Priority Queue :

usage → questions

class + pq + comparable

Interface with oops.

pq. function →

	Complexity
add / push →	O(log n)
remove/ pop →	O(log n)
top / peek →	O(1)
size →	O(1)
isEmpty → True / false	O(1)

why it is log(n)]x → Creation

removal is done on priority Element ← remove/ pop

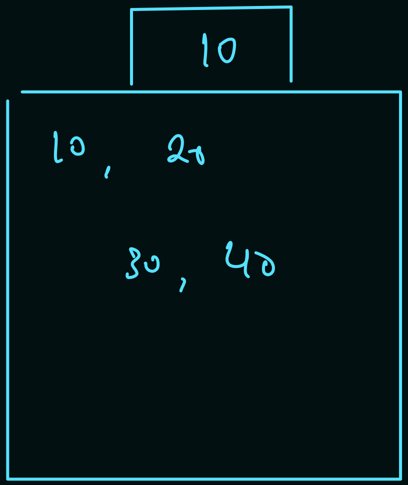
→ priority → min

pq.add(10);

20;

30;

40;



K Largest Elements: k= 3

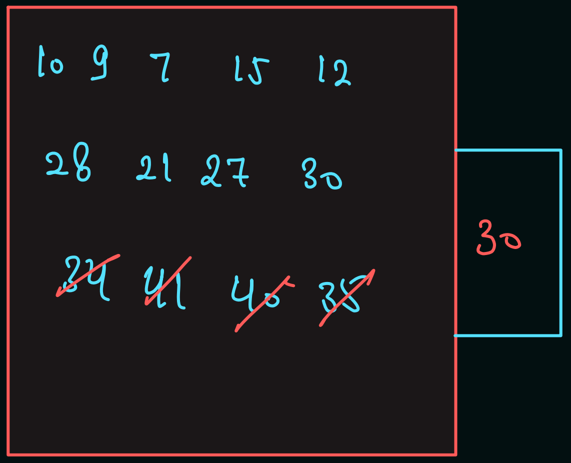
array→ 10 9 7 15 12 28 21 27 30 34 41 40 35

Approach-1

① Add all Element in pq.

② print top k Elements.

Time Complexity → nlogn



max-priority

41
40
35
34
} k largest

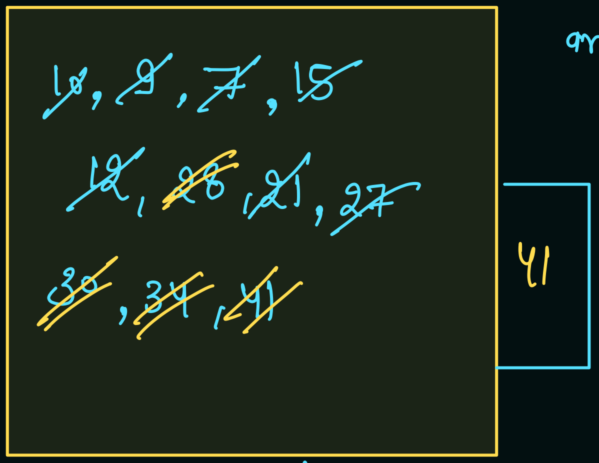
Approach-2

① Add k Elements in min priority queue.

② Add next Element in priority queue if it is greater than peek Element & remove peek Element.

T→ nlogk

k=4



min-priority Queue.

array→ 10 9 7 15 12 28 21 27 30 34 41 4 23

klogk

skip skip.

decreasing order→

28 30 34 41

k Largest Element] Increasing order

using Stack

↳ order → Decreasing.

