Revision.

PQ.

Integer — min

max

cust

even + min

even

FIFO

A

1337. The K Weakest Rows in a Matrix

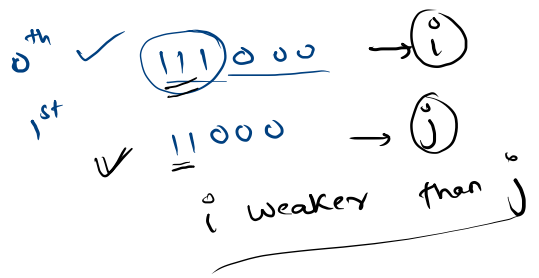
Easy 3346 202 Add to List Share

You are given an `m x n` binary matrix `mat` of `1`'s (representing soldiers) and `0`'s (representing civilians). The soldiers are positioned **in front** of the civilians. That is, all the `1`'s will appear to the **left** of all the `0`'s in each row.

A row `i` is **weaker** than a row `j` if one of the following is true:

- The number of soldiers in row `i` is less than the number of soldiers in row `j`.
- Both rows have the same number of soldiers and `i < j`.

Return *the indices of the* `k` ***weakest*** *rows in the matrix ordered from weakest to strongest.*



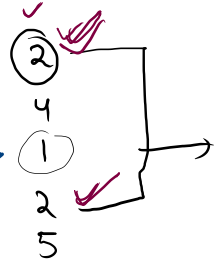
Example 1:

```
Input: mat =
[[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]],
k = 3
Output: [2,0,3]
Explanation:
The number of soldiers in each row is:
- Row 0: 2
- Row 1: 4
- Row 2: 1
- Row 3: 2
- Row 4: 5
The rows ordered from weakest to strongest are [2,0,3,1,4].
```

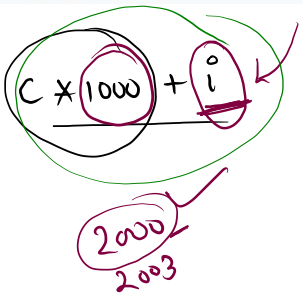
Example 2:

```
Input: mat =
[[1,0,0,0],
 [1,1,1,1],
 [1,0,0,0],
 [1,0,0,0]],
k = 2
Output: [0,2]
Explanation:
The number of soldiers in each row is:
- Row 0: 1
- Row 1: 4
- Row 2: 1
- Row 3: 1
The rows ordered from weakest to strongest are [0,2,3,1].
```

Input: mat =
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]],
k = 3



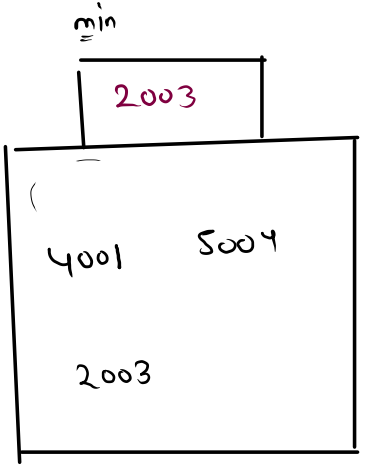
2 info



k times.

[2 0 3]

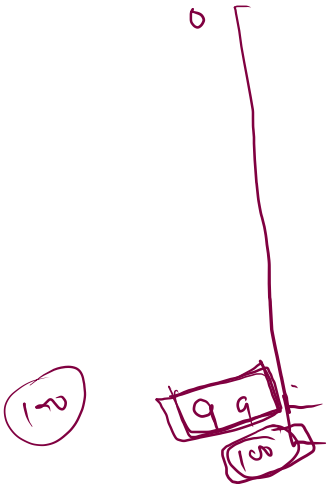
2003 % 1000 = 3



2 % 5 = 2

99 % 100 = 99

100 % 100 = 0



99 % 1000

99 % 100

Example 1:

```
Input: mat =
[[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]],
k = 3
Output: [2,0,3]
Explanation:
The number of soldiers in e
- Row 0: 2
- Row 1: 4
```

C=5

500 + 4


```java
class Solution {
    public int[] kWeakestRows(int[][] mat, int k) {
        PriorityQueue<Integer> pq = new PriorityQueue<>();  //min

        for(int i = 0; i < mat.length; i++){
            int count = 0;
            for(int j = 0; j < mat[0].length; j++){
                count += mat[i][j];
            }
            pq.add(count*100 + i);
        }

        int [] ans = new int[k];
        for(int i = 0; i < k; i++){
            ans[i] = pq.remove() % 100;
        }
        // int idx = 0;
        // while(k-- > 0){
        //     ans[idx++] = pq.remove() % 100;
        // }
        return ans;
    }
}
```

