Day 13

13 December 2022 22:00

Subset Sum Problem

Medium Accuracy: **32.0%** Submissions: **103K+** Points: **4**

Given an array of non-negative integers, and a value *sum*, determine if there is a subset of the given set with sum equal to given *sum*.

Example 1:

Input:
N = 6
arr[] = {3, 34, 4, 12, 5, 2}
sum = 9
Output: 1
Explanation: Here there exists a subset with sum = 9, 4+3+2 = 9.

non-negative integers
0 ≤ +ive

Sum

T.C

3	34	4	12	5	2
0	1	2	3	4	5

N=6

Sum = 9 (9) (9)

subset {5,4} {4, 3, 2}

Yes

3	34	4	12	5	2

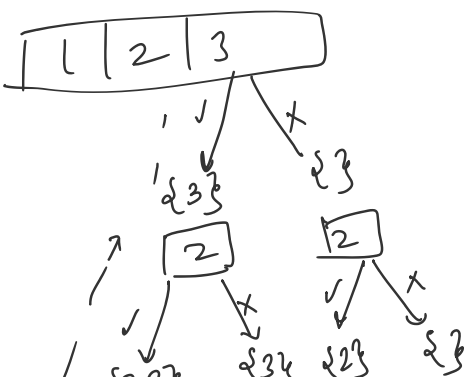
Sum=30

{3, 4, 12, 5, 2}

False

{26}

sum → time

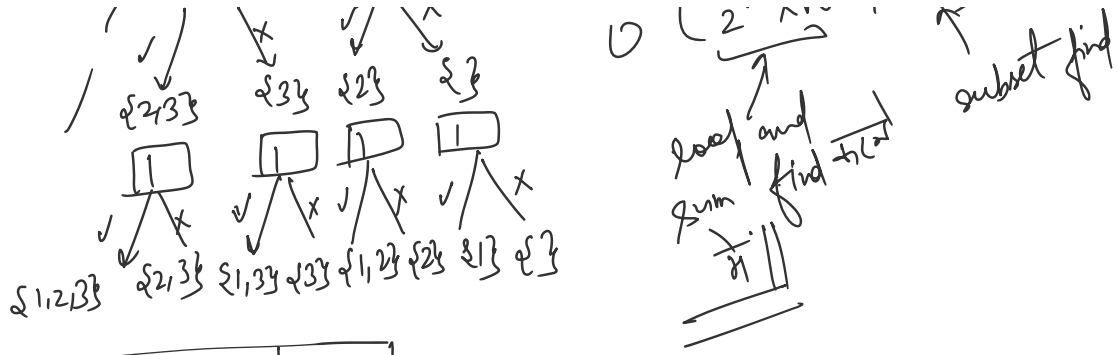


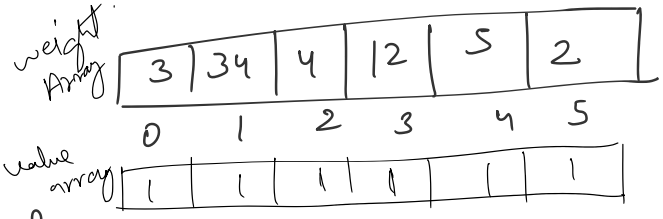
N → O(2^N x N)

[[1,2,3]

O(2^N x N + 2^N)

subset find





W=9

weight array

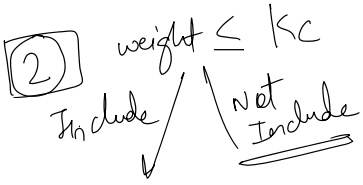
value array

if (weightItem ≤ kc)

