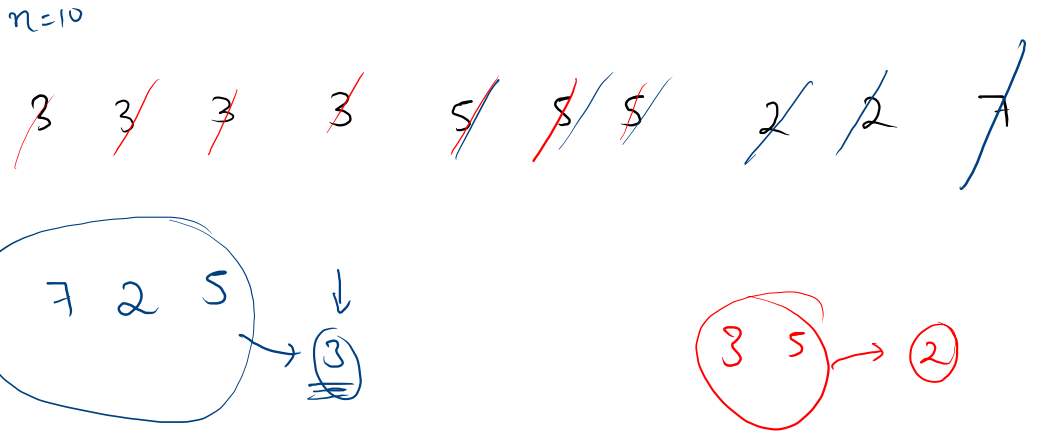
Reduce Array Size to the half 1

You are given an integer array arr. You can choose a set of integers and remove all the occurrences of these integers in the array.

Return the minimum size of the set so that at least half of the integers of the array are removed.

Sample Input 0

```
10
3
3
3
3
5
5
5
2
2
7
```



Sample Output 0

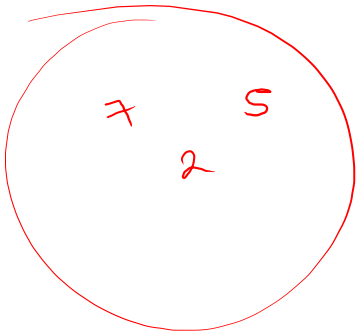
```
2
```

removed = 1 + 2 + 3

r ≥ 5

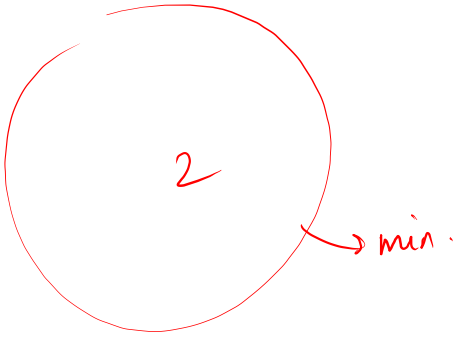
total = 10

3 3 3 3 ~~2~~ ~~2~~ ~~5~~ ~~5~~ ~~5~~ ~~7~~



ans = 3.

2 2 2 2 2 2 2 2 2 2 1 7 9 3 3 4



$7 < n/2$ $r \geq n/2$

3 3 3 3 2 2 7 5 5 5

$r < n/2$

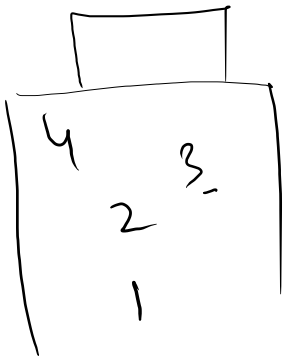
k	v
3	4
2	2
5	3
7	1

removed = ∅ 4 7

1. freq map.
2. priority Queue → max

count = ∅ 2 }

(3, 5)



2 2 3 3 3 3 7 5 5 5

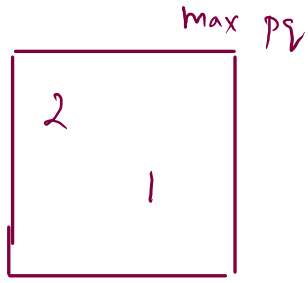
n=10

2 - 2
3 - 4
5 - 3
7 - 1

removed = 0 + 4 + 3

count = 0 + 1 + 1

Ans = 2




```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();
        HashMap<Integer, Integer> hm = new HashMap<>();
        for(int i = 0; i < n; i++){
            int x = scn.nextInt();
            hm.put(x, hm.getOrDefault(x, 0)+1);
        }

        PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());    //max
        pq.addAll(hm.values());
        int removed = 0;
        int count = 0;
        while(removed < n/2){
            removed += pq.remove();
            count++;
        }
        System.out.println(count);
    }
}
```

subtract numbers 1

You are given a non-negative integer array **nums**. In one operation, you must:

- Choose a positive integer x such that x is less than or equal to the **smallest non-zero element in nums**.
- Subtract x from every **positive** element in **nums**.

