**Problem link**: [Partitions With Given Difference](https://www.codingninjas.com/codestudio/problems/partitions-with-given-difference_3751628?source=youtube&campaign=striver_dp_videos&utm_source=youtube&utm_medium=affiliate&utm_campaign=striver_dp_videos&leftPanelTab=0)

**Approach**: Acc to the ques

s1- s2 = d …(i)

S1+s2 = totalSum …(ii)

After solving eqn i) & ii)

S2 = (totalSum-d)/2;

Now, the question boils down to find subsets with target = (totalSum-d)/2

Edge cases: i) (totalSum-d) < 0; sum should be +ve

ii) (totalSum-d)%2 ==1 (odd); so that we will not get a float value when divide it by 2

**1) recursion:** tc = O(2^n), sc = O(N) for recursion stack space

**//recursive soln**

**//s2 = (totalSum-d)/2**

**int partitions(int ind, int target, vector<int> &arr){**

**if(ind==0){**

**if(target==0 && arr[ind]==0) return 2; //take it or not**

**if(target==0 || target==arr[ind]) return 1;**

**return 0;**

**}**

**int notTake = partitions(ind-1, target, arr);**

**int take = 0;**

**if(arr[ind]<= target)**

**take = partitions(ind-1, target-arr[ind], arr);**

**return (take+notTake);**

**}**

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

**int totalSum= 0;**

**for(auto &it: arr) totalSum += it;**

**if(totalSum-d<0 || (totalSum-d)%2) return 0;**

**return partitions(n-1, (totalSum-d)/2, arr);**

**}**

2) **Memoization**: dp[n][target+1]

tc= O(n\* target)

sc = O(n\* target) + O(N)

**//memoization soln**

**//s2 = (totalSum-d)/2**

**int mod = 1e9+7;**

**int partitions(int ind, int target, vector<int> &arr, vector<vector<int>> dp){**

**if(ind==0){**

**if(target==0 && arr[ind]==0) return 2; //take it or not**

**if(target==0 || target==arr[ind]) return 1;**

**return 0;**

**}**

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

**int notTake = partitions(ind-1, target, arr, dp);**

**int take = 0;**

**if(arr[ind]<= target)**

**take = partitions(ind-1, target-arr[ind], arr, dp);**

**return dp[ind][target] = (take+notTake)%mod;**

**}**

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

**int totalSum= 0;**

**for(auto &it: arr) totalSum += it;**

**if(totalSum-d<0 || (totalSum-d)%2) return 0;**

**vector<vector<int>> dp(n, vector<int>((totalSum-d)/2+1, -1));**

**return partitions(n-1, (totalSum-d)/2, arr, dp);**

**}**

3) **Tabulation**: build a dp[n][target+1] table from bottom to up.

tc = O(n\* target)

sc = O(n\*target)

**//tabuliztion soln**

**//s2 = (totalSum-d)/2**

**int mod = (int) 1e9+7;**

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

**int totalSum= 0;**

**for(auto &it: arr)**

**totalSum += it;**

**if(totalSum-d<0 || (totalSum-d)%2)**

**return 0;**

**//dp[][]**

**int target = (totalSum-d)/2;**

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

**if(arr[0]==0) dp[0][0] = 2;**

**else**

**dp[0][0]= 1;**

**//target = 0, dp[0][0] =2;**

**if(arr[0] != 0 && arr[0]<= target) dp[0][arr[0]]= 1;**

**//build dp table**

**for(int ind= 1; ind<n; ind++){**

**for(int tar= 0; tar<= target; tar++){**

**int notTake = dp[ind-1][tar];**

**int take = 0;**

**if(arr[ind]<= tar)**

**take = dp[ind-1][tar-arr[ind]];**

**dp[ind][tar] = (take+notTake)%mod;**

**}**

**}**

**return dp[n-1][target];**

**}**

4) **Space optimization**: use prev[target+1] to store [i-1]th row values & curr[target+1] to store current row values

tc = O(n\*target)

sc = O(2\*target)

**//tabuliztion soln(space- optimized)**

**//s2 = (totalSum-d)/2**

**int mod = (int) 1e9+7;**

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

**int totalSum= 0;**

**for(auto &it: arr)**

**totalSum += it;**

**if(totalSum-d<0 || (totalSum-d)%2)**

**return 0;**

**//dp[][]**

**int target = (totalSum-d)/2;**

**vector<int> prev(target+1, 0), curr(target+1, 0);**

**if(arr[0]==0) prev[0] = 2;**

**else**

**prev[0]= 1;**

**//target = 0, dp[0][0] =2;**

**if(arr[0] != 0 && arr[0]<= target) prev[arr[0]]= 1;**

**//build dp table**

**for(int ind= 1; ind<n; ind++){**

**for(int tar= 0; tar<= target; tar++){**

**int notTake = prev[tar];**

**int take = 0;**

**if(arr[ind]<= tar)**

**take = prev[tar-arr[ind]];**

**curr[tar] = (take+notTake)%mod;**

**}**

**prev = curr;**

**}**

**return prev[target];**

**}**