# 题目

给定一个非空的正整数数组nums，请判断能否将这些数字分成元素和相等的两部分。

示例 1：

输入：nums = [1,5,11,5]

输出：true

解释：nums 可以分割成 [1, 5, 5] 和 [11] 。

示例 2：

输入：nums = [1,2,3,5]

输出：false

解释：nums不可以分为和相等的两部分

提示：

1 <= nums.length <= 200

1 <= nums[i] <= 100

注意：本题与主站416题相同：

https://leetcode-cn.com/problems/partition-equal-subset-sum/

# 分析

## 方法一：动态规划

class Solution {

public:

bool canPartition(vector<int>& nums) {

int sum = accumulate(nums.begin(), nums.end(), 0);

// 和奇数直接滚蛋

if (sum & 1) return false;

// 转换为背包问题，是否能够

int target = sum / 2;

int n = nums.size();

vector<vector<int>> dp(n + 1, vector<int>(target + 1));

for (int i = 1; i <= n; ++i)

for (int j = 0; j <= target; ++j) {

dp[i][j] = dp[i - 1][j];

if (j >= nums[i - 1])

dp[i][j] = max(dp[i][j], dp[i - 1][j - nums[i - 1]] + nums[i - 1]);

}

return dp[n][target] == target;

}

};

或：

class Solution {

public:

bool canPartition(vector<int>& nums) {

int len = nums.size();

if(len==1) return false;

int sum = 0;

for(int i : nums) sum += i;

if(sum&1) return false;

int tar = sum >> 1 ;

bool dp[tar+1];

memset(dp,0,sizeof(dp));//初始化

dp[0] = true;

for(int i =0 ; i< len ; ++i){

for(int j=tar ; j>=nums[i] ; --j){

if(dp[j-nums[i]]==true) dp[j]=true;

}

}

return dp[tar];

}

};

## 方法二：路径压缩

bool canPartition(vector<int>& nums) {

int sum = accumulate(nums.begin(), nums.end(), 0);

if (sum & 1) return false;

int target = sum / 2;

vector<int> dp(target + 1);

for (const int num : nums)

for (int j = target; j >= num; --j)

dp[j] = max(dp[j], dp[j - num] + num);

return dp[target] == target;

}