Merge K sorted arrays

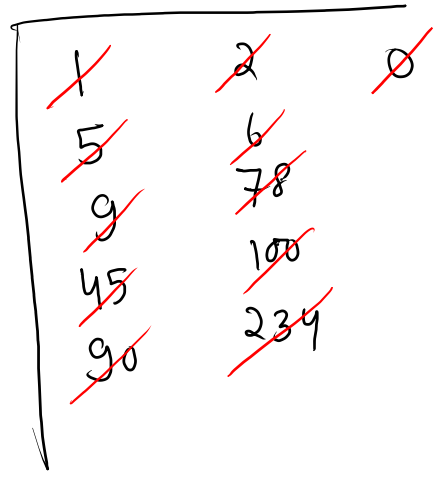
$k = 4$

$n = 3$ | 1 | 5 | 9 |

$n = 2$ | 45 | 90 |

$n = 5$ | 2 | 6 | 78 | 100 | 234 |

$n = 1$ | 0 |



Code T.C = O(N log N), let N = (k * n)

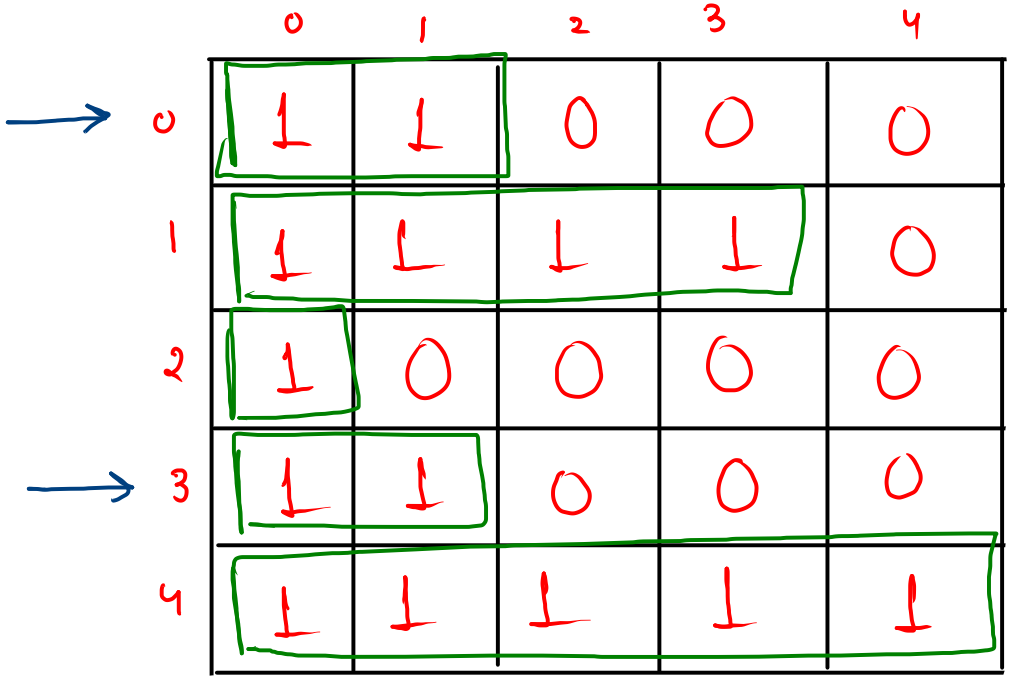
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    int k = scn.nextInt();
    for (int i = 0; i < k; i++) {
        int n = scn.nextInt();
        for (int j = 0; j < n; j++) {
            int val = scn.nextInt();
            pq.add( val );
        }
    }

