Merge K sorted arrays

K = 4 (all are in ↑ing order)

- n = 3 [1 , 5 , 9]
- n = 2 [45 , 90]
- n = 5 [2 , 6 , 78 , 90 , 101]
- n = 1 [0]

approach

min heap

1	2
5	6
9	78
45	90
90	101
	0

o/p

ans = [0 , 1 , 2 , 5 , 6 , 9 , 45 , 78 , 90 , 90 , 101]

Code

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);

    PriorityQueue<Integer> pq = new PriorityQueue<>();
    int k = scn.nextInt();
    for (int i = 0; i < k; i++) {
        int n = scn.nextInt();
        for (int j = 0; j < n; j++) {
            int val = scn.nextInt();
            pq.add(val);
        }
    }

    int[] ans = new int[pq.size()];
    int i = 0;
    while ( !pq.isEmpty() ) {
        int rem = pq.poll();
        ans[i] = rem;
        i++;
    }

    // printing
    for (int j : ans) {
        System.out.print(j + " ");
    }
}
```

$T.C = O(n \log n)$

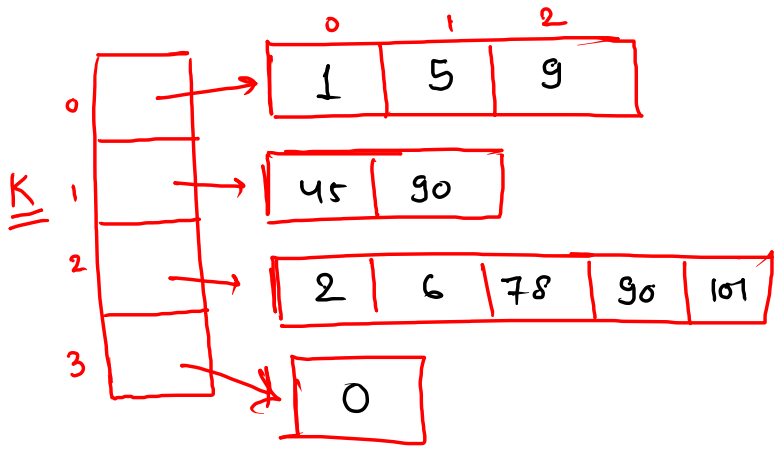
where n is total no. of elements from all the arrays

Better approch

K = 4 (all are in ↑ing order)

- n = 3 [1 , 5 , 9]
- n = 2 [45 , 90]
- n = 5 [2 , 6 , 78 , 90 , 101]
- n = 1 [0]

assume



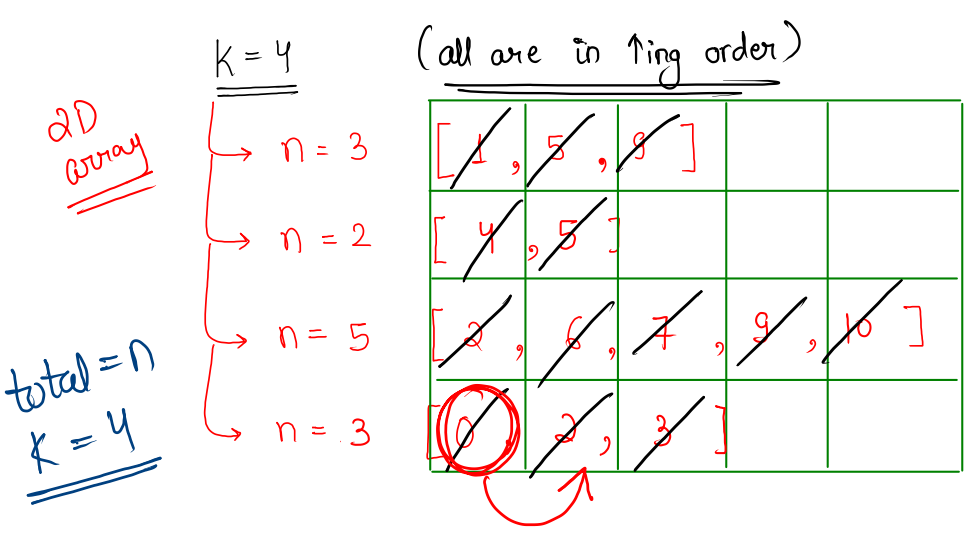
imagine

arraylist of arraylist

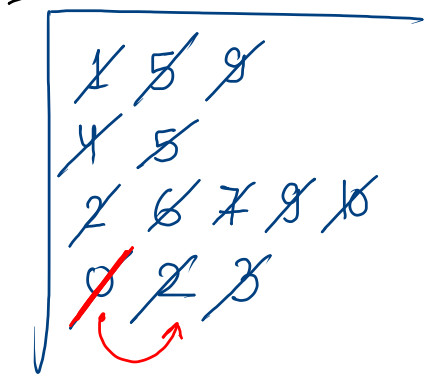
1	5	9		
45	90			
2	6	78	90	101
0				

how to make arraylist of arraylist

