Huahua's Tech Road
August 15, 2021

花花酱 LeetCode 494. Target Sum

BY ZXI ON JANUARY 10, 2018

花花酱 LeetCode 494. Target Sum 上 - 刷题找工作 E…

题目大意：给你一串数字，你可以在每个数字前放置+或-，问有多少种方法可以使得表达式的值等于target。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.

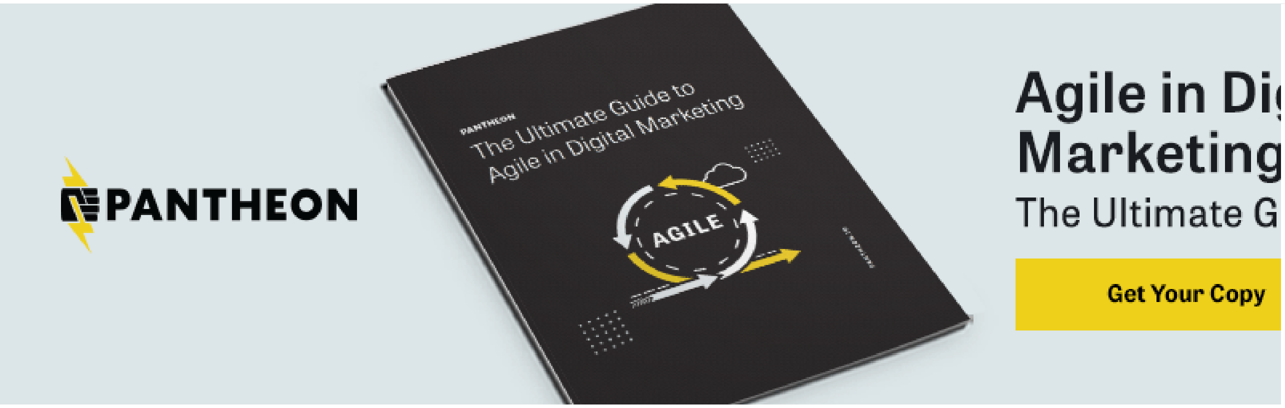
Example 1:

```
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.



Idea: DP

Why DP works?

Let's look at a simpler problem: Can the following equation can be true?

$\pm a_1 \pm a_2 \pm a_3 ... \pm a_n$ = target

$O(2^n)$ combinations, but
S = 2*sum(a) + 1 total possible sums, S <= 2000 + 1

We use V_i to denote the possible sums by using first i elements

$V_0 = \{0\}$
$V_i = \{V_{i-1} + a_i\} \cup \{V_{i-1} - a_i\}$
Check target in V_n

DP works because $|V_n| <= S << O(2^n)$

Time complexity is $Sum\{2*|V_i|\} <= n * S = O(n * S)$

input: [1,1,1,1,1], target = 3
2^5 = 32 combination, but total 6 distinct values, max is 11 (2*5 + 1)

i	a_i	V_i
0	-	{0}
1	1	{-1, 1}
2	1	{-2, 0, 2}
3	1	{-3, -1, 1, 3}
4	1	{-4, -2, 0, 2, 4}
5	1	{-5, -3, -1, 1, 3, 5}

494. Target Sum

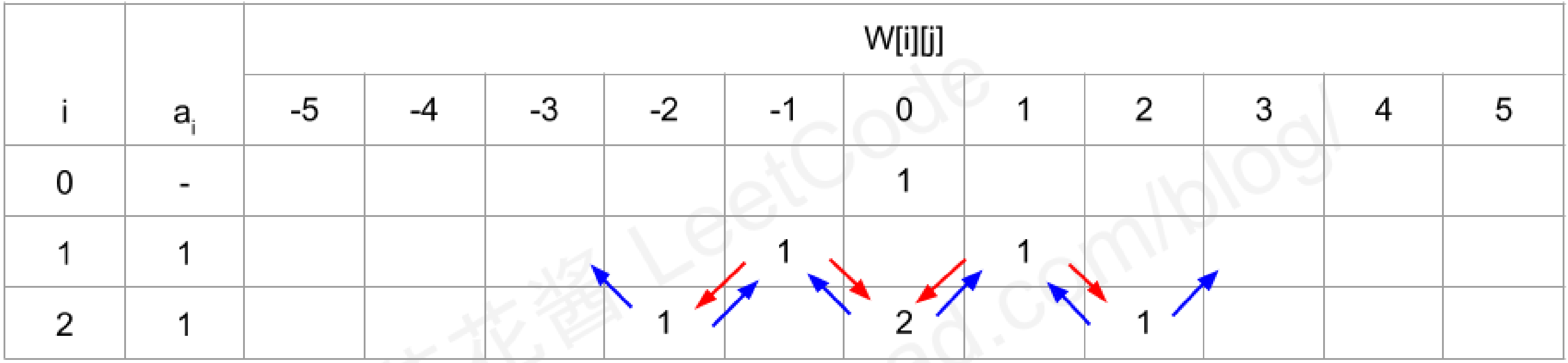
Solution 1:
Brute force: DFS
Time complexity $O(2^n)$, n = 20, AC 585 ms
Space complexity: O(n)