    while ( !pq.isEmpty() ) {
        System.out.print( pq.poll() + " " );
    }
}
```

k arrays

n no. of elements in each array

weakest rows (Imp)

$m = 5$, $n = 5$, $k = 3$



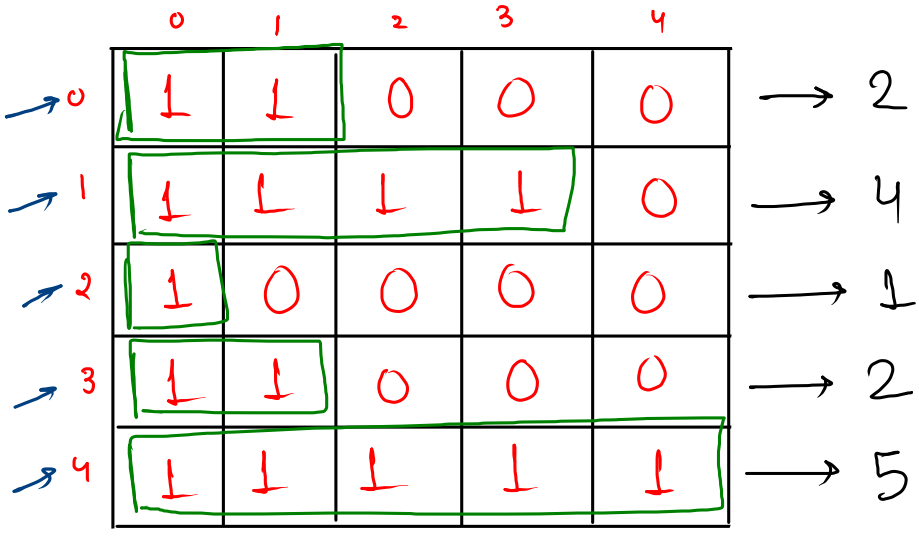
weakest row

2
0
3

} ans

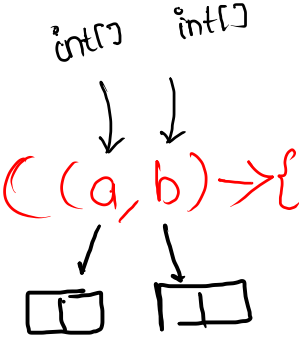
Imp:-

- less soldiers means weak row
- if soldiers are same then lower index row is weak row



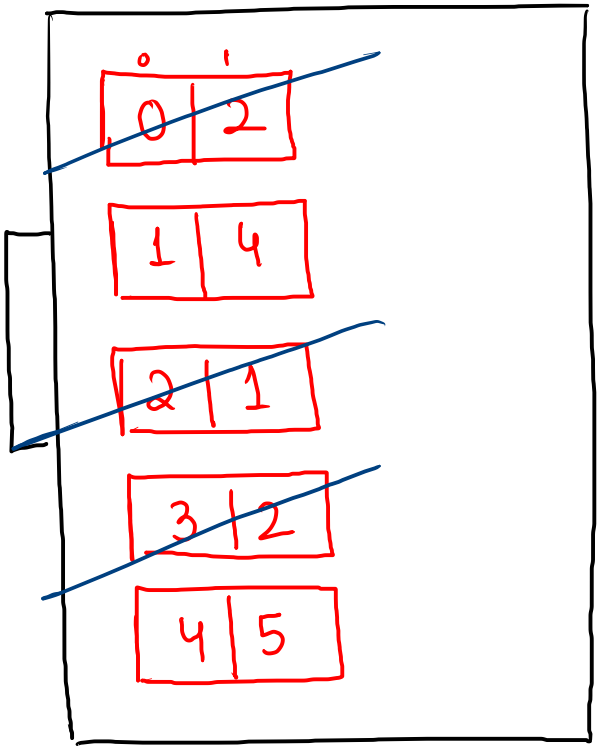
arr[i][j]

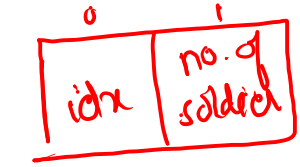
arr[i] → row




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

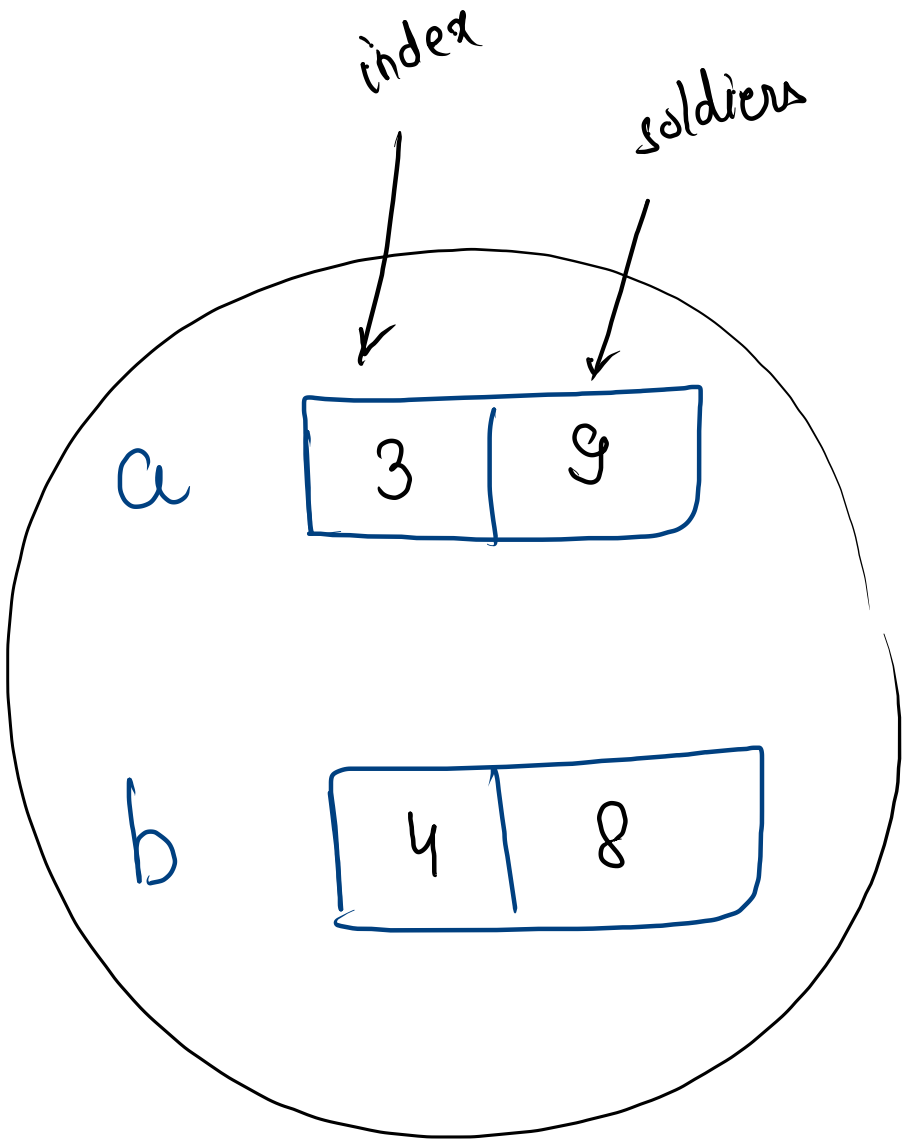
PQ



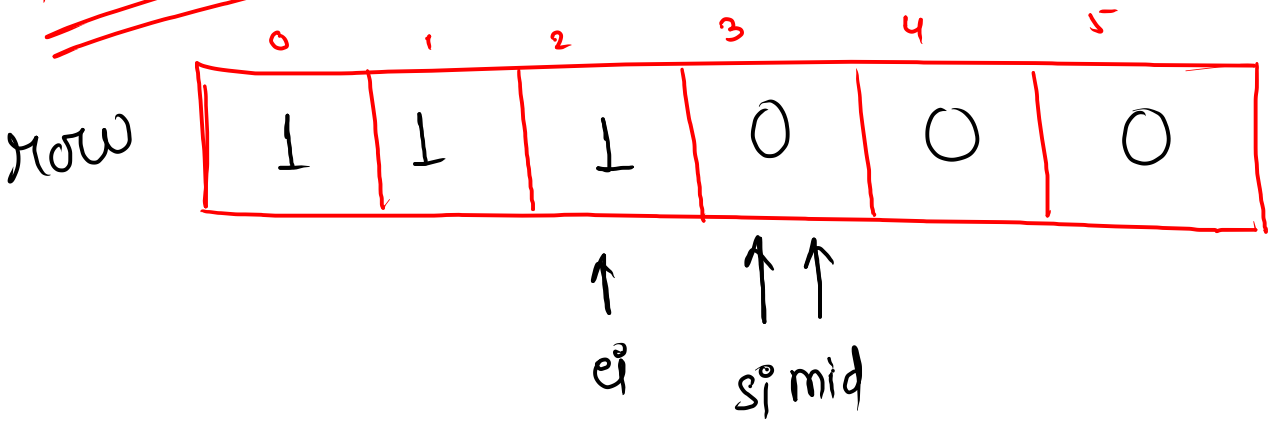