Time→ nlogk

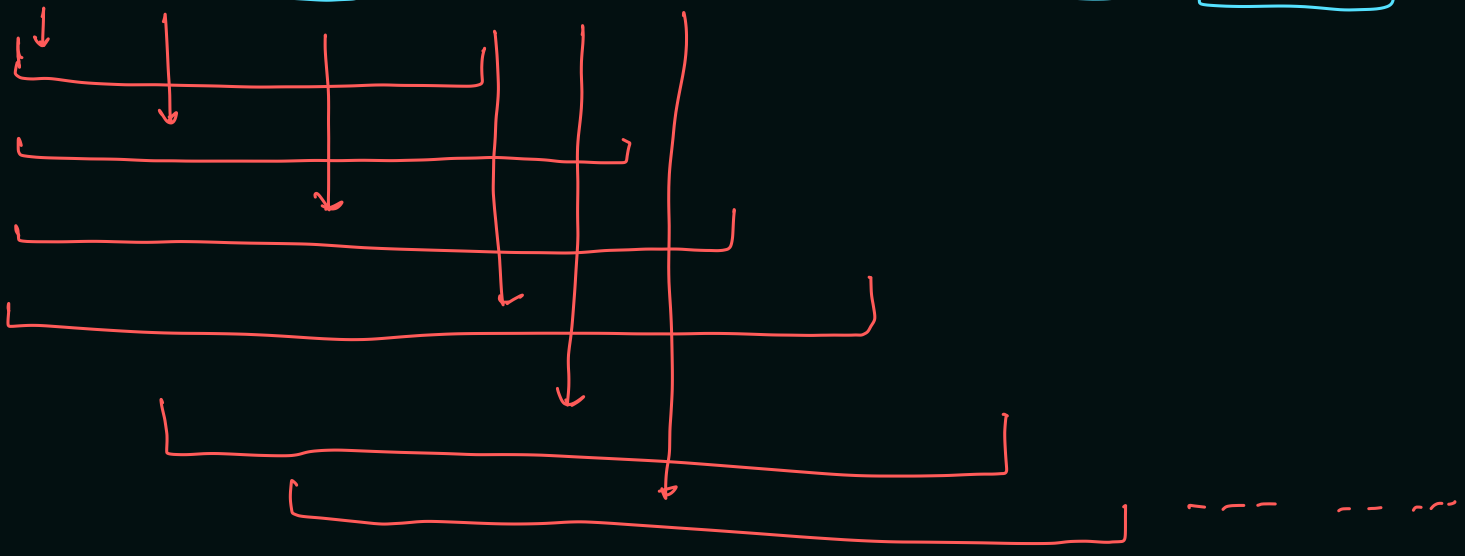
Sort K-sorted array: k=3

Element is spot k-distance either left or either Right from its actual position

↓

these type of array is known as k-sorted arry.

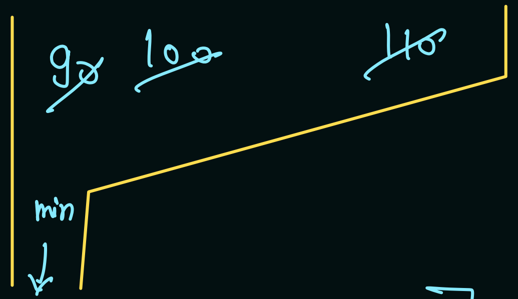
Index	0	1	2	3	4	5	6	7	8	9	10
Sorted array.	10	20	30	40	50	60	70	80	90	100	110



k=3

0	1	2	3	4	5	6	7	8	9
30	40	20	10	80	60	70	100	110	90

min-pq.



Time Complexity

↳ nlogk

✓10 ✓20 ✓30 40 60 70 80 90 100 110] ← sort

Median Priority Queue:

no. of Elements → odd → middle value is median

Even → first mid is median.