```
Solution 2:
DP

ways[i][j] # of ways to sum up to j using nums[0~i]
ways[i][j] =   ways[i - 1][j - nums[i]]
             + ways[i - 1][j + nums[i]]

Init: ways[-1][0] = 1, one way to sum up 0, do nothing
Ans:  ways[n-1][S]

Time complexity O(n * sum), 16 ms
Space complexity: O(n * sum) -> O(sum)
```

input: [1,1,1,1,1], target = 3
sum = 5, range = -5 ~ 5

		W[i][j]										
i	a_i	-5	-4	-3	-2	-1	0	1	2	3	4	5
0	-						1					
1	1					1		1				
2	1				1		2		1			
3	1			1		3		3		1		
4	1		1		4		6		4		1	
5	1	1		5		10		10		5		1

Sum of (W[5][j]) is 32 = 2^5



i	a_i	W[i][j]										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
0	-						1					
1	1					1		1				
2	1				1		2		1			

Transition 1: Push

```
Scan j for W[i - 1]

W[i][j - num_i] += W[i - 1][j]
W[i][j + num_i] += W[i - 1][j]
```

Transition 2: Pull

```
Scan j for W[i]

W[i][j] =    W[i - 1][j - num_i]
           + W[i - 1][j + num_i]
```

Optimization: Subset sum

Let **P** denotes a set of nums have a + sign in front of it

Let **N** denotes a set of nums have a - sign in front of it

$P \cup N = \{a_1, a_2, \ldots, a_n\} \mid P \cap N = \emptyset$

sum(**P**) - sum(**N**) = target

sum(**P**) ~~- sum(**N**)~~ + sum(**P**) + ~~sum(**N**)~~ = target + sum(**P**) + sum(**N**)

2*sum(**P**) = target + sum(a)

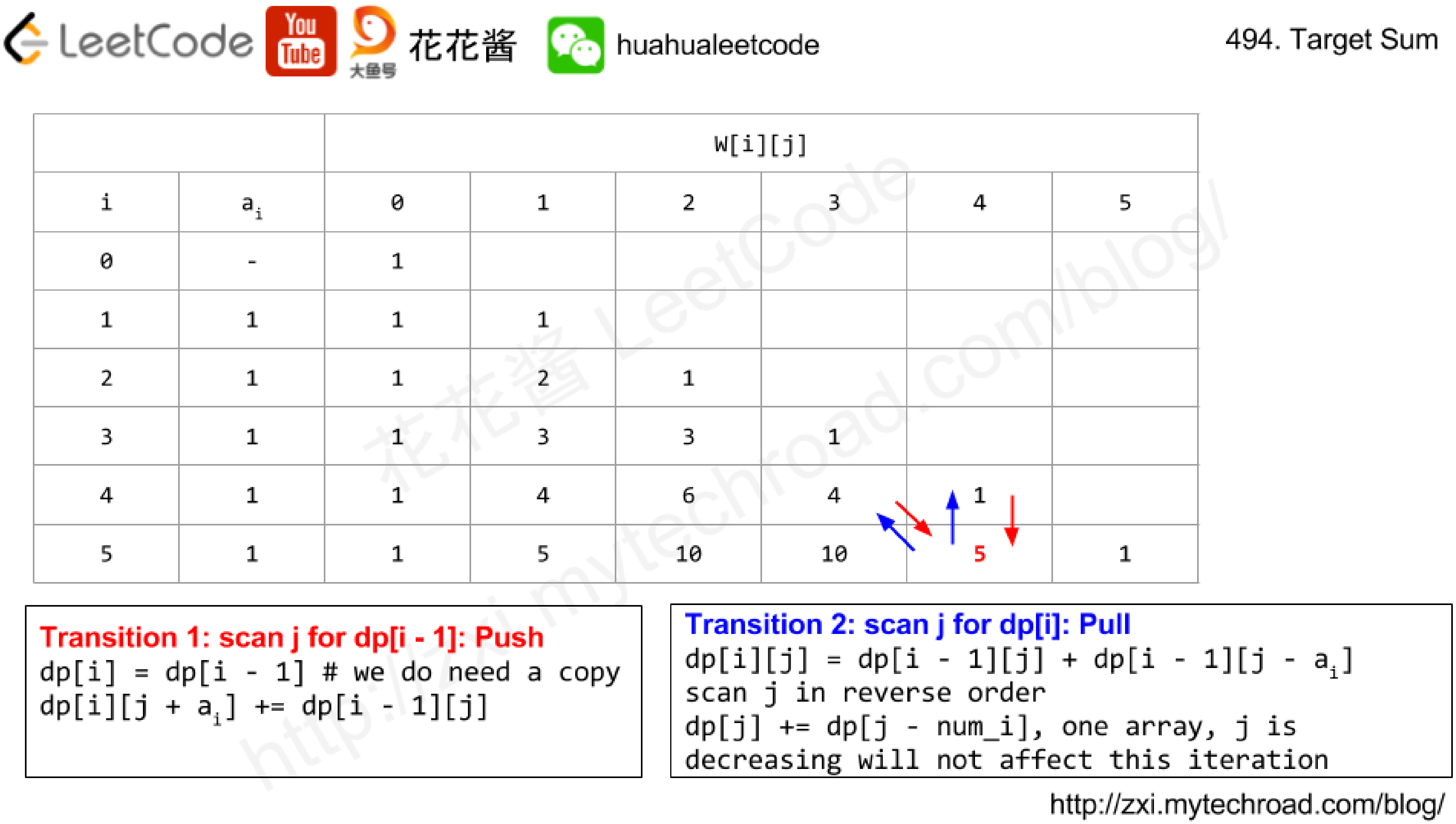
sum(**P**) = (target + sum(a)) / 2 <- 0-1 Knapsack problem

Simpler questions: using the given nums, can be sum up to target?
We use V_i to denote the possible sums by using any subset of the first i elements
$V_0 = \{0\}$
$V_i = \{V_{i-1}\} \cup \{V_{i-1} + a_i\}$ # V_i contains V_{i-1} , do we need a copy?
Ans: Check target in V_n
