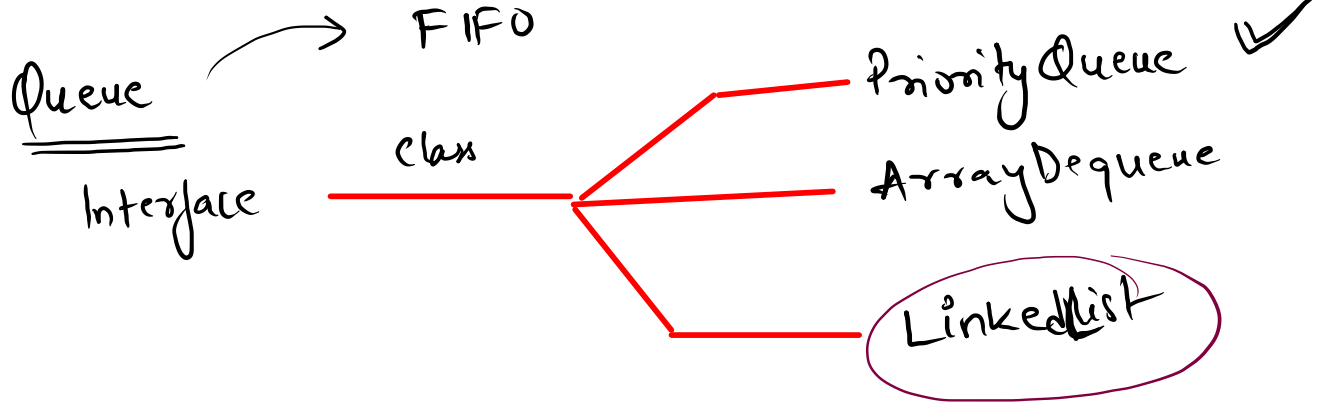


Revision.



add()

remove()

peek()

size()

~ front

First Negative Integer 2

Sample Input 0

5 2
-8 2 3 -6 10

Sample Output 0

-8 0 -6 -6

$k \rightsquigarrow$ window size
 $k=2$

-8 2 3 -6 10

-8 0 -6 -6

?

$k=2$

-8 2 3

-6 10

$k=2$

$i < k$ first k elements
 $A[i] < 0$
add(i)

3

-8 -6

peek < $(-k+1)$

2 - 2 + 1
0 + 1

0 < 1

4 - 2 + 1
3 < 3

queue.size() == 0
 $\hookrightarrow 0$

else $A[peek]$

remove

add -ve in

```

17 Queue<Integer> qu = new LinkedList<>();
18 int i = 0;
19
20 while(i < k){
21     if(A[i] < 0){
22         qu.add(i);
23     }
24     i++;
25 }
26
27 while(i < n){
28     if(qu.size() == 0){
29         System.out.print(0 + " ");
30     }
31     else{
32         System.out.print(A[qu.peek()] + " ");
33     }
34
35     while(qu.size() != 0 && qu.peek() < i-k+1){
36         qu.remove();
37     }
38
39     if(A[i] < 0){
40         qu.add(i);
41     }
42     i++;
43
44 }
45 if(qu.size() == 0){
46     System.out.print(0 + " ");
47 }
48 else{
49     System.out.print(A[qu.peek()] + " ");
50 }
51
52

```

$$k=3$$

$$k=3$$



$$i = k + 1$$

$$3 - 3 + 1$$

$$0 + 1$$

$$0 < 1$$

$$-8$$

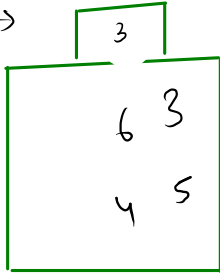
$$0$$

Priority Queue.

↳ DS.

min

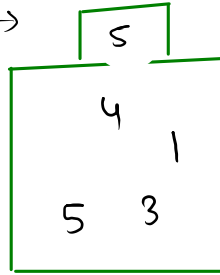
peek →



customize.

max

peek →



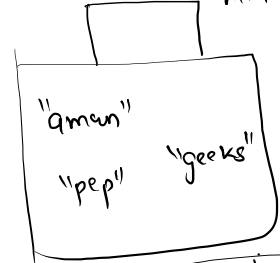
1 →
n → $n \log n$

$\log n$ → insert
 $O(1)$ → remove.

down heapify
up heapify

pa } binary.
Heap.
Tree based
↓
Tree.

