

494. Target Sum

Description

Hints

Submissions

Discuss

Solution

Pick One

You are given a list of non-negative integers, a_1, a_2, \dots, a_n , 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:

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

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

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

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

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

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.

【问题分析】

- 1、该问题求解数组中数字只和等于目标值的方案个数，每个数字的符号可以为正或负(减整数等于加负数)。
- 2、该问题和矩阵链乘很相似，是典型的动态规划问题
- 3、举例说明: $\text{nums} = \{1,2,3,4,5\}$, $\text{target}=3$, 一种可行的方案是 $+1-2+3-4+5 = 3$

该方案中数组元素可以分为两组，一组是数字符号为正($P=\{1,3,5\}$)，另一组数字符号为负($N=\{2,4\}$)

因此: $\text{sum}(1,3,5) - \text{sum}(2,4) = \text{target}$

$$\text{sum}(1,3,5) - \text{sum}(2,4) + \text{sum}(1,3,5) + \text{sum}(2,4) = \text{target} + \text{sum}(1,3,5) + \text{sum}(2,4)$$

$$2\text{sum}(1,3,5) = \text{target} + \text{sum}(1,3,5) + \text{sum}(2,4)$$

$$2\text{sum}(P) = \text{target} + \text{sum}(\text{nums})$$

$$\text{sum}(P) = (\text{target} + \text{sum}(\text{nums})) / 2$$

由于 target 和 $\text{sum}(\text{nums})$ 是固定值，因此原始问题转化为求解 nums 中子集的和等于 $\text{sum}(P)$ 的方案个数问题

- 4、求解 nums 中子集只和为 $\text{sum}(P)$ 的方案个数(nums 中所有元素都是非负)

该问题可以通过动态规划**算法**求解

举例说明：给定集合 $\text{nums}=\{1,2,3,4,5\}$, 求解子集，使子集中元素之和等于9 = $\text{new_target} = \text{sum}(P) = (\text{target} + \text{sum}(\text{nums}))/2$

定义 $\text{dp}[10]$ 数组, $\text{dp}[10] = \{1,0,0,0,0,0,0,0,0,0\}$

$\text{dp}[i]$ 表示子集合元素之和等于当前目标值的方案个数, 当前目标值等于9减去当前元素值

当前元素等于1时, $\text{dp}[9] = \text{dp}[9] + \text{dp}[9-1]$

```

public class L494 {
    /*
     * 这道题目用动态规划
     */
    public int findTargetSumWays(int[] nums, int S) {
        int sum = 0;

        for(int i = 0; i < nums.length; i++) {
            sum += nums[i];
        }

        if(S > sum || (sum + S) % 2 == 1) {
            return 0;
        }
        return subsetSum(nums, (sum + S) / 2);
    }

    private int subsetSum(int [] nums, int S) {
        int [] dp = new int [S + 1];
        dp[0] = 1;
        //因为这里是从nums[0]到nums[nums.length]看有多少之和为dp[S]。
        for(int i = 0; i < nums.length; i++) {
            //dp[j]表示从nums[0]到nums[i]之间和等于dp[j]的个数
            for(int j = S; j >= nums[i]; j--) {
                dp[j] += dp[j - nums[i]];
            }
        }
        return dp[S];
    }
}

```