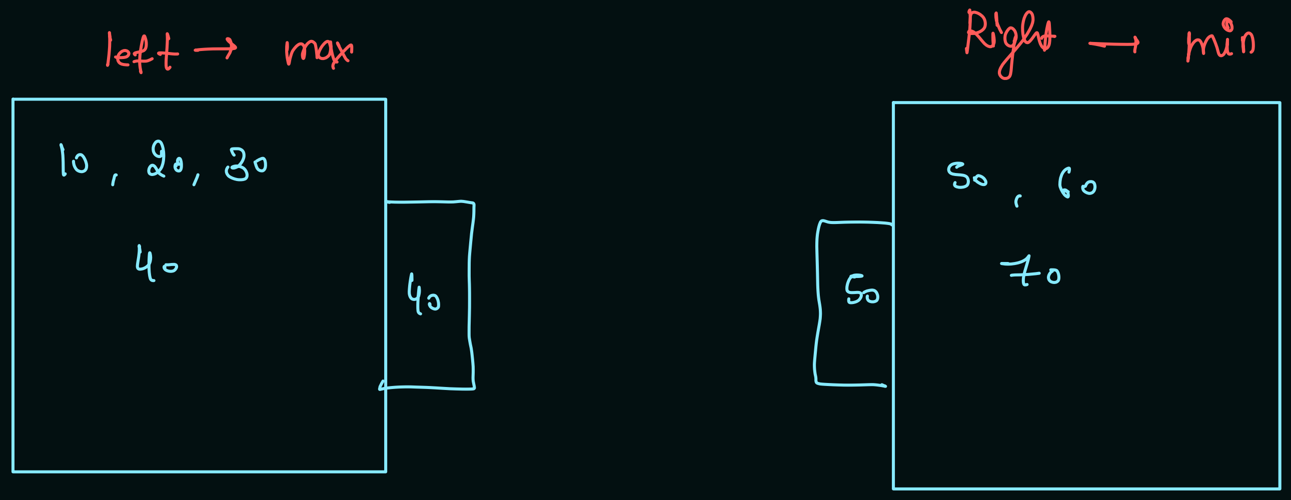
Ex 1 → 10 20 30 40 50 60 → Even → 30 median.

30: first mid

40: Second Mid

Ex 2 - 10 20 30 $\boxed{40}$ 50 60 70 → odd → 40 median

40: mid



add

remove

peek

size

10 ✓

20 ✓

30 ✓

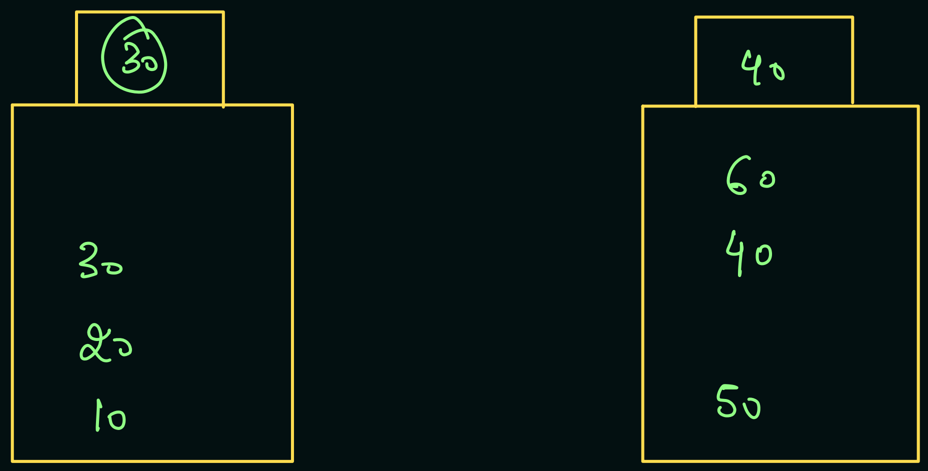
40 ✓

50 ✓

60 ✓

$\boxed{70}$

→ size diff. b/w left & Right ≤ 1.