```
if( a[1] != b[1] ){
    return a[1]-b[1];
} else {
    return a[0]-b[0];
}
```

ans = 2 0 3


index
soldiers
a
3
9
b
4
8

find fn




```
while( si <= ei ){
    mid = (si + ei)/2;
    if( row[mid] == 1 ){
        si = mid+1;
    } else{
        ei = mid-1;
    }
}
return si;
```

O(logN)

```java
public static void weakestRow(int[][] arr, int m, int n, int k) {
    PriorityQueue<int[]> pq = new PriorityQueue<>((a, b) -> {
        if ( a[1] != b[1] ) {
            return a[1] - b[1];
        } else {
            return a[0] - b[0];
        }
    });
    for (int i = 0; i < m; i++) {
        int soldiers = find(arr[i]);
        int[] row = new int[2];
        row[0] = i;
        row[1] = soldiers;
        pq.add(row);
    }
    for (int i = 0; i < k; i++) {
        int[] temp = pq.poll();
        System.out.print( temp[0] + " " );
    }
}
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;
}
```

$O(m \log(m))$

$O(m)$

$O(\log n)$

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

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

initial startups :- minor development

- theory (Java)
- OOP's
- Dev. :- arch. (spring boot)

} 3 lpa ⋮ 10 lpa

high startup :- 50% dev and 50% DSA } 11 lpa ⋮ 25 lpa

arrays strings

MNC's :- 40 lpa ... 60 lpa 70% DSA