**Perfect Sum Problem: -**

Medium Accuracy: 20.58% Submissions: 204K+ Points: 4

Given an array **arr[]** of non-negative integers and an integer **sum**, the task is to count all subsets of the given array with a sum equal to a given **sum**.

Note: Answer can be very large, so, output answer modulo 109+7

**Example 1:**

**Input**: N = 6, arr[] = {2, 3, 5, 6, 8, 10}

sum = 10

**Output:** 3

**Explanation**: {2, 3, 5}, {2, 8}, {10} are   
possible subsets.

**Example 2:**

**Input**: N = 5, arr[] = {1, 2, 3, 4, 5}

sum = 10

**Output:** 3

**Explanation**: {1, 2, 3, 4}, {1, 4, 5},

{2, 3, 5} are possible subsets.

**Your Task:**  
You don't need to read input or print anything. Complete the function **perfectSum()**which takes **N,**array**arr[]** and **sum**as input parameters and returns an integer value

**Expected Time Complexity:** O(**N\*sum**)  
**Expected Auxiliary Space:** O(**N\*sum**)  
  
**Constraints:**  
1 ≤ **N\*sum** ≤ 106

0<=arr[I]<=106

**Code: -**

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution{

public:

int mod = 1000000000+7;

int helper(int arr[], int ind, int target, vector<vector<int>> &dp){

// base case

if(ind == -1){

if(target == 0) return 1;

else return 0;

}

if(target < 0) return 1;

// dp found case

if(dp[ind][target] != -1)

return dp[ind][target];

// recursive cases

int take\_child = 0, nottake\_child = 0;

// take that current item

if(target >= arr[ind])

take\_child = helper(arr, ind-1, target-arr[ind], dp);

// don't take that current item

nottake\_child = helper(arr, ind-1, target, dp);

// return from current state

return (dp[ind][target] = (take\_child + nottake\_child) % mod);

}

int perfectSum(int arr[], int n, int sum){

vector<vector<int>> dp(n+1, vector<int>(sum+1, -1));

return helper(arr, n-1, sum, dp);

}

};

//{ Driver Code Starts.

int main()

{

int t;

cin >> t;

while (t--)

{

int n, sum;

cin >> n >> sum;

int a[n];

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

cin >> a[i];

Solution ob;

cout << ob.perfectSum(a, n, sum) << "\n";

}

return 0;

}

// } Driver Code Ends

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

**S.C: - O(N \* sum), without recursive call**