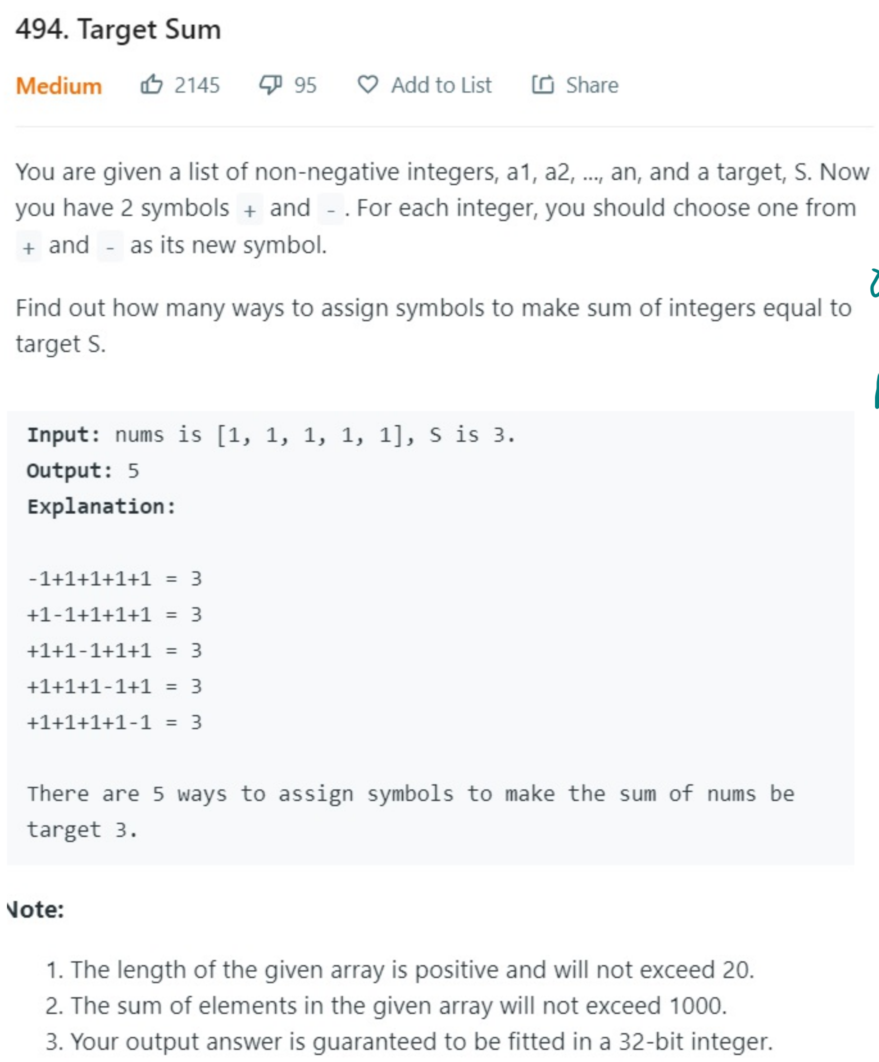


494. Target Sum

Medium 2145 95 Add to List Share

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.

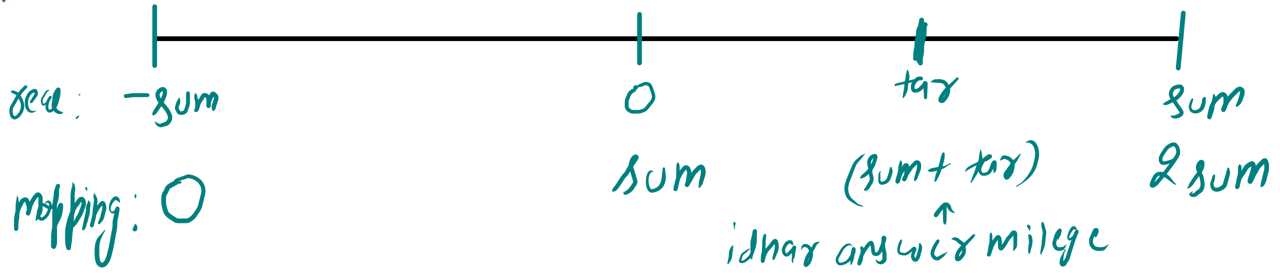
```
Input: nums is [1, 1, 1, 1, 1], S is 3.
Output: 5
Explanation:

-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.
```

