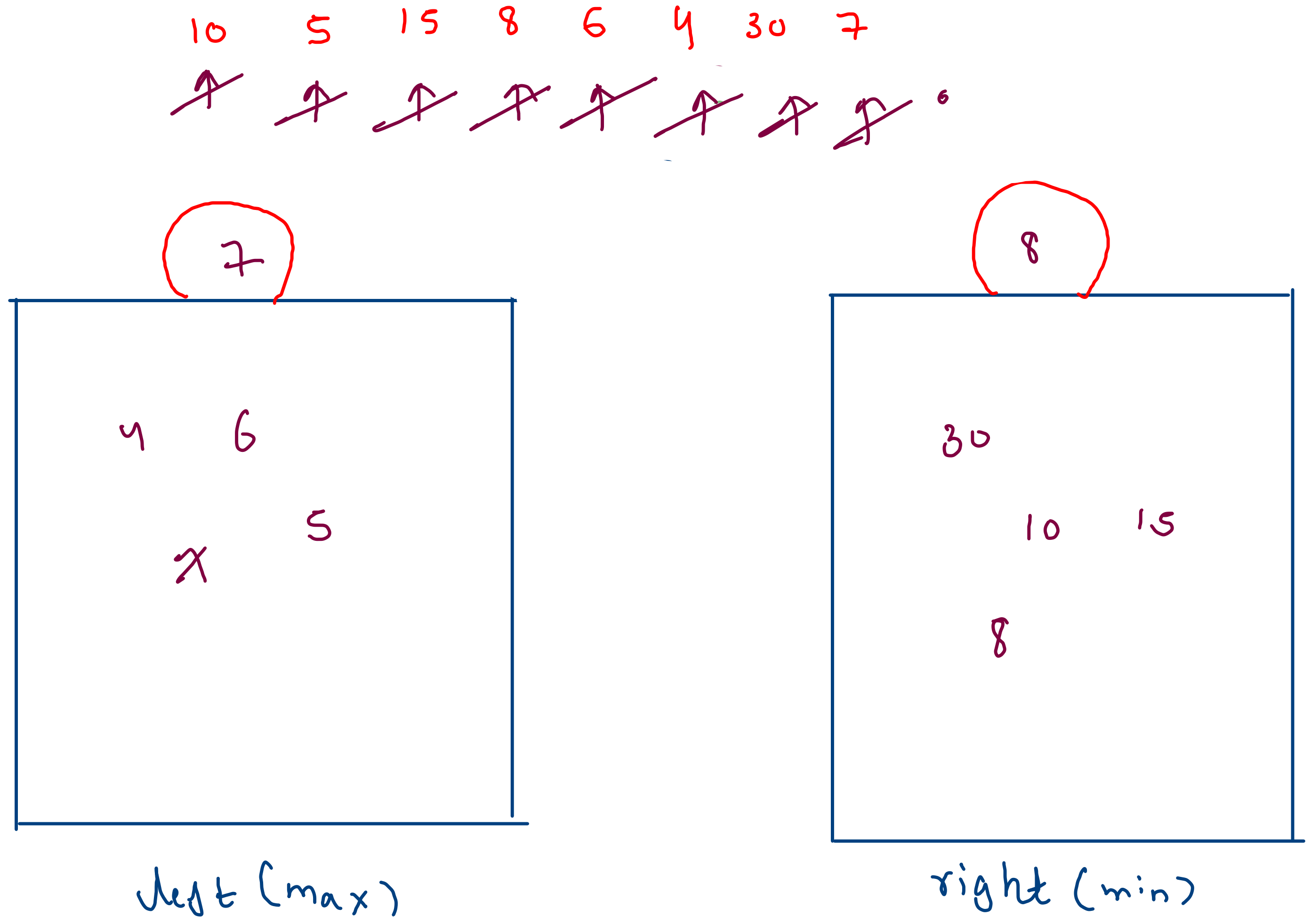
mpq. remove() → remove



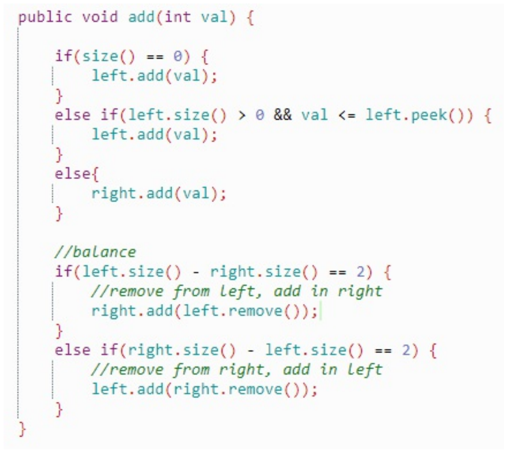




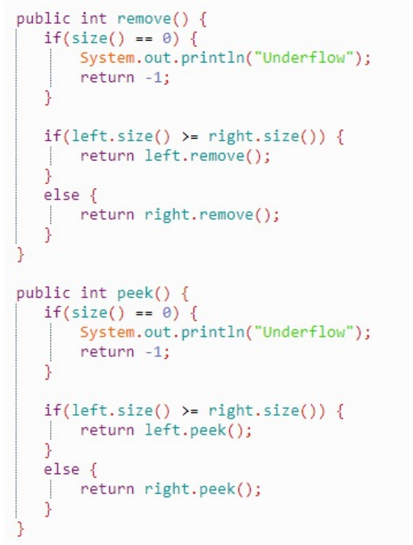




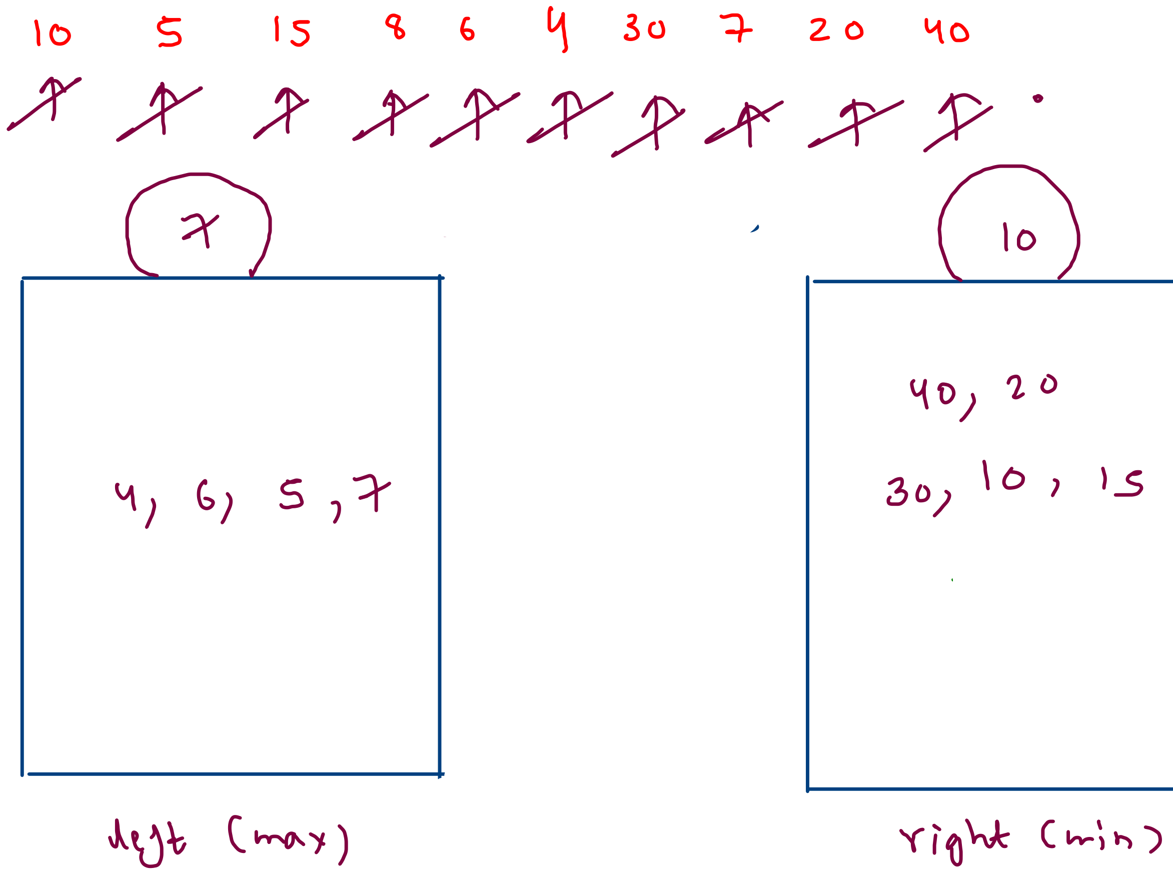
10 5 15 8 6 4 30 7
7
4 6
5
~~7~~
left (max)
8
30
10 15
8
right (min)



```java
public void add(int val) {

    if(size() == 0) {
        left.add(val);
    }
    else if(left.size() > 0 && val <= left.peek()) {
        left.add(val);
    }
    else{
        right.add(val);
    }

    //balance
    if(left.size() - right.size() == 2) {
        //remove from left, add in right
        right.add(left.remove());
    }
    else if(right.size() - left.size() == 2) {
        //remove from right, add in left
        left.add(right.remove());
    }
}
```



```java
public int remove() {
   if(size() == 0) {
       System.out.println("Underflow");
       return -1;
   }

   if(left.size() >= right.size()) {
       return left.remove();
   }
   else {
       return right.remove();
   }
}

public int peek() {
   if(size() == 0) {
       System.out.println("Underflow");
       return -1;
   }

   if(left.size() >= right.size()) {
       return left.peek();
   }
   else {
       return right.peek();
   }
}
```



