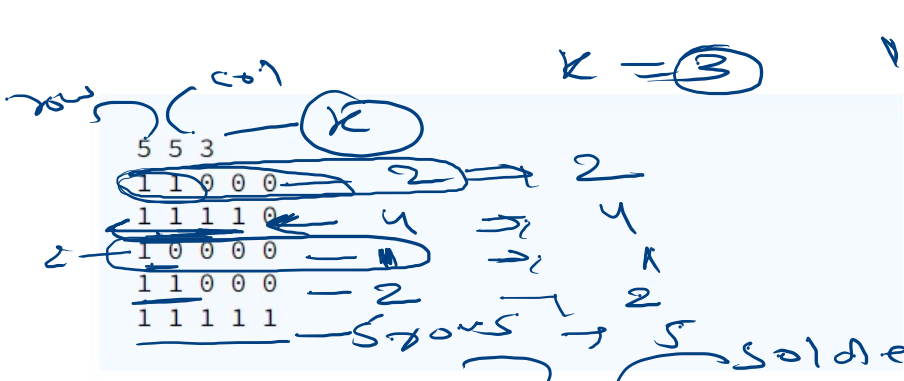
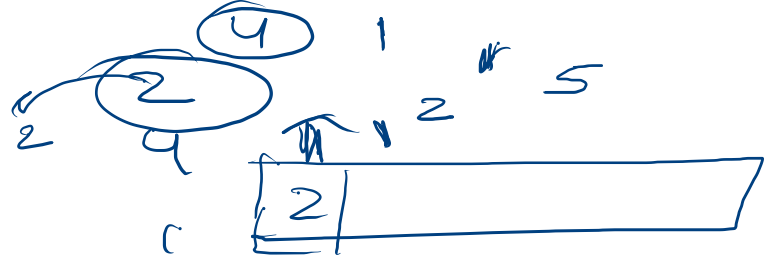




Row-0 → < 0, 2 >

Row-1 → < 1, 4 >

Row-2 → < 2, 1 >

Row-3 → < 3, 2 >

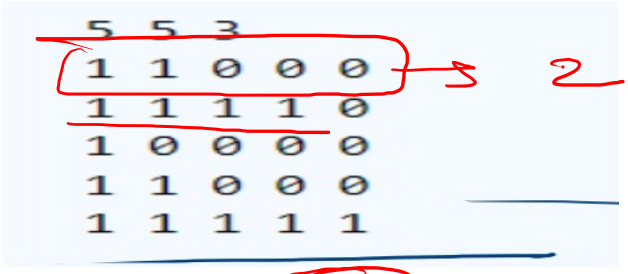
Row-4 → < 4, 5 >

{ 2, 0, 3 }

{ 2, 0, 3 }

while (x--)

```
5 5 3
1 1 0 0 0
1 1 1 1 0
1 0 0 0 0
1 1 0 0 0
1 1 1 1 1
```



Solder Time

(2, 0)

Binary search

0 0 2 mid

1 1 0 0 0

s mid e

(s + e)/2

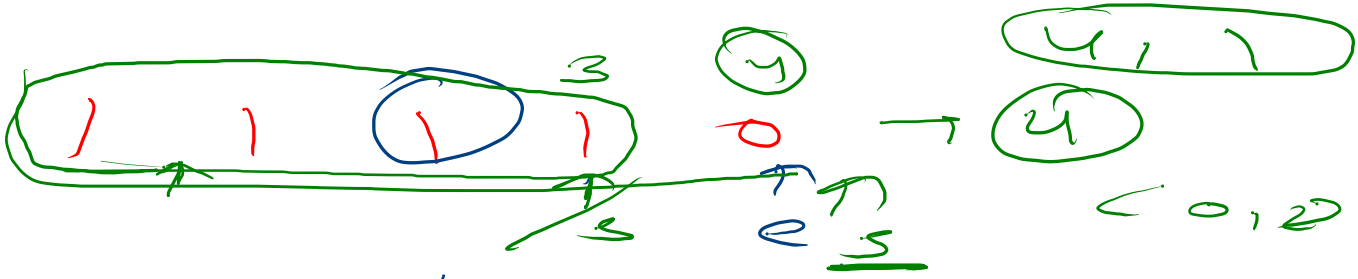
while (s <= e)