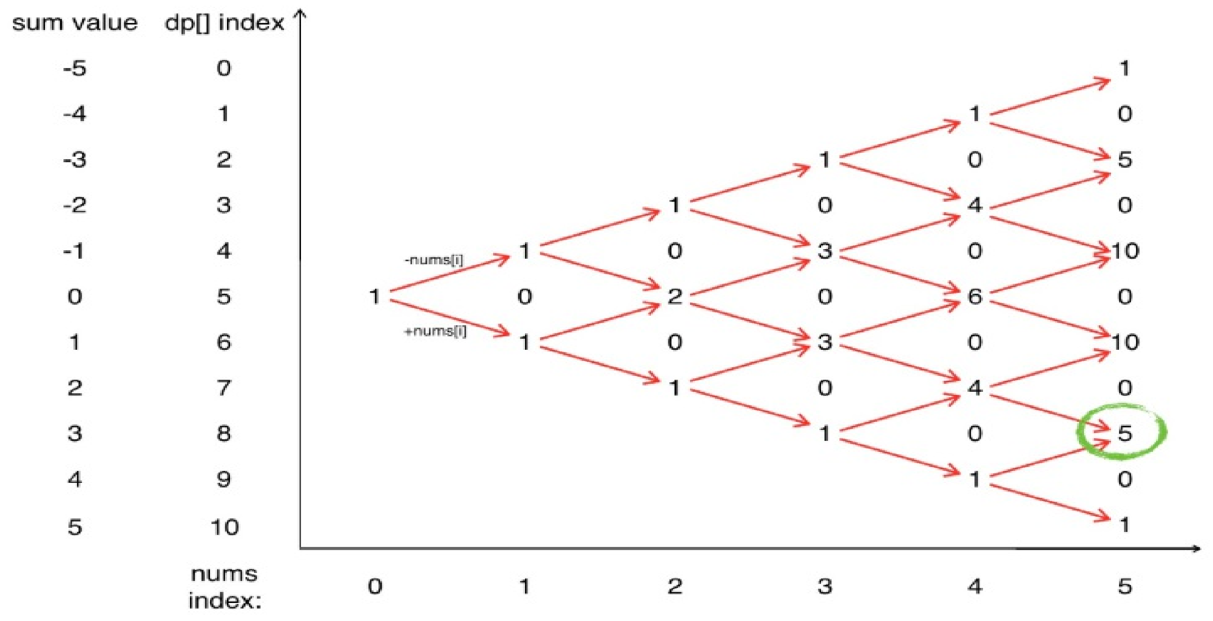
Note:

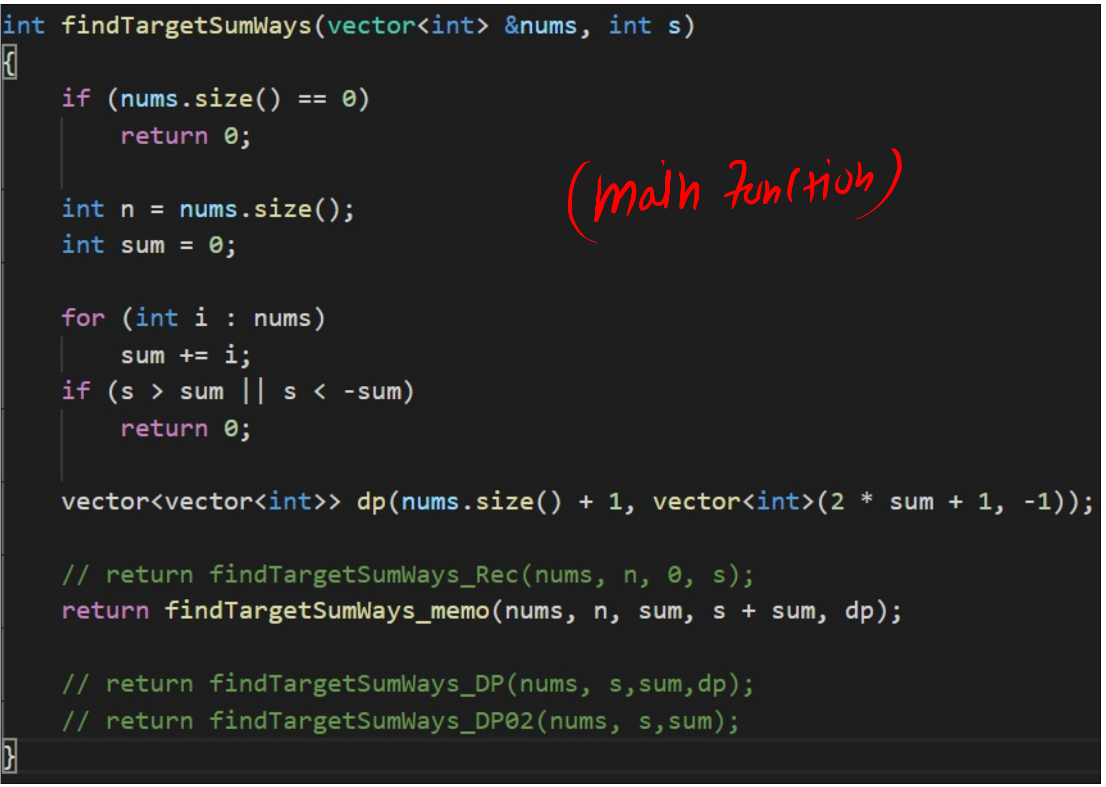
1. The length of the given array is positive and will not exceed 20.
2. The sum of elements in the given array will not exceed 1000.
3. Your output answer is guaranteed to be fitted in a 32-bit integer.

agar sarre number as a negative add krenge to -sum bnega. or agar sabhi ko as a positive add krenge to +sum bnega.

because dp mein negative index nahi hota hai isliye -sum map to 0, 0 map to sum, and sum map to 2*sum.

