题目：

You are given a list of non-negative integers, a1, a2, ..., an, and a target, S. Now you have 2 symbols `+` and `-`. For each integer, you should choose one from `+` and `-` as its new symbol.

Find out how many ways to assign symbols to make sum of integers equal to target S.

Example 1:

```
Input: nums is [1, 1, 1, 1, 1], S is 3.
Output: 5
Explanation:

-1+1+1+1+1 = 3
+1-1+1+1+1 = 3
+1+1-1+1+1 = 3
+1+1+1-1+1 = 3
+1+1+1+1-1 = 3

There are 5 ways to assign symbols to make the sum of nums be target 3.
```

Note:

1. The length of the given array is positive and will not exceed 20.
2. The sum of elements in the given array will not exceed 1000.
3. Your output answer is guaranteed to be fitted in a 32-bit integer.

1.时间：O（2^N）;空间：O（N）

```
class Solution {
public:
    int findTargetSumWays(vector<int>& nums, int S) {
        if (nums.empty()) return 0;
        int result = 0;
        dfs(nums, S, 0, result);
        return result;
    }
private:
    void dfs(const std::vector<int>& nums, const int remain, const int index, int&
result){
        if (index == nums.size()){
            if (remain == 0) result++;
            return;
        }
        dfs(nums, remain - nums[index], index + 1, result);
        dfs(nums, remain + nums[index], index + 1, result);
    }
};
```

2.时间：O（N*SUM）; 空间：O（SUM）

```
class Solution {
```

```
public:
    int findTargetSumWays(vector<int>& nums, int S) {
        if (nums.empty()) return 0;
        int sum = std::accumulate(nums.begin(), nums.end(), 0);
        if (sum < S || -sum > S) return 0;
        std::vector<int> dp(2 * sum + 1, 0);
        dp[sum + 0] = 1;
        for (int i = 0; i < nums.size(); ++i){
            std::vector<int> next(2 * sum + 1, 0);
            for (int k = 0; k < 2 * sum + 1; ++k){
                if (dp[k] != 0){
                    next[k + nums[i]] += dp[k];
                    next[k - nums[i]] += dp[k];
                }
            }
            dp = std::move(next);
        }
        return dp[sum + S];
    }
};
```