# 题目

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。

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。

提示：

1 <= stones.length <= 30

1 <= stones[i] <= 1000

# 分析

## 方法一：最大堆

将所有石头的重量放入最大堆中。每次依次从队列中取出最重的两块石头a和b，必有a≥b。如果 a>ba>b，则将新石头a−b放回到最大堆中；如果a=b，两块石头完全被粉碎，因此不会产生新的石头。重复上述操作，直到剩下的石头少于2块。

最终可能剩下1块石头，该石头的重量即为最大堆中剩下的元素，返回该元素；也可能没有石头剩下，此时最大堆为空，返回0。

class Solution {

public:

int lastStoneWeight(vector<int>& stones) {

priority\_queue<int> q;

for (int s: stones) {

q.push(s);

}

while (q.size() > 1) {

int a = q.top();

q.pop();

int b = q.top();

q.pop();

if (a > b) {

q.push(a - b);

}

}

return q.empty() ? 0 : q.top();

}

};

复杂度分析：

时间复杂度：O(nlogn)，其中n是石头数量。每次从队列中取出元素需要花费O(logn) 的时间，最多共需要粉碎n−1次石头。

空间复杂度：O(n)。