0 + 4 / 2 = 2

0 + 2 / 2

return s

2 / 2 = 1



$< 0, 2$

while (s < e) {

int s = 0, e = arr.length - 1

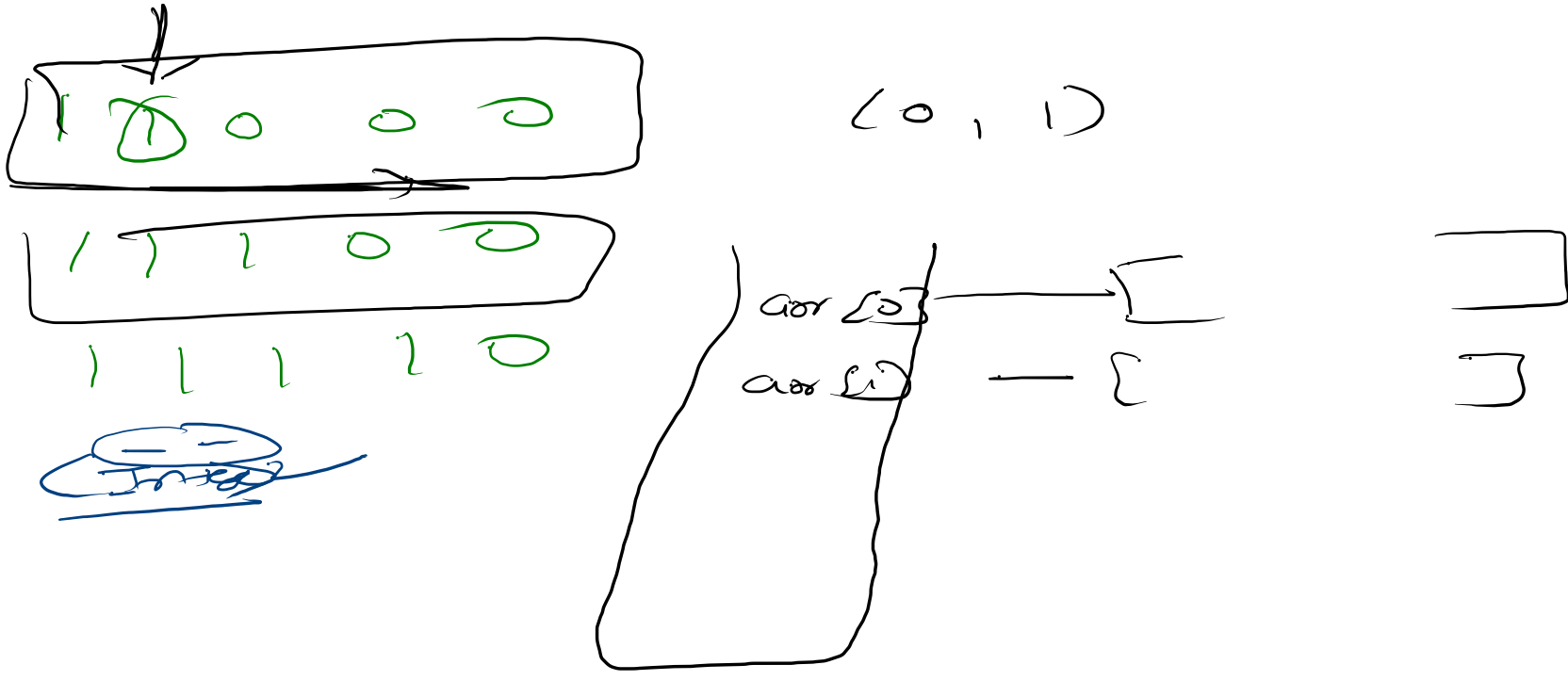
s = mid + 1;

$\frac{0+4}{2} = 2$

3

return start

$\frac{3+4}{2} = 3$



weakest rows

```java
public static int[] weakestRow(int[][] arr,int r,int c,int k){
    int[] res = new int[k];
    PriorityQueue<int[]> pq = new PriorityQueue<>((a,b)->a[0]!=b[0]?b[0]-a[0]:b[1]-a[1]);
    for(int i=0;i<arr.length;i++){
        pq.add(new int[]{binarySeach(arr[i]),i});
        if(pq.size()>k){
            pq.poll();
        }
    }
    while(pq.size()>0){
        res[--k] = pq.poll()[1];
    }
    return res;
}
public static int binarySeach(int[] arr){
    int s=0;
    int e = arr.length-1;
    while(s<e){
        int mid = (s+e)/2;
        if(arr[mid]==1){
            s = mid+1;
        }else{
            e = mid;
        }
    }
    return s;
}
```

```
5 5 3
1 1 0 0 0
1 1 1 1 0
1 0 0 0 0
1 1 0 0 0
1 1 1 1 1
```

Brute force

optimize

calculate the time and space complexity

weakest rows

```
 public static void main(String[] args) {
     /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solutior
     Scanner sc = new Scanner(System.in);
     int r = sc.nextInt();
     int c = sc.nextInt();
     int k = sc.nextInt();
     int[][] mat = new int[r][c];
     for(int i=0;i<r;i++){
         for(int j=0;j<c;j++){
             mat[i][j]=sc.nextInt();
         }
     }
     weakestRow(mat,r,c,k);
 }
public static void weakestRow(int[][] arr,int r,int c,int k){
     int[] res = new int[r];
         for(int i=0;i<r;i++){
             int count=0;
             for(int j=0;j<c;j++){
                if(arr[i][j]==1){
                    count++;
                }
             }
             res[i] = count;

         }
     // System.out.println(Arrays.toString(res));
    while(k-->0){
        int min = Integer.MAX_VALUE;
        int inx = -1;
        for(int i=0;i<res.length;i++){
            if(res[i]<min){
                min = res[i];
                inx = i;
            }
        }
        System.out.print(inx+" ");
        res[inx] = Integer.MAX_VALUE;
    }
     }
```


max
min
4 , 5 , 6 , 2 ,,
4 , 5 , 6 , 2 , 1
[,

```
5 5 3
1 1 0 0 0
1 1 1 1 0
1 0 0 0 0
1 1 0 0 0
1 1 1 1 1
```