String priority.
min → len
max → len
min → lexo



Sorting
Comparator
lambda.

PQ

```
1 import java.util.PriorityQueue;
2 import java.util.*;
3 public class Main
4 {
5     public static void main(String[] args) {
6         PriorityQueue<Integer> pq = new PriorityQueue<>();
7
8         //add
9         pq.add(50);
10        pq.add(70);
11        pq.add(10);
12        pq.add(30);
13
14        pq.remove();
15        System.out.println(pq.peek());
16        System.out.println(pq.size());
17
18
19
20
21    }
```

```
1 import java.util.PriorityQueue;
2 import java.util.*;
3 public class Main
4 {
5     public static void main(String[] args) {
6         // PriorityQueue<Integer> pq = new PriorityQueue<>(); //min
7
8         // PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
9         PriorityQueue<Integer> pq = new PriorityQueue<>((a,b)->{
10             return b-a;
11         }); //max
12
13         //add
14         pq.add(50);
15         pq.add(70);
16         pq.add(10);
17         pq.add(30);
18
19         while(pq.size() != 0){
20             System.out.println(pq.remove());
21         }
22
23     }
```

1046. Last Stone Weight

Easy 5444 93 Add to List Share

You are given an array of integers `stones` where `stones[i]` is the weight of the i^{th} stone.

We are playing a game with the stones. On each turn, we choose the heaviest two stones and smash them together. Suppose the heaviest two stones have weights x and y with $x \leq y$. The result of this smash is:

- If $x == y$, both stones are destroyed, and
- If $x \neq y$, the stone of weight x is destroyed, and the stone of weight y has new weight $y - x$.

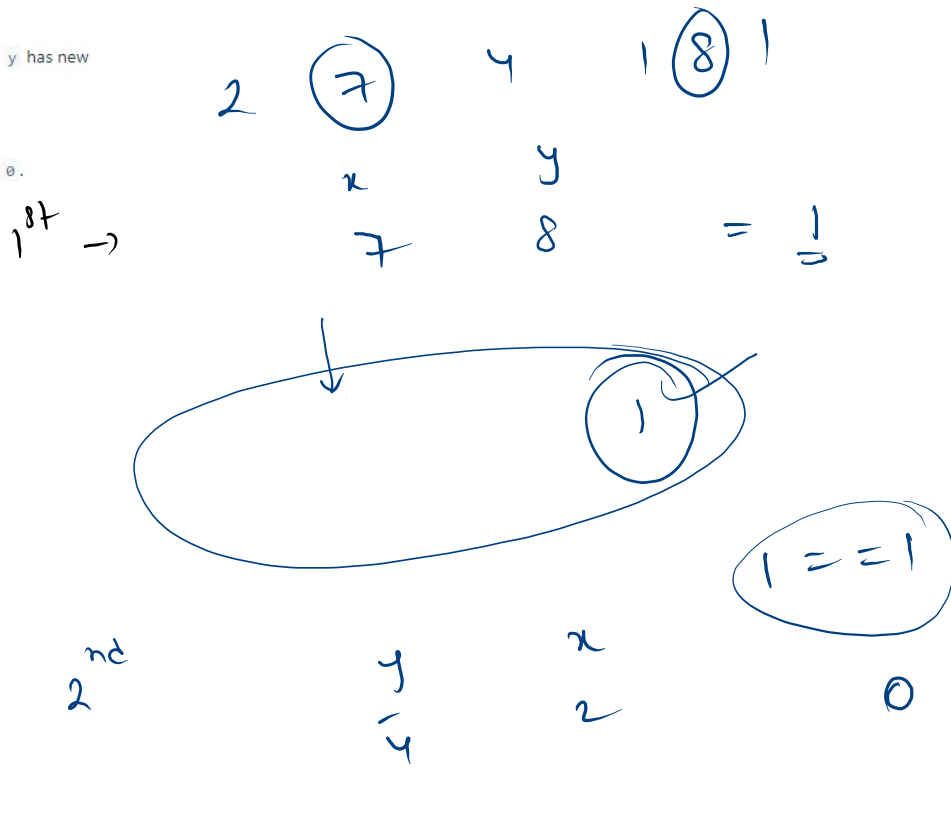
At the end of the game, there is **at most one** stone left.