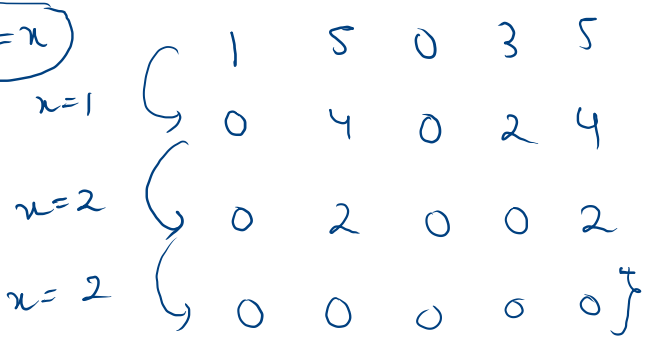
Return the **minimum** number of operations to make every element in **nums** equal to 0.

Sample Input 0

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

Sample Output 0

```
3
```



eg. 0 0 0 0

ans = 0

eg. 1 2 3 4 5

ans = 5

eg. 0 4 2

ans = 2 [0 2 0; 0 0 0]

eg. 1 2 2 2 3 4 5 5

ans = 5

eg. 4 2

ans = 2

eg. 1 1 0 0 5 5 4

ans = 3

eg. 0 4 2

x=2 0 2 0

x=2 0 0 0

0 4 2 2

x=2 0 2 0 0

x=2 0 0 0 0

0 4 4 2 2

x=2 0 2 2 0 0

x=2 0 0 0 0 0

unique non zero

eg.

	1 ✓	2 ✓	3	4	5
$x=1$	0	1	2	3	4 ←
$x=1$	0	0	1	2	3
$x=1$	0	0	0	1	2
$x=1$	0	0	0	0	1
$x=1$	0	0	0	0	0

unique non zero = 5 = 5 moves.

	1	0	0	2	2	3	4	5
$x=1$	0	0	0	1	1	2	3	4
$x=1$	0	0	0	0	0	1	2	3
$x=1$	0	0	0	0	0	0	1	2
$x=1$	0	0	0	0	0	0	0	1
$x=1$	0	0	0	0	0	0	0	0

eg. 4 4 2 2 0 0 1 1 1 3 5

1 5 0 3 5

```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();
        HashSet<Integer> hs = new HashSet<>();
        while(n-- > 0){
            int x = scn.nextInt();
            if(x != 0){
                hs.add(x);
            }
        }
        System.out.println(hs.size());
    }
}
```

	1	5	0	3	5
$x=1$	0	4	0	2	4
$x=2$	0	2	0	0	2
$x=2$	0	0	0	0	0

weakest rows

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.*

Sample Input 0

```
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
```

Sample Output 0

```
2 0 3
```

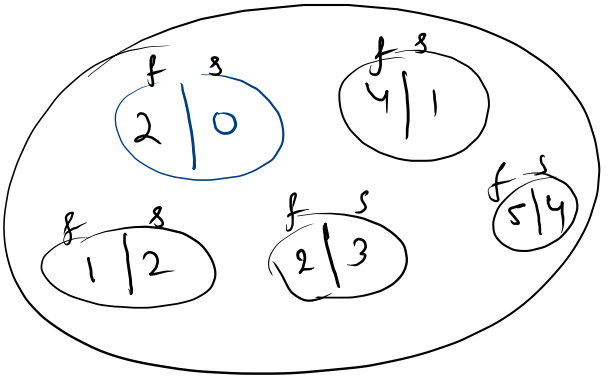
0 2

1 4

2 1

3 2

4 5



row	soldier
0	
N	S
0	2
1	4
2	1
3	2
4	5

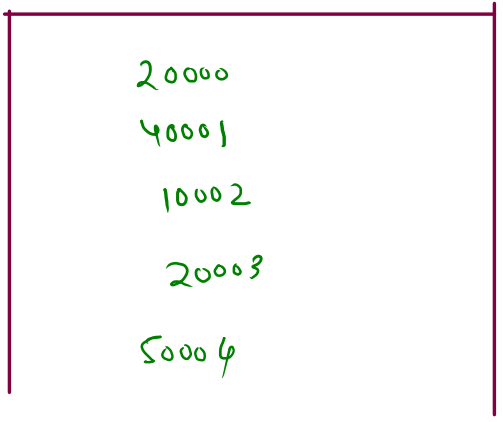
max rows = 10000

0 9999

S * 10000 + i

2, 0

0, 2



min

k = 3

x = 10002 / 20000 / 20003

x % 10000 ⇓ 2

⇓ 0

⇓ 3

203

i → 8

2

8 * 10 + i

m = 10000

0 9999

0

1

2

20

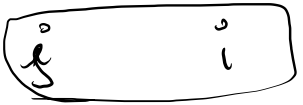
30 % 10 = 0

2 = 30

10

1 = 30

20



$5 * 3 + 2 = 17$

$$
\begin{array}{r}
3 \\
5\overline{)17} \\
15 \\
\hline
2
\end{array}
$$

```java
public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int m = scn.nextInt();
        int n = scn.nextInt();
        int k = scn.nextInt();
        int [][] A = new int[m][n];
        for(int i = 0; i < m; i++){
            for(int j = 0; j < n; j++){
                A[i][j] = scn.nextInt();
            }
        }

        PriorityQueue<Integer>pq = new PriorityQueue<>();

        for(int i = 0; i < m; i++){
            int s = 0;
            for(int j = 0; j < n; j++){
                s += A[i][j];
            }
            pq.add(s * 10000 + i);
        }
        while(k-- > 0){
            int rem = pq.remove();
            System.out.print(rem % 10000 + " ");
        }
    }
}
```