k=3

102
200
203

% 100 = 2 0 3

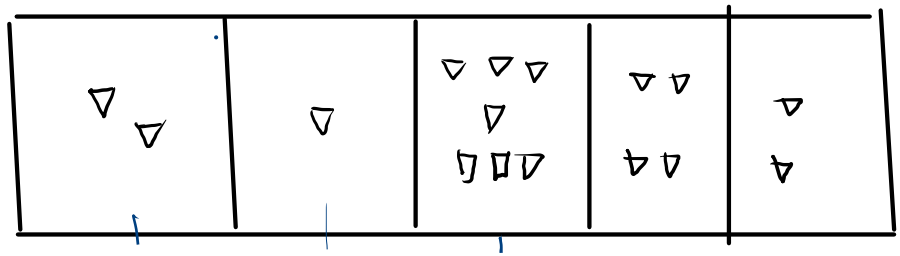
maximum diamonds

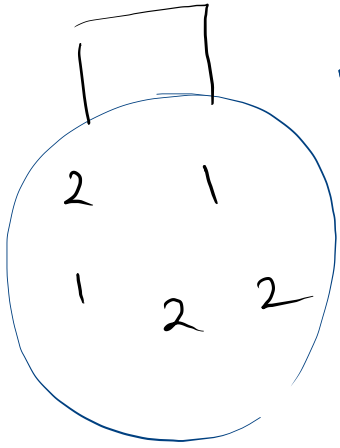
K → min
n → size

5 3
2 1 7 4 2

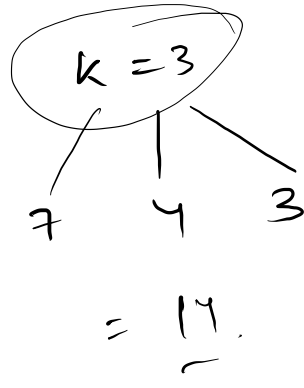
Sample Output 0

14





max

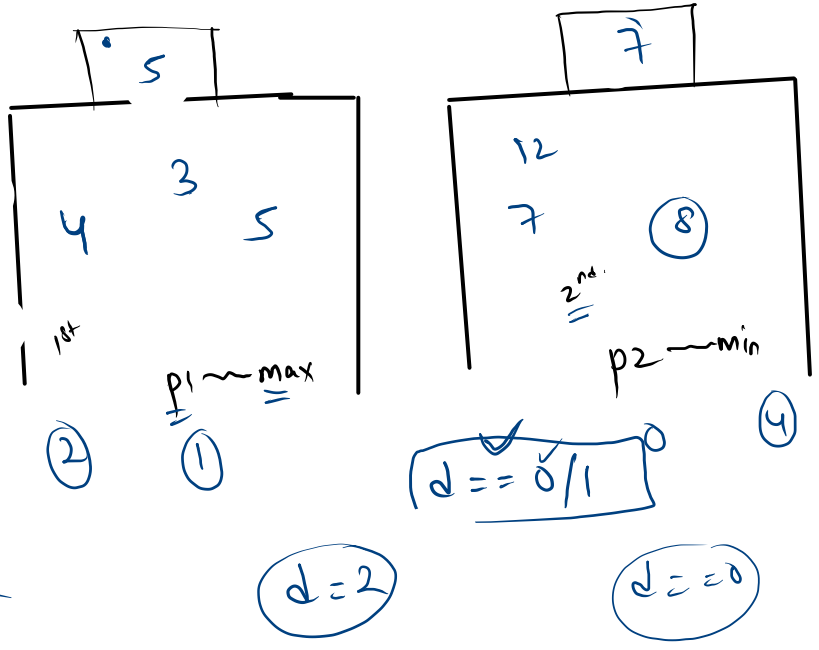


```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
       Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int k = scn.nextInt();

        PriorityQueue<Integer> pq = new PriorityQueue<>((a,b)->{
            return b-a;
        });
        for(int i = 0; i < n; i++){
            pq.add(scn.nextInt());
        }
        int ans = 0;
        for(int i = 0; i < k; i++){
            int rem = pq.remove();
            ans += rem;
            pq.add(rem/2);
        }
        System.out.println(ans);
    }
}
```

Find the running Median.

add ✓
balance
find

STDIN	Function
6	a[] size n = 6
12	a = [12, 4, 5, 3, 8, 7]
4	
5	
3	
8	
7	



```
STDIN   Function
-----   --------
6       a[] size n = 6
12      a = [12, 4, 5, 3, 8, 7]
4
5
3
8
7
```

12.0
8.0
5.0
4.5
5.0
6.0

add
balance
ans

p1.peek > ele

12 > 4

4 > 5

4 > 3

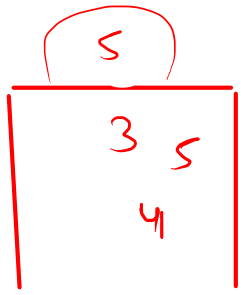
4 > 8

d = 1

1st

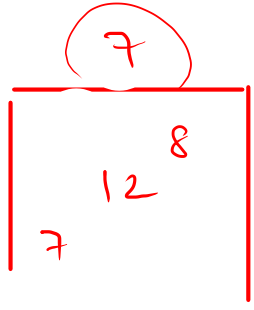
p1 (half)

max



p2 (2nd half)

