给定一个非负整数数组，a1, a2, ..., an, 和一个目标数，S。现在你有两个符号 `+` 和 `-`。对于数组中的任意一个整数，你都可以从 `+` 或 `-`中选择一个符号添加在前面。

返回可以使最终数组和为目标数 S 的所有添加符号的方法数。

示例 1:

```
输入: nums: [1, 1, 1, 1, 1], S: 3
输出: 5
解释:

-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

一共有5种方法让最终目标和为3。
```

注意:

1. 数组的长度不会超过20，并且数组中的值全为正数。
2. 初始的数组的和不会超过1000。
3. 保证返回的最终结果为32位整数。

思路，该题可以用深度遍历，查找每一种运算的可能性，在中间进行优化剪纸，若是当前的和加上以后全部的和还小于目标数，则不需要继续遍历了。同理，若减去后面的数还大于，也不需要了。然后动态统计相等的个数。

```
class Solution {
public:
    void dfs(vector<int>& nums,vector<int> &sums,int sum,int s,int &num,int i){
        if(i<0){
            if(s==sum){
                num++;
            }
            return ;
        }
        if(sum+sums[i]>=s){
            dfs(nums,sums,sum+nums[i],s,num,i-1);
        }
```

```
        if(sum-sums[i]<=s){
            dfs(nums,sums,sum-nums[i],s,num,i-1);
        }
    }
    int findTargetSumWays(vector<int>& nums, int S) {
        int result=0;
        vector<int> sums;
        int sum=0;
        for(int num:nums){
            sum+=num;
            sums.push_back(sum);
        }
        dfs(nums,sums,0,S,result,nums.size()-1);
        return result;
    }
};
```