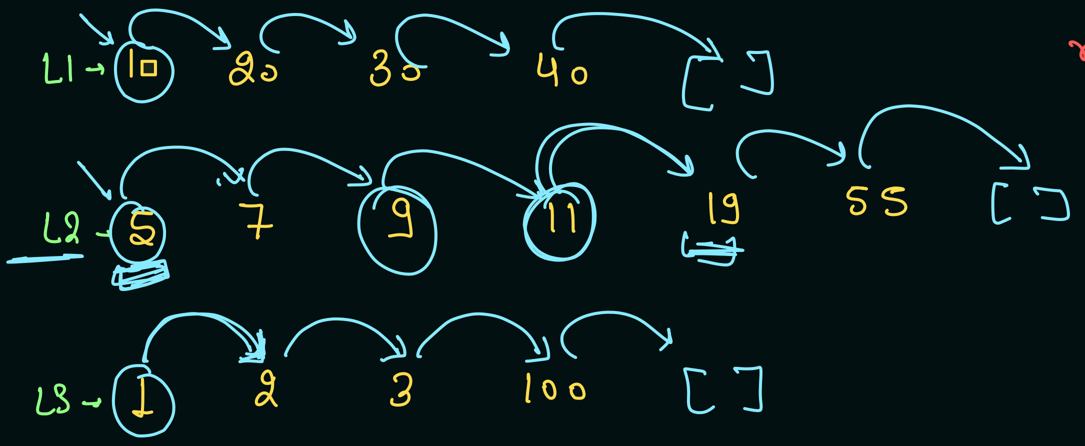
left → max

Right - min.

Size →

	left size		right size
	z	=	z
✓	y+1	>	y
✓	x	<	x+1

Merge K- Sorted List:



method 1 → add all element in pq. then one by one remove min. n Log(n)

using pointer → n×k

method 2 (using pq)

n log k

pq.

data

List Index

priority on data.

~~10-1~~, ~~5-2~~ ~~55-2~~

~~1-3~~ ~~3-3~~ ~~30-1~~

~~2-3~~ ~~100-3~~ ~~40-1~~

~~7-2~~ ~~9-2~~

~~11-2~~ ~~20-1~~

~~19-2~~

~~100-3~~

→ 1, 2, 3, 5, 7, 9, 10, 11, 19, 20, 30

40, 55, 100.