Return the weight of the last remaining stone. If there are no stones left, return `0`.

Input: stones = [2,7,4,1,8,1]

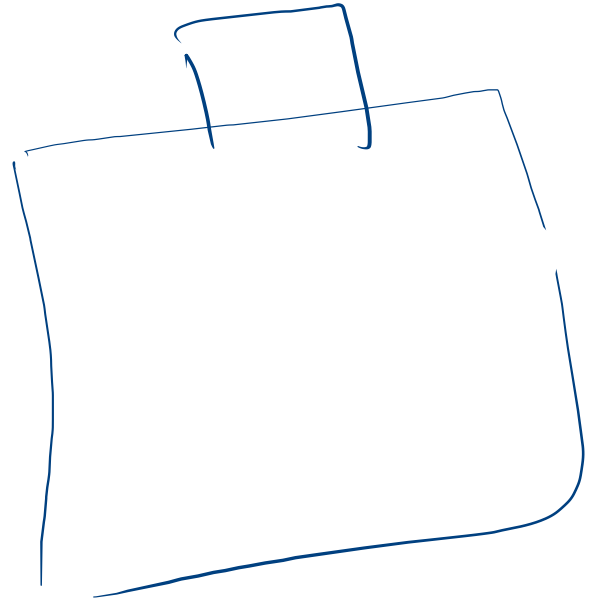
Output: 1

Explanation:



	2	2
--	---	---

x y
 2 $2 = 0$
 \checkmark



①

2 7 4 1 8 1

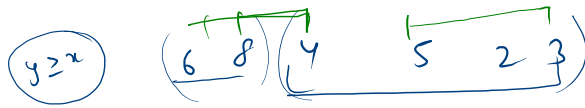
②

2/2

```
1  class Solution {
2      public int lastStoneWeight(int[] stones) {
3          PriorityQueue<Integer> pq = new PriorityQueue<>((a,b)->{
4              return b-a;          // max pq
5          });
6
7          for(int ele : stones){
8              pq.add(ele);
9          }
10
11         while(pq.size() > 1){
12             int y = pq.remove();    //8
13             int x = pq.remove();    // 7
14
15             if(y-x > 0){
16                 pq.add(y-x);
17             }
18         }
19
20         return pq.size() == 0 ? 0 : pq.peek();
21
22     }
23 }
```

minimum digits

Sample Input 0



(2) any

4 5 2 3
6 8



$$\begin{array}{r} 4523 \\ + 68 \\ \hline v1 \end{array}$$

(2)

$$\begin{array}{r} y = 684 \\ + x = 523 \\ \hline v2 \end{array}$$

res ↓
↓
= sum ↓

6 8 4 5 2 3

$$\begin{array}{r} 68452 \\ + 3 \\ \hline 68455 \checkmark \end{array}$$

>

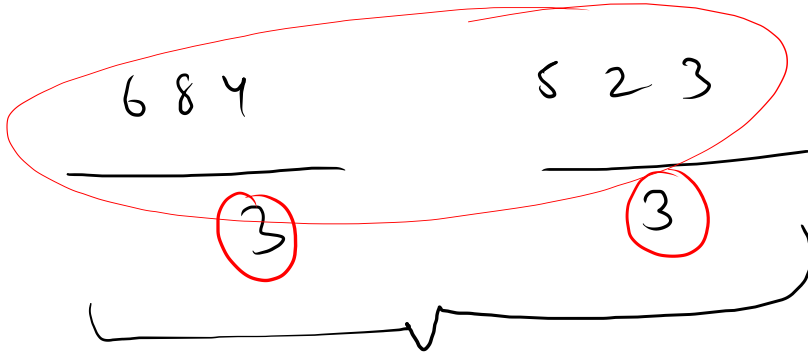
$$\begin{array}{r} 6845 \\ + 23 \\ \hline 6868 \end{array}$$

(?)

