1046. Last Stone Weight

Easy 2809 59 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 <= y`. The result of this smash is:

- If `x == y`, both stones are destroyed, and
- If `x != 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 smallest possible weight of the left stone*. If there are no stones left, return `0`.

Example 1:

```
Input: stones = [2,7,4,1,8,1]
Output: 1
Explanation:
We combine 7 and 8 to get 1 so the array converts to [2,4,1,1,1] then,
we combine 2 and 4 to get 2 so the array converts to [2,1,1,1] then,
we combine 2 and 1 to get 1 so the array converts to [1,1,1] then,
we combine 1 and 1 to get 0 so the array converts to [1] then that's the value of the last stone.
```

Example 2:

```
Input: stones = [1]
Output: 1
```

Constraints:

- `1 <= stones.length <= 30`
- `1 <= stones[i] <= 1000`

Accepted 256,999 Submissions 401,042

Java Autocomplete

```java
class Solution {
    public int lastStoneWeight( int[] stones ) {

        PriorityQueue<Integer> stoneQueue = new PriorityQueue<>(Collections.reverseOrder());

        for (int stone : stones) {
            stoneQueue.add(stone);
        }

        while (stoneQueue.size() != 1) {

            int first = stoneQueue.remove();
            int second = stoneQueue.remove();

            stoneQueue.add(Math.abs(first - second));
        }

        return stoneQueue.peek();
    }
}
```

Testcase Run Code Result Debugger

Accepted Runtime: 1 ms

Your input `[2,7,4,1,8,1]`

Output `1` Diff

Expected `1`