Partitions with Given Difference

Given an array arr, partition it into two subsets(possibly empty) such that each element must belong to only one subset. Let the sum of the elements of these two subsets be S1 and S2.

Given a difference d, count the number of partitions in which S1 is greater than or equal to S2 and the difference between S1 and S2 is equal to d. Since the answer may be large return it modulo 109 + 7.

Example 1:

Input:

n = 4

d = 3

arr[] = { 5, 2, 6, 4}

Output: 1

Explanation:

There is only one possible partition of this array. Partition : {6, 4}, {5, 2}. The subset difference between subset sum is: (6 + 4) - (5 + 2) = 3.

Example 2:

Input:

n = 4

d = 0

arr[] = {1, 1, 1, 1}

Output: 6

Explanation:

we can choose two 1's from indices {0,1}, {0,2}, {0,3}, {1,2}, {1,3}, {2,3} and put them in S1 and remaning two 1's in S2.

Thus there are total 6 ways for partition the array arr.

Your Task:

You don't have to read input or print anything. Your task is to complete the function countPartitions() which takes the integer n and d and array arr and returns the count of partitions.

Expected Time Complexity: O( n\*sum(arr))

Expected Space Complexity: O( sum(arr))

Constraint:

1 <= n <= 500

0 <= d <= 25000

0 <= arr[i] <= 50

class Solution {

public:

int countPartitions(int n, int d, vector<int>& a) {

int mod =1e9 +7;

int sum = accumulate(a.begin(), a.end(), 0);

if ((sum + d) % 2 != 0) {

return 0;

}

int target = (sum + d) / 2;

vector<int> dp(target + 1, 0);

dp[0] = 1;

for (int i = 0; i < n; i++) {

for (int j = target; j >= a[i]; j--) {

dp[j] = (dp[j] + dp[j - a[i]]) %mod;

}

}

return dp[target];

}

};

Link : <https://www.geeksforgeeks.org/problems/partitions-with-given-difference/1>