# 题目

楼下水果店正在促销，你打算买些苹果，arr[i] 表示第 i 个苹果的单位重量。

你有一个购物袋，最多可以装 5000 单位重量的东西，算一算，最多可以往购物袋里装入多少苹果。

示例 1：

输入：arr = [100,200,150,1000]

输出：4

解释：所有 4 个苹果都可以装进去，因为它们的重量之和为 1450。

示例 2：

输入：arr = [900,950,800,1000,700,800]

输出：5

解释：6 个苹果的总重量超过了 5000，所以我们只能从中任选 5 个。

提示：

1 <= arr.length <= 10^3

1 <= arr[i] <= 10^3

# 分析

## 方法一：排序/贪心

思路

本题要求最多可以买到的苹果数量，固定容量的购物袋，苹果的重量越小，能放入的数量越多。比如 100 容量的购物袋，可以放 10 个重量为 10 的苹果，只能放 5 个重量为 20 的苹果。要保证放的苹果最多，需要尽可能的保证每个苹果的重量最小。

因此，我们可以逐个放入苹果，每次放入的苹果都选择剩余的苹果中最小的那个，直到总的容量超过购物袋的容量。

算法

对数组 arr 排序，快速找到苹果中重量最小的那个。从小到大遍历 arr 数组，将当前下标的苹果加入到购物袋，将重量加到当前总重量 sum 中， 如果 sum > 5000 说明已经放不下了，返回之前所有苹果的个数。如果整个数组的和小于 5000，说明可以放下所有的苹果。

代码

class Solution {

public:

int maxNumberOfApples(vector<int>& arr) {

sort(arr.begin(), arr.end());

int sum = 0;

int count = 0;

for(int v : arr) {

sum += v;

if(sum > 5000){

break;

}

count++;

}

return count;

}

};

复杂度分析

时间复杂度：O(NlogN)，其中n为数组arr的长度。排序算法为快排，时间复杂度为O(NlogN)，遍历arr数组的时间复杂度为O(N)。

空间复杂度：O(logN)，排序算法为快排，空间复杂度为O(logN)。

另一种写法：

class Solution

{

public:

int maxNumberOfApples(vector<int>& arr)

{

sort(arr.begin(), arr.end());

int res = 0;

int cur\_sum = 0;

for (int x: arr)

{

cur\_sum += x;

if (cur\_sum > 5000)

break;

res ++;

}

return res;

}

};