data

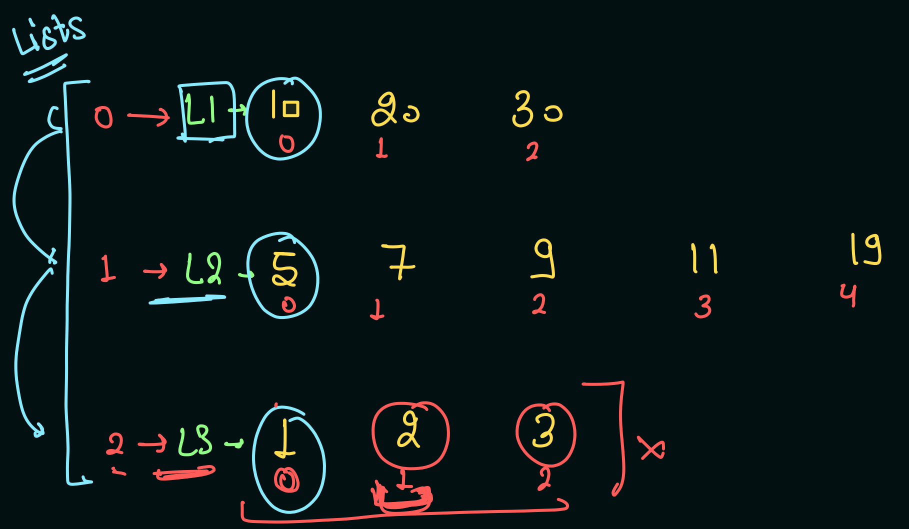
List Index

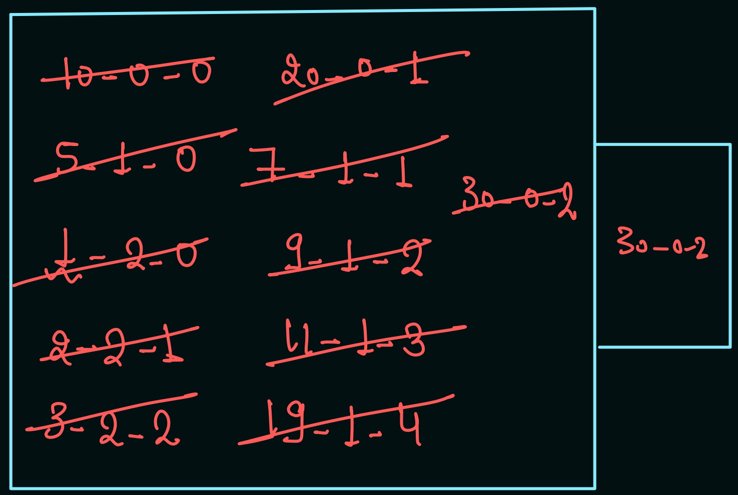
data Index

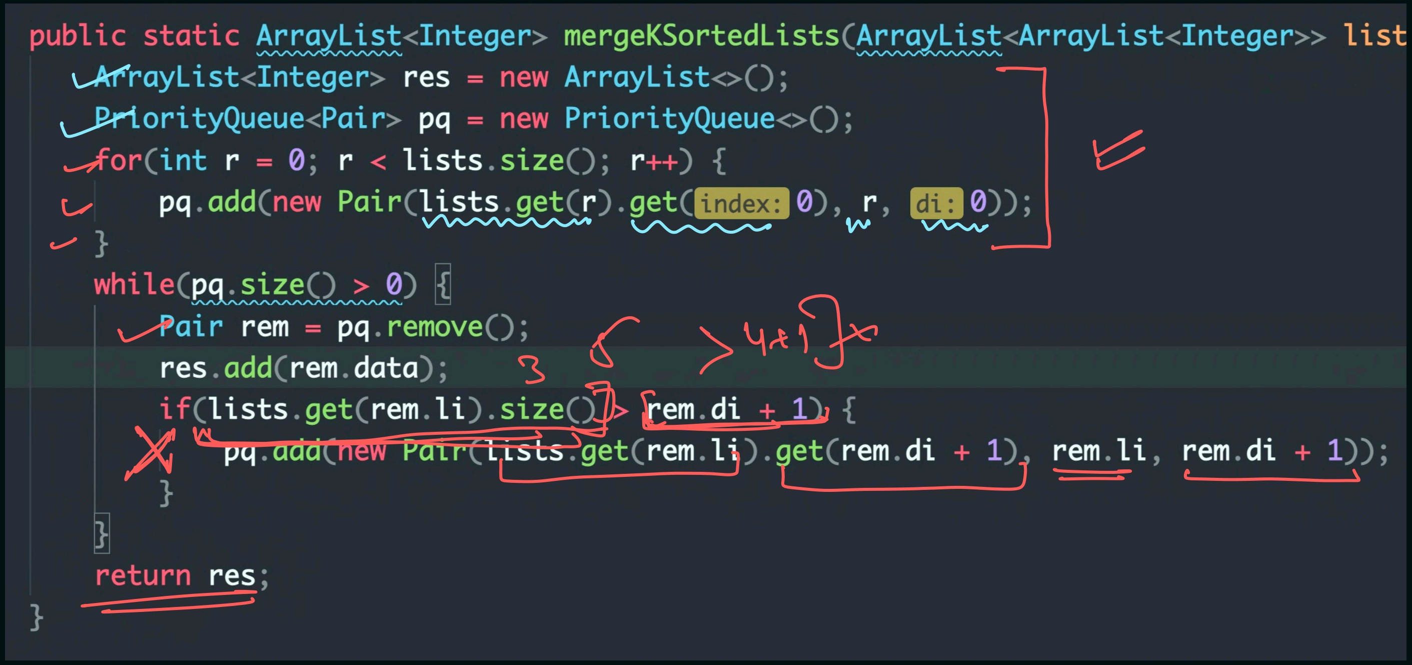
→ singleremove() in AL → O(n)

↑

first data







```
public static ArrayList<Integer> mergeKSortedLists(ArrayList<ArrayList<Integer>> list
    ArrayList<Integer> res = new ArrayList<>();
    PriorityQueue<Pair> pq = new PriorityQueue<>();
    for(int r = 0; r < lists.size(); r++) {
        pq.add(new Pair(lists.get(r).get(index: 0), r, di: 0));
    }
    while(pq.size() > 0) {
        Pair rem = pq.remove();
        res.add(rem.data);
        if(lists.get(rem.li).size() > rem.di + 1) {
            pq.add(new Pair(lists.get(rem.li).get(rem.di + 1), rem.li, rem.di + 1));
        }
    }
    return res;
}
```

result →

1 2 3 5 7 9 10

11 19 20 30