```
ArrayList< ArrayList<Integer>> arr = new ArrayList<>()
```



PQ



Ans = 0 1 2 2 3 4 5 5 6 7 9 9 10

T.C = K log(K)

you smallest element can only be in range of "first un-used elements of each row"

code creating input

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);

    ArrayList<ArrayList<Integer>> arr = new ArrayList<>();
    int k = scn.nextInt();
    for (int i = 0; i < k; i++) {
        int n = scn.nextInt();
        ArrayList<Integer> innerList = new ArrayList<>();
        for (int j = 0; j < n; j++) {
            int val = scn.nextInt();
            innerList.add(val);
        }
        arr.add( innerList );
    }

    // print
    for (int i = 0; i < arr.size(); i++) {
        for (int j = 0; j < arr.get(i).size() ; j++) {
            System.out.print( arr.get(i).get(j) + " " );
        }
        System.out.println();
    }
}
```

weakest rows (gmp)

rows from strong to weak

arr

enemy

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

$4 \rightarrow 1 \rightarrow 3 \rightarrow 0 \rightarrow 2$

$K = 3$ ans = 2 0 3

PQ

2
0
3
1

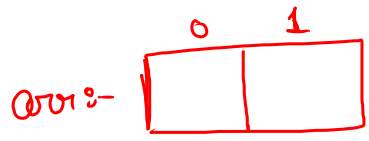
Imp note:-

- row with less no. of soldiers is weak
- if soldiers are equal, then row is less index is weak

every time we will store 2 values

↳ 1 :- no. of soldiers

↳ 2 :- index of row

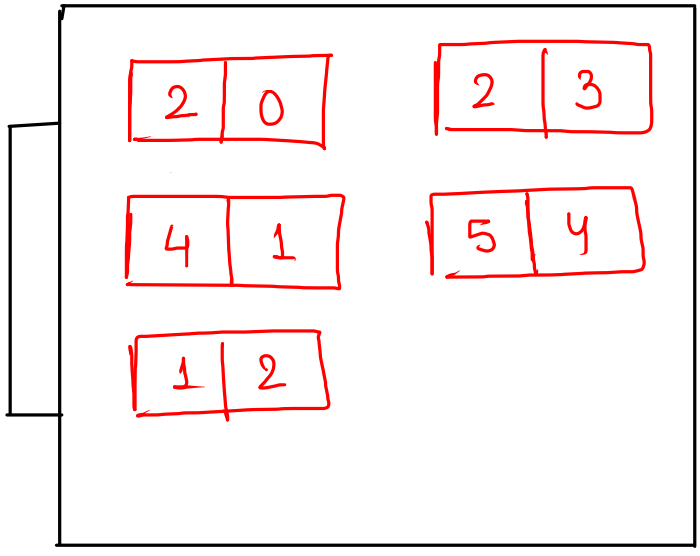


assuming :- 0^{th} idx :- no. of soldiers

1^{st} idx :- idx of row

PQ

want the weakest row at top



lambda function

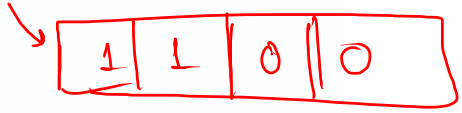
```
PriorityQueue<int[]> pq = new PriorityQueue<>((a, b)-> {
    if (a[0] != b[0])
        return a[0] - b[0];
    else
        return a[1] - b[1];
});
```

Code1

```java
public static void kthSmallest(int[][] arr, int m, int n, int k) {
    PriorityQueue<int[]> pq = new PriorityQueue<>((a, b) -> {
        if ( a[0] != b[0] ) {
            return a[0] - b[0];
        } else {
            return a[1] - b[1];
        }
    });

    for (int i = 0; i < m; i++) {
        int soldiers = find( arr[i] );
        int idxOfRow = i;

        int[] row = new int[2];
        row[0] = soldiers;
        row[1] = idxOfRow;

        pq.add(row);
    }

    // print top k values
    for (int i = 0; i < k; i++) {
        int[] rem = pq.poll();
        System.out.print( rem[1] + " " );
    }
}

public static int find(int[]  arr) {
    int count = 0;
    for (int i = 0; i < arr.length; i++) {
        if ( arr[i] == 1 ) {
            count++;
        }
    }
    return count;
}
```

$$T.C = ((m*n) + m\log(m))$$

$$\simeq O(m*n)$$



Code 2

```java
public static void kthSmallest(int[][] arr, int m, int n, int k) {
    PriorityQueue<int[]> pq = new PriorityQueue<>((a, b) -> {
        if ( a[0] != b[0] ) {
            return a[0] - b[0];
        } else {
            return a[1] - b[1];
        }
    });

    for (int i = 0; i < m; i++) {
        int soldiers = find( arr[i] );
        int idxOfRow = i;

        int[] row = new int[2];
        row[0] = soldiers;
        row[1] = idxOfRow;

        pq.add(row);
    }

    // print top k values
    for (int i = 0; i < k; i++) {
        int[] rem = pq.poll();
        System.out.print( rem[1] + " " );
    }
}

public static int find(int[]  arr) {
    int si = 0;
    int ei = arr.length - 1;
    while ( si <= ei ) {
        int mid = (si + ei) / 2;
        if ( arr[mid] == 1 ) {
            si = mid + 1;
        } else {
            ei = mid - 1;
        }
    }
    return si;
}
```

m

n

gmp

$T.C = O(m\log(n) + m\log(m))$

$T.C \cong O(m\log(n))$

$S.C = O(m)$