k = 4

l0 → 2 5 7 9 11 •

l1 → 1 3 13 19

l2 → 5 12 16 18 20

l3 → 4 6 10 •

Pair:

val

idx

li

(val - idx - li)

PQ

~~(2 - 0 - 0)~~ ~~(5 - 1 - 0)~~

~~(1 - 0 - 1)~~ ~~(3 - 1 - 1)~~

~~(5 - 0 - 2)~~ (13 - 2 - 1)

~~(4 - 0 - 3)~~ ~~(7 - 2 - 0)~~

~~(6 - 1 - 3)~~ (12 - 1 - 2)

~~(10 - 2 - 3)~~ ~~(9 - 3 - 0)~~

~~(11 - 4 - 0)~~

ml → 1 2 3 4 5 5 6 7 9 10 11

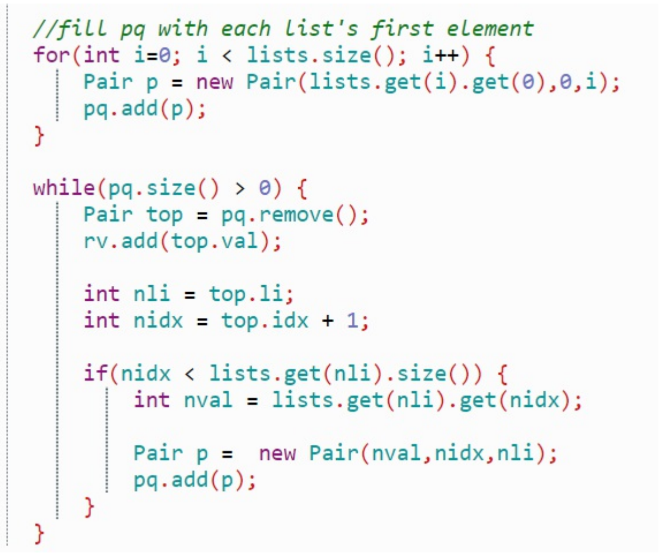
l0 → 2 7 9 11

l1 → 1 3 19

l2 → 5 12 16 18 20

l3 → 4 6 10

(val- idx - li)



```
//fill pq with each list's first element
for(int i=0; i < lists.size(); i++) {
    Pair p = new Pair(lists.get(i).get(0),0,i);
    pq.add(p);
}

while(pq.size() > 0) {
    Pair top = pq.remove();
    rv.add(top.val);

    int nli = top.li;
    int nidx = top.idx + 1;

    if(nidx < lists.get(nli).size()) {
        int nval = lists.get(nli).get(nidx);

        Pair p =  new Pair(nval,nidx,nli);
        pq.add(p);
    }
}
```

1 2 3 4 5 6

7 9 10 11 12 16

18 19 20

merged list.

~~2 - 0 - 0~~ ~~3 - 1 - 1~~

~~1 - 0 - 1~~ ~~7 - 1 - 0~~

~~5 - 0 - 2~~ ~~19 - 2 - 1~~

~~4 - 0 - 3~~ ~~6 - 1 - 3~~

~~12 - 1 - 2~~ ~~10 - 2 - 3~~

~~9 - 2 - 0~~

~~11 - 3 - 0~~ ~~16 - 2 - 2~~

~~18 - 3 - 2~~ ~~20 - 4 - 2~~