dp[i][j] := whether we can use the first i elements to sum up to j / j in V_i

Init: dp[0][0] = True

```
Push: scan j for dp[i - 1]
for i in 1..n:
  for j in 0..S:
   if dp[i - 1][j]:
     dp[i][j + $a_i$] = True
```

```
Pull: scan j for dp[i]
for i in 1..n:
  for j in 0..S:
    dp[i][j] = dp[i-1][j] or dp[i-1][j - $a_i$]
```



LeetCode 花花酱 huahualeetcode

494. Target Sum

		W[i][j]					
i	a_i	0	1	2	3	4	5
0	-	1					
1	1	1	1				
2	1	1	2	1			
3	1	1	3	3	1		
4	1	1	4	6	4	1	
5	1	1	5	10	10	5	1

Transition 1: scan j for dp[i - 1]: Push

dp[i] = dp[i - 1] # we do need a copy
dp[i][j + a_i] += dp[i - 1][j]

Transition 2: scan j for dp[i]: Pull

dp[i][j] = dp[i - 1][j] + dp[i - 1][j - a_i]
scan j in reverse order
dp[j] += dp[j - num_i], one array, j is
decreasing will not affect this iteration

http://zxi.mytechroad.com/blog/

Solution 1: DP

Time complexity: O(n*sum)

Space complexity: O(n*sum)

C++

```cpp
// Author: Huahua
// Running time: 16 ms
class Solution {
public:
    int findTargetSumWays(vector<int>& nums, int S) {
      const int n = nums.size();
      const int sum = std::accumulate(nums.begin(), nums.end(), 0);
      if (sum < S) return 0;
      const int offset = sum;
      // ways[i][j] means total ways to sum up to (j - offset) using nums[0] ~ nums[i - 1].
      vector<vector<int>> ways(n + 1, vector<int>(sum + offset + 1, 0));
      ways[0][offset] = 1;
      for (int i = 0; i < n; ++i) {
        for (int j = nums[i]; j < 2 * sum + 1 - nums[i]; ++j)
          if (ways[i][j]) {
            ways[i + 1][j + nums[i]] += ways[i][j];
            ways[i + 1][j - nums[i]] += ways[i][j];
          }
      }

      return ways.back()[S + offset];
    }
};
```

C++ SC O(n)