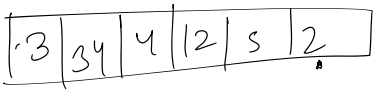
{ Included knapsack

Not Included

}

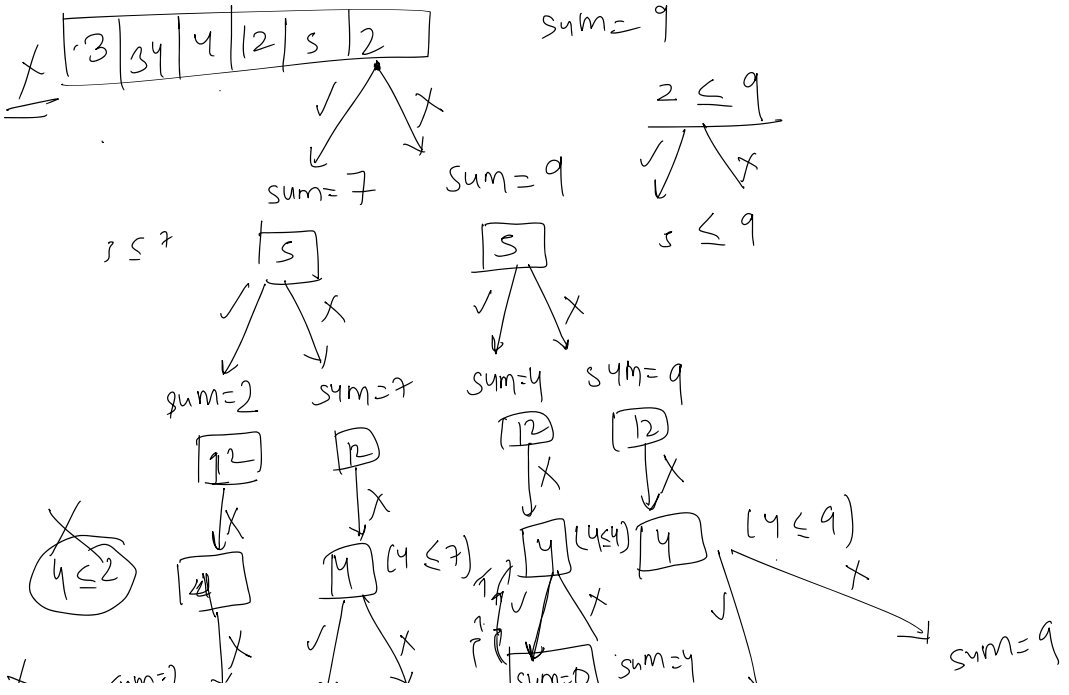


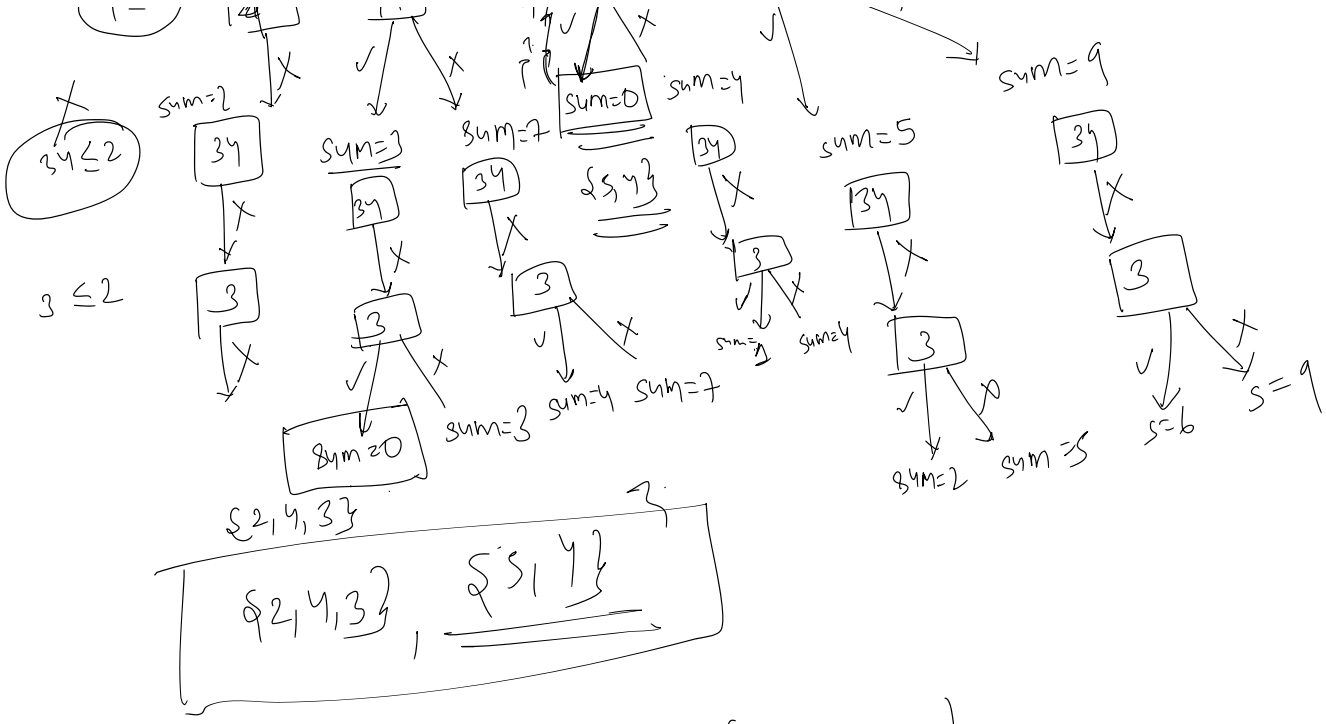
T.C



sum=9

2 and





Choice + Decision (Recursion)

```
class Solution{
public:
    bool solve(vector<int> &arr, int n, int sum){
        // base case
        if(sum == 0) return true;
        if(n == 0) return false;
        if(arr[n-1] <= sum){
          //  include                          not include
          return solve(arr,n-1,sum-arr[n-1]) || solve(arr,n-1,sum);
        }else{
            return solve(arr,n-1,sum);
        }
    }
    bool isSubsetSum(vector<int>arr, int sum){
        int n = arr.size();
        return solve(arr,n,sum);
    }
};
```

Time Limited Exceed.

T.C- O(2^N)

S.C-O(N)

① Change variable

2- (2D)

3 — (3D)

n - (n D)

sum+1

n+1

```cpp
class Solution{
public:
    bool solve(vector<int> &arr, int n, int sum,vector<vector<int>>& dp){
        // base case
        if(sum == 0) return true;
        if(n == 0) return false;
        if(dp[n][sum]!= -1) return dp[n][sum];
        if(arr[n-1] <= sum){
          //  include                          not include
          return dp[n][sum] = solve(arr,n-1,sum-arr[n-1],dp) || solve(arr,n-1,sum,dp);
