Last Stone Weight

Solution

We have a collection of stones, each stone has a positive integer weight.

Each turn, we choose the two **heaviest** stones and smash them together. Suppose the stones have weights `x` and `y` with `x <= y`. The result of this smash is:

- If `x == y`, both stones are totally destroyed;
- If `x != y`, the stone of weight `x` is totally destroyed, and the stone of weight `y` has new weight `y-x`.

At the end, there is at most 1 stone left. Return the weight of this stone (or 0 if there are no stones left.)

Example 1:

```
Input: [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 last stone.
```

Note:

1. `1 <= stones.length <= 30`
2. `1 <= stones[i] <= 1000`

Hide Hint #1

Simulate the process. We can do it with a heap, or by sorting some list of stones every time we take a turn.

Java

```java
class Solution {
    public int lastStoneWeight(int[] stones) {
        PriorityQueue<Integer> q = new PriorityQueue<Integer>((a,b) -> {
             return b - a;
        });

        for(int s : stones) { // o(nlogn)
            q.add(s);
        }

        while(q.size() > 1) { // o(n) since every smashing remove 1 item
            int stone1 = q.remove(); // logn
            int stone2 = q.remove(); // logn
            int diff = Math.abs(stone1-stone2);
            if(diff != 0) {
                q.add(diff); // logn
            }
        }

        return q.size() != 0 ? q.remove() : 0;
    }
}
```

Custom Testcase (Contribute)

Run Code Submit