min



① Find The Index of Rotation

```
5
5 1 2 3 4
```

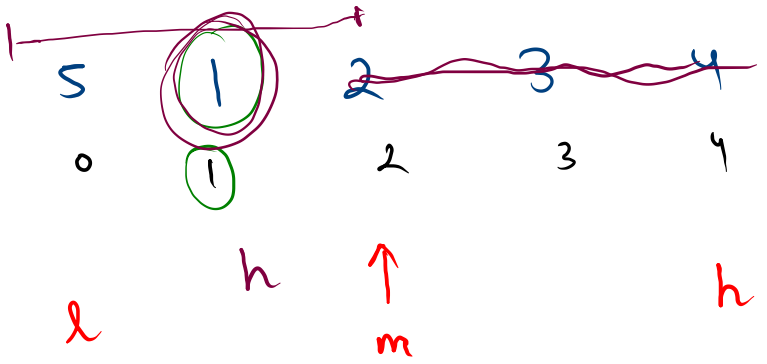
② find min in rotated sorted array

③ find max in rotated sorted array

④ find index of max in rotated sorted array

⑤ find index of min in rotated sorted array

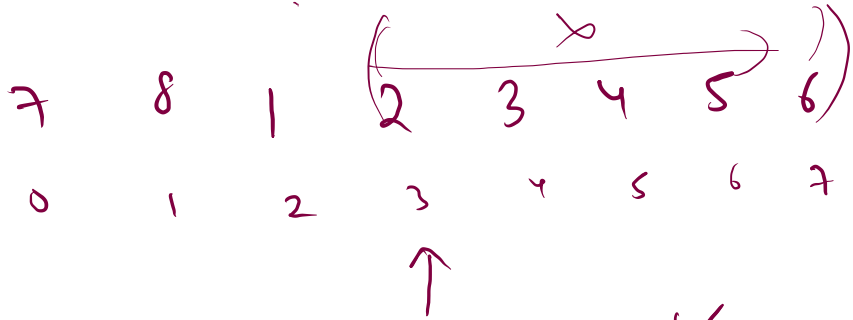
⑥ find no. of rotation



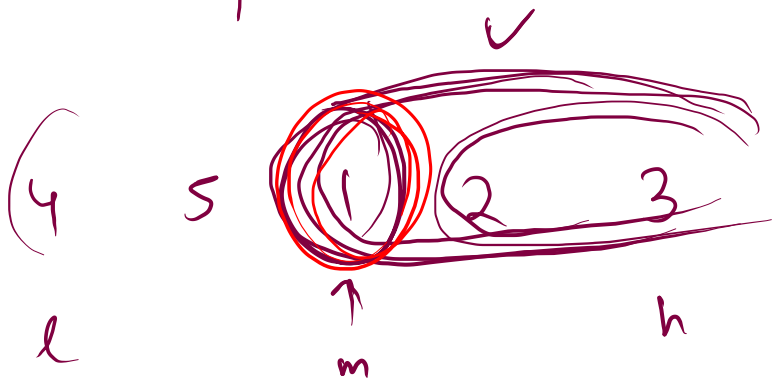
mid = $\lfloor 2 \rfloor_2$.

$A[l] \leq A[m]$ ~ false

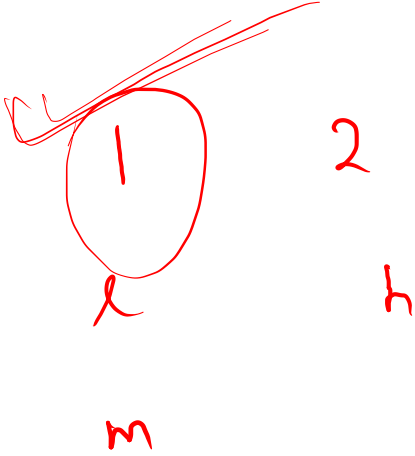
Unsorted region.
↳ yang.



$A[l] \leq A[m]$



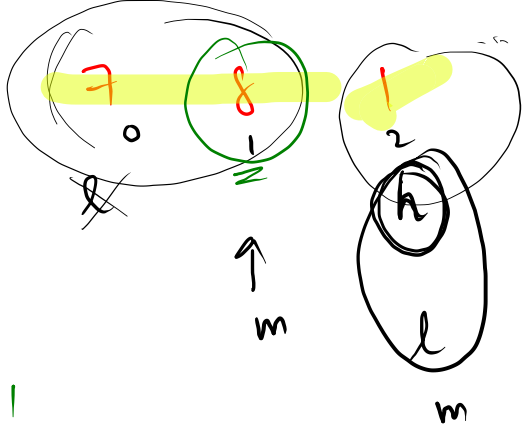
unsorted.
↳ ans may or may not be.



$A[l] \leq A[m]$

ans = ~~2~~ 1

index = m(~~3~~) 2



$A[l] \leq A[m]$ ✓

2 - 1

2, 7

$A[l] \leq A[m]$

✓

ans = 1

index = 2

s

l

h

m

2-1

1

A[l] ≤ A[m]

4 ≤ 4 ✓

∞, 1

tar=2

find last occu