```cpp
// Author: Huahua
// Running time: 12 ms
class Solution {
public:
    int findTargetSumWays(vector<int>& nums, int S) {
      const int n = nums.size();
      const int sum = std::accumulate(nums.begin(), nums.end(), 0);
      if (sum < std::abs(S)) return 0;
      const int kOffset = sum;
      const int kMaxN = sum * 2 + 1;
      vector<int> ways(kMaxN, 0);
      ways[kOffset] = 1;
      for (int num : nums) {
        vector<int> tmp(kMaxN, 0);
        for (int i = num; i < kMaxN - num; ++i)
          if (ways[i]) {
            tmp[i + num] += ways[i];
            tmp[i - num] += ways[i];
          }
        std::swap(ways, tmp);
      }
      return ways[S + kOffset];
    }
};
```

Java

```
// Author: Huahua
// Running time: 28 ms
class Solution {
  public int findTargetSumWays(int[] nums, int S) {
    int sum = 0;
    for (final int num : nums)
      sum += num;
    if (sum < S) return 0;
    final int kOffset = sum;
    final int kMaxN = sum * 2 + 1;
    int[] ways = new int[kMaxN];
    ways[kOffset] = 1;
    for (final int num : nums) {
      int[] tmp = new int[kMaxN];
      for (int i = num; i < kMaxN - num; ++i) {
        tmp[i + num] += ways[i];
        tmp[i - num] += ways[i];
      }
      ways = tmp;
    }
    return ways[S + kOffset];
  }
}
```

C++ / V2

```
// Author: Huahua
// Running time: 17 ms
class Solution {
```

```
public:
  int findTargetSumWays(vector<int>& nums, int S) {
   const int n = nums.size();
   const int sum = std::accumulate(nums.begin(), nums.end(), 0);
   if (sum < std::abs(S)) return 0;
   const int kOffset = sum;
   const int kMaxN = sum * 2 + 1;
   vector<int> ways(kMaxN, 0);
   ways[kOffset] = 1;
   for (int num : nums) {
    vector<int> tmp(kMaxN, 0);
    for (int i = 0; i < kMaxN; ++i) {
     if (i + num < kMaxN) tmp[i] += ways[i + num];
     if (i - num >= 0)    tmp[i] += ways[i - num];
    }
    std::swap(ways, tmp);
   }
   return ways[S + kOffset];
  }
};
```

Solution 2: DFS

Time complexity: O(2^n)

Space complexity: O(n)

C++

```
// Author: Huahua
// Running time: 422 ms
class Solution {
public:
 int findTargetSumWays(vector<int>& nums, int S) {
  const int sum = std::accumulate(nums.begin(), nums.end(), 0);
  if (sum < std::abs(S)) return 0;
  int ans = 0;
  dfs(nums, 0, S, ans);
  return ans;
 }
private:
  void dfs(const vector<int>& nums, int d, int S, int& ans) {
    if (d == nums.size()) {
      if (S == 0) ++ans;
      return;
    }
    dfs(nums, d + 1, S - nums[d], ans);
    dfs(nums, d + 1, S + nums[d], ans);
  }
};
```

Java

```
// Author: Huahua
// Running time: 615 ms
class Solution {
  private int ans;
  public int findTargetSumWays(int[] nums, int S) {
    int sum = 0;
    for (final int num : nums)
      sum += num;
    if (sum < Math.abs(S)) return 0;
    ans = 0;
    dfs(nums, 0, S);
    return ans;
  }

  private void dfs(int[] nums, int d, int S) {
    if (d == nums.length) {
      if (S == 0) ++ans;
      return;
    }
    dfs(nums, d + 1, S - nums[d]);
    dfs(nums, d + 1, S + nums[d]);
  }
}
```

Solution 3: Subset sum

Time complexity: O(n*sum)

Space complexity: O(sum)

C++ w/ copy