$$\begin{array}{r} 68 \\ 2345 \end{array}$$

$$\begin{array}{r} 684 \\ + 235 \\ \hline 719 \end{array}$$

balance



3×3

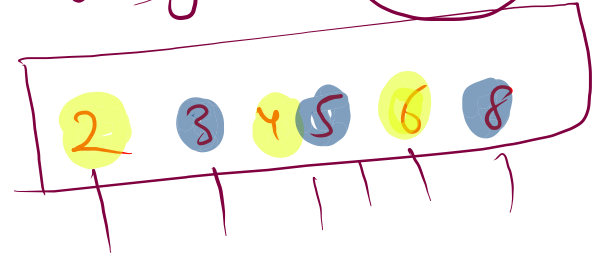
682
453

1435 ✓

1
684
523

1207

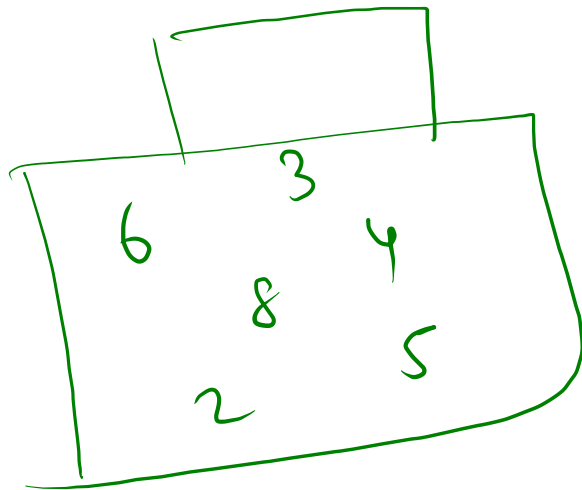
sorting ✓ $n \log n$



246
358

604 ↓

6 8 4 5 2 3



Integer
(min)

form 2 long.

2 4 6

3 5 8

29.

```
PriorityQueue<Integer> pq = new PriorityQueue<>(); //min

for(int ele : A){
    pq.add(ele);
}

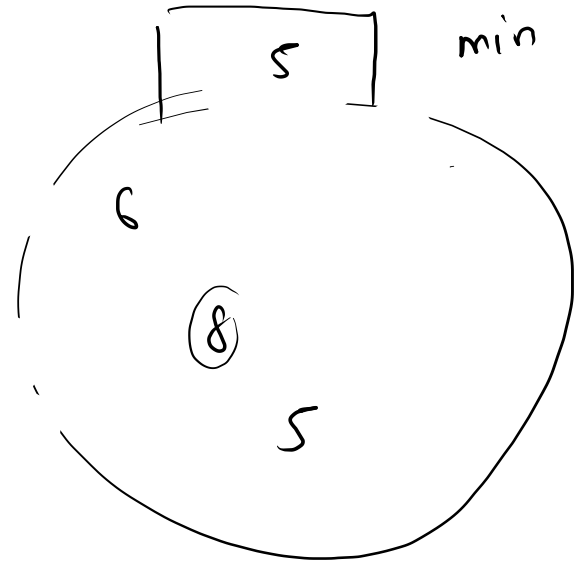
long v1 = 0;
long v2 = 0;

boolean even = true;

while(pq.size() > 0){
    if(even){
        v1 = v1 * 10 + pq.remove();
        even = false;
    }
    else{
        v2 = v2 * 10 + pq.remove();
        even = true;
    }
}

System.out.println(v1 + v2);
```

6 8 4 5 2 3



Subtract numbers.

Smallest positive

Sample Input 0

5
1 5 0 3 5

Sample Output 0

3

1st
2nd
3rd

1	5	0	3	5
0	4	0	<u>2</u>	4
0	<u>2</u>	0	0	2
0	0	0	0	0

ans = ~~2~~
3

fre unique number

Ans

1st

2nd

3rd

4th

5th

5	4	3	2	1
4	3	2	1	0
3	2	1	0	0
2	1	0	0	0
1	0	0	0	0
0	0	0	0	0

$$a_4 = 5$$

④

	5	4	3	2	1
3		2	1	0	2
2		1	0	0	1
1		0	0	0	0
0		0	0	0	0

ans = 4

```
public static void main(String[] args) {  
    Scanner scn = new Scanner(System.in);  
    int n = scn.nextInt();  
    HashSet<Integer> hs = new HashSet<>();  
    for(int i = 0; i < n; i++){  
        int val = scn.nextInt();  
        if(val > 0){  
            hs.add(val);  
        }  
    }  
    System.out.println(hs.size());  
}
```