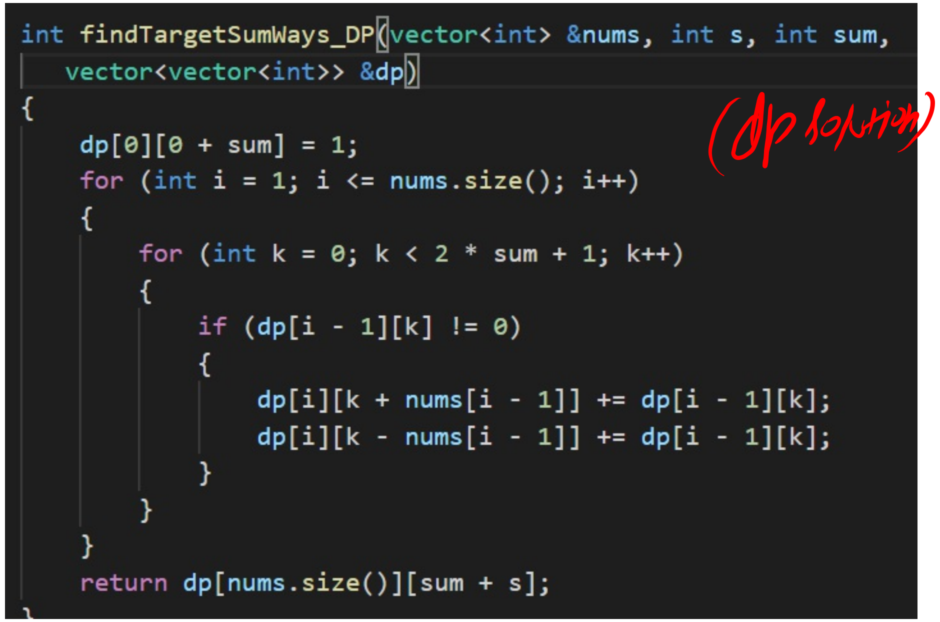






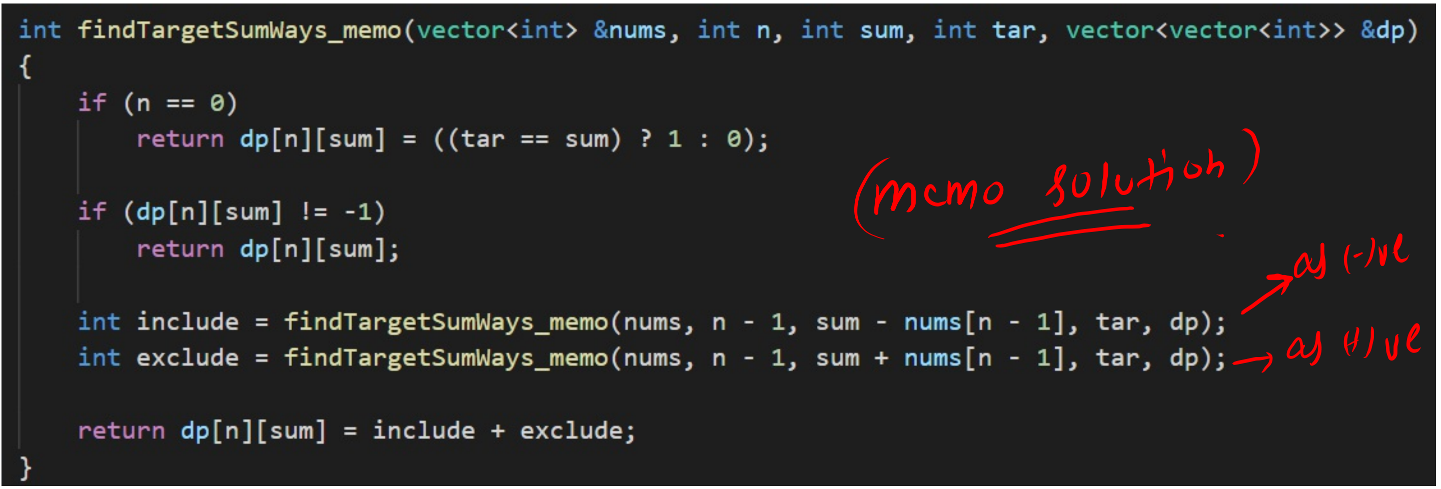
```cpp
int findTargetSumWays(vector<int> &nums, int s)
{
    if (nums.size() == 0)
        return 0;

    int n = nums.size();
    int sum = 0;

    for (int i : nums)
        sum += i;
    if (s > sum || s < -sum)
        return 0;

    vector<vector<int>> dp(nums.size() + 1, vector<int>(2 * sum + 1, -1));

    // return findTargetSumWays_Rec(nums, n, 0, s);
    return findTargetSumWays_memo(nums, n, sum, s + sum, dp);

    // return findTargetSumWays_DP(nums, s,sum,dp);
    // return findTargetSumWays_DP02(nums, s,sum);
}
```

(main function)



```cpp
int findTargetSumWays_DP(vector<int> &nums, int s, int sum,
   vector<vector<int>> &dp)
{
    dp[0][0 + sum] = 1;
    for (int i = 1; i <= nums.size(); i++)
    {
        for (int k = 0; k < 2 * sum + 1; k++)
        {
            if (dp[i - 1][k] != 0)
            {
                dp[i][k + nums[i - 1]] += dp[i - 1][k];
                dp[i][k - nums[i - 1]] += dp[i - 1][k];
            }
        }
    }
    return dp[nums.size()][sum + s];
```

(dp solution)



```cpp
int findTargetSumWays_memo(vector<int> &nums, int n, int sum, int tar, vector<vector<int>> &dp)
{
    if (n == 0)
        return dp[n][sum] = ((tar == sum) ? 1 : 0);

    if (dp[n][sum] != -1)
        return dp[n][sum];

    int include = findTargetSumWays_memo(nums, n - 1, sum - nums[n - 1], tar, dp);
    int exclude = findTargetSumWays_memo(nums, n - 1, sum + nums[n - 1], tar, dp);

    return dp[n][sum] = include + exclude;
}
```

(memo solution)

$\rightarrow$ as (-)ve

$\rightarrow$ as (+)ve