1. 题目
2. 题干

有一堆石头，每块石头的重量都是正整数。

每一回合，从中选出两块 最重的 石头，然后将它们一起粉碎。假设石头的重量分别为 x 和 y，且 x <= y。那么粉碎的可能结果如下：

如果 x == y，那么两块石头都会被完全粉碎；

如果 x != y，那么重量为 x 的石头将会完全粉碎，而重量为 y 的石头新重量为 y-x。

最后，最多只会剩下一块石头。返回此石头的重量。如果没有石头剩下，就返回 0。

1. 示例

输入：[2,7,4,1,8,1]

输出：1

解释：

先选出 7 和 8，得到 1，所以数组转换为 [2,4,1,1,1]，

再选出 2 和 4，得到 2，所以数组转换为 [2,1,1,1]，

接着是 2 和 1，得到 1，所以数组转换为 [1,1,1]，

最后选出 1 和 1，得到 0，最终数组转换为 [1]，这就是最后剩下那块石头的重量。

1. 题解
2. 思路

思路很简单，首先对数组进行排序，每次选择数组最后两个数字进行比较。主要是实现的技术问题，这里需要使用一个可以实现动态伸缩的数据结构，并且需要对该书结构进行排序，这里可以选择优先队列，也可以选择栈。动态地删除比较过的数字，直到只剩下一个数字，返回即可，

同时，也可以使用普通的数组，不需要动态地删除容量，而是通过替换为0来表示该数已经被比较过了。每次比较后将数组中比较的两个数替换为两数相减后的值和一个0，再进行排序，这样0就会到数组的前方，我们永远比较的是数组的最后两个数，直到倒数第二个数也为0，则说明只有一个数了，这个数就是结果。

1. 代码实现

Java：

1. 数组实现

class Solution {

    public int lastStoneWeight(int[] stones) {

        int n = stones.length;

        if(n == 1){

            return stones[0];

        }

        Arrays.sort(stones);

        while(stones[stones.length-2] != 0){

            stones[stones.length-1] = stones[stones.length-1] - stones[stones.length-2];

            stones[stones.length-2] = 0;

            Arrays.sort(stones);

        }

        return stones[stones.length-1];

    }

}

1. 优先队列

class Solution {

    public int lastStoneWeight(int[] stones){

        PriorityQueue<Integer> queue = new PriorityQueue<>(new Comparator<Integer>() {

            @Override

            public int compare(Integer o1, Integer o2) {

                return o2.compareTo(o1);

            }

        });

        for(int stone : stones){

            queue.offer(stone);

        }

        while(queue.size() >= 2){

            int newStone = queue.poll() - queue.poll();

            queue.offer(newStone);

        }

        return queue.poll();

    }

}