**377. Combination Sum IV: -**

Medium Accepted: 389.7K Submissions: 741.1K Acceptance Rate: 52.6%

Given an array of **distinct** integers nums and a target integer target, return *the number of possible combinations that add up to* target.

The test cases are generated so that the answer can fit in a **32-bit** integer.

**Example 1:**

**Input:** nums = [1,2,3], target = 4

**Output:** 7

**Explanation:**

The possible combination ways are:

(1, 1, 1, 1)

(1, 1, 2)

(1, 2, 1)

(1, 3)

(2, 1, 1)

(2, 2)

(3, 1)

Note that different sequences are counted as different combinations.

**Example 2:**

**Input:** nums = [9], target = 3

**Output:** 0

**Constraints:**

* 1 <= nums.length <= 200
* 1 <= nums[i] <= 1000
* All the elements of nums are **unique**.
* 1 <= target <= 1000

**Follow up:** What if negative numbers are allowed in the given array? How does it change the problem? What limitation we need to add to the question to allow negative numbers?

**Code: -**

class Solution {

public:

    int helper(vector<int>& nums, int target, vector<int>& dp){

        // base case

        if(target == 0)         return 1;

        if(target < 0)          return 0;

        // dp found case

        if(dp[target] != -1)

            return dp[target];

        // calling for every possible solution

        int ways=0;

        for(int i=0; i<nums.size(); ++i)

            ways = ways + helper(nums, target-nums[i], dp);

        return dp[target] = ways;

    }

    int combinationSum4(vector<int>& nums, int target) {

        vector<int> dp(target+1, -1);

        return helper(nums, target, dp);

    }

};

**T.C: - O(target \* N)**

**S.C: - O(target)**