```
/*
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。
提示：
    1 <= stones.length <= 30
    1 <= stones[i] <= 1000
来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/last-stone-weight
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

- 方法一:循环排序,选出最大的两个数进行比较,直到不为0的石头的数量≤ 1;
- 方法一的复杂度:每次排序$O(n \log n)$,排序次数$O(n)$,所以事件复杂度为$O(n^2 \log n)$
- 方法二:使用优先队列管理,取出最大的两个后,将石头粉碎后的结果再次压入队列(只需要最多压入一个即可),直到第二大的数为0,就直接返回第一大的数即可.
- 优先队列内部使用堆排序,内部Heapify过程的时间复杂度是$O(n)$,每次新插入或取出一个元素的时间复杂度为$O(\log n)$,综合而言n个元素的原数组使用队列实现本题的时间复杂度为$n \log n$.

方法一:C++_暴力排序法

```
class Solution
{
    public:
        int lastStoneWeight(vector<int>& stones)
        {
            if(stones.size()==1)
            {
                return stones[0];
            }
            else if(stones.size()<=0)
            {
                return 0;
            }
            else
            {
                while (1)
                {
                    sort(stones.begin(),stones.end(),greater<int>());
                    if(stones[1]==0)
                    {
                        return stones[0];
                    }
                    else
                    {
                        stones[0] = stones[0]-stones[1];
                        stones[1] = 0;
                    }
```

```
                }
            }
        }
};
/*
执行结果：
通过
显示详情
执行用时 :4 ms, 在所有 C++ 提交中击败了84.90% 的用户
内存消耗 :8.3 MB, 在所有 C++ 提交中击败了100.00%的用户
*/
```

方法二:C++_优先队列

```
class Solution
{

    public:
        int lastStoneWeight(vector<int>& stones)
        {
            priority_queue<int> pqi;
            int maxdata1 = 0;
            int maxdata2 = 0;
            int i        = 0;

            for(i = 0;i<stones.size();i++)
            {
                pqi.push(stones[i]);
            }

            while (pqi.size() > 1)
            {
                maxdata1 = pqi.top();
                pqi.pop();
                maxdata2 = pqi.top();
                pqi.pop();
                if(maxdata1!=maxdata2)
                {
                    pqi.push(maxdata1-maxdata2);
                }
            }

            if(pqi.size() ==0)
            {
                return 0;
            }
            else
            {
                return pqi[0];
            }
        }
};
/*
执行结果：
通过
显示详情
```

```
43 执行用时 :4 ms, 在所有 C++ 提交中击败了84.90% 的用户
44 内存消耗 :8.3 MB, 在所有 C++ 提交中击败了100.00%的用户
45 */
```

AlimyBreak
2019.08.27