```
// Author: Huahua
// Running time: 7 ms
class Solution {
public:
    int findTargetSumWays(vector<int>& nums, int S) {
      S = std::abs(S);
      const int sum = std::accumulate(nums.begin(), nums.end(), 0);
      if (sum < S || (S + sum) % 2 != 0) return 0;
      const int target = (S + sum) / 2;
      vector<int> dp(target + 1, 0);
      dp[0] = 1;
      for (int num : nums) {
        vector<int> tmp(dp);
        for (int j = 0; j <= target - num; ++j)
          tmp[j + num] += dp[j];
        std::swap(dp, tmp);
      }

      return dp[target];
    }
};
```

C++ w/o copy

```
// Author: Huahua
// Running time: 6 ms
class Solution {
public:
    int findTargetSumWays(vector<int>& nums, int S) {
      S = std::abs(S);
      const int n = nums.size();
      const int sum = std::accumulate(nums.begin(), nums.end(), 0);
      if (sum < S || (S + sum) % 2 != 0) return 0;
      const int target = (S + sum) / 2;
      vector<int> dp(target + 1, 0);
      dp[0] = 1;
      for (int num : nums)
        for (int j = target; j >= num; --j)
          dp[j] += dp[j - num];
      return dp[target];
    }
};
```

请尊重作者的劳动成果，转载请注明出处！花花保留对文章 / 视频的所有权利。
如果您喜欢这篇文章 / 视频，欢迎您捐赠花花。
If you like my articles / videos, donations are welcome.

Published in [Dynamic Programming] and [Medium]

combination DFS sum target

zxi

More from **Dynamic Programming**

More posts in Dynamic Programi

花花酱 LeetCode 1883. Minimum Skips to Arrive at Meeting On Time

花花酱 LeetCode 1879. Minimum XOR Sum of Two Arrays

花花酱 LeetCode 1872. Stone Game VIII

花花酱 LeetCode 1866. Number of Ways to Rearrange Sticks With K Sticks Visible

花花酱 LeetCode 1824. Minimum Sideway Jumps

More from **Medium**

More posts in Medium »

花花酱 LeetCode 769. Max Chunks To Make Sorted

花花酱 LeetCode 763. Partition Labels

花花酱 LeetCode 322. Coin Change

花花酱 LeetCode 652. Find Duplicate Subtrees

花花酱 LeetCode 416. Partition Equal Subset Sum

Be First to Comment

Leave a Reply

You must be logged in to post a comment.

DONATION

如果您喜欢我们的内容，欢迎捐赠花花
If you like my blog, donations are welcome

Venmo 微信打赏 Paypal

Donate

Shopping on Amazon to support us
您可以通过在亚马逊购物来支持我们

CATEGORIES

Admin (7)

Array (112)

Bash (1)

Binary Search (46)

Bit (29)

Brute Force (1)

C++ (2)

Career (1)

Concurrent (5)

Contest (1)

CS (3)

Data Structure (18)

Desgin (6)

Divide and conquer (7)

Dynamic Programming (181)

Easy (24)

Game Theory (2)

Generation (1)

Geometry (41)

Graph (67)

Greedy (75)

Hard (17)

Hashtable (127)

Heap (7)

mytechroad.com

Binary Search – Huahua’s Tech Road

mytechroad.com

Two pointers – Huahua’s Tech Road

mytechroad.com

花花酱 LeetCode 301. Remove Invalid Parentheses

mytechroad.com

花花酱 LeetC Serialize and Deserialize B

mytechroad.com

花花酱 LeetC Possible Full Trees

mytechroad.com

- Leetcode (35)
- Linux (1)
- List (30)
- Math (81)
- Medium (25)
- Priority Queue (5)
- Programming Language (3)
- Python (3)
- Queue (4)
- Randomization (1)
- Recursion (12)
- Search (80)
- Simulation (82)
- Sliding Window (14)
- SP (16)
- SQL (3)
- Stack (19)
- String (127)
- Template (1)
- Tree (109)
- Trie (2)
- Two pointers (23)
- Uncategorized (19)
- ZOJ (3)

YOUTUBE CHANNEL

Subscribe bilibili

TAGS

META

array BFS binary search bit BST combination counting design DFS dp easy frequency geometry graph greedy grid hard hashtable heap list math matrix medium O(n) Palindrome permutation prefix prefix sum priority queue recursion search shortest path simulation sliding window sort sorting stack string subarray subsequence sum tree two pointers union find xor

Log in

Entries feed

Comments feed

WordPress.org