        }else{
            return dp[n][sum] = solve(arr,n-1,sum,dp);
        }
    }
    bool isSubsetSum(vector<int>arr, int sum){
        int n = arr.size();
        vector<vector<int>> dp(n+1,vector<int>(sum+1,-1));
        return solve(arr,n,sum,dp);
    }
};
//} Driver Code Ends
```

(n+1) x (sum+1)

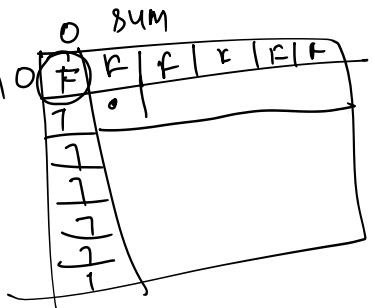
memoization

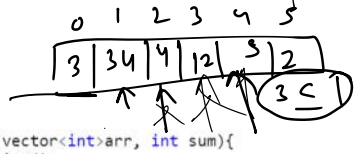
recursion stack overflow

T.C → O(n x sum)

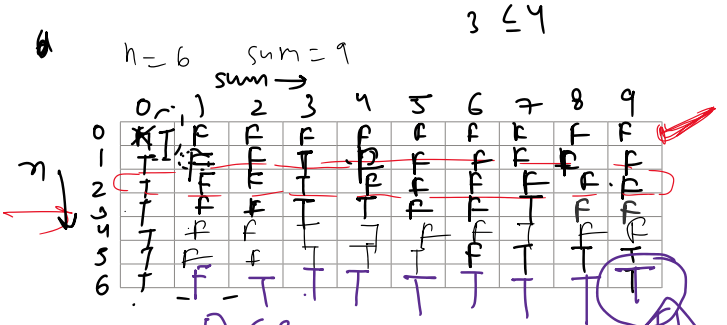
S.C → O(n x sum)

Iterative (Tabulation) (Bottom to Top)





3 ≤ 4




```cpp
bool isSubsetSum(vector<int>arr, int sum){
    int n = arr.size();
    vector<vector<int>> dp(n+1,vector<int>(sum+1));
    // return solve(arr,n,sum,dp);
    // dp[i][j]
    // i-> n
    // j-> sum
    for(int j = 0;j < sum+1;j++){
        dp[0][j] = false;
    }
    for(int i = 0;i < n+1;i++){
        dp[i][0] = true;
    }

    for(int i = 1;i < n+1;i++){
        for(int j = 1;j < sum+1;j++){
            if(arr[i-1] <= sum){
              //  include                          not include
                dp[i][j] = dp[i-1][j-arr[i-1]] || dp[i-1][j];
            }else{
                dp[i][j] = dp[i-1][j];
            }
        }
    }
    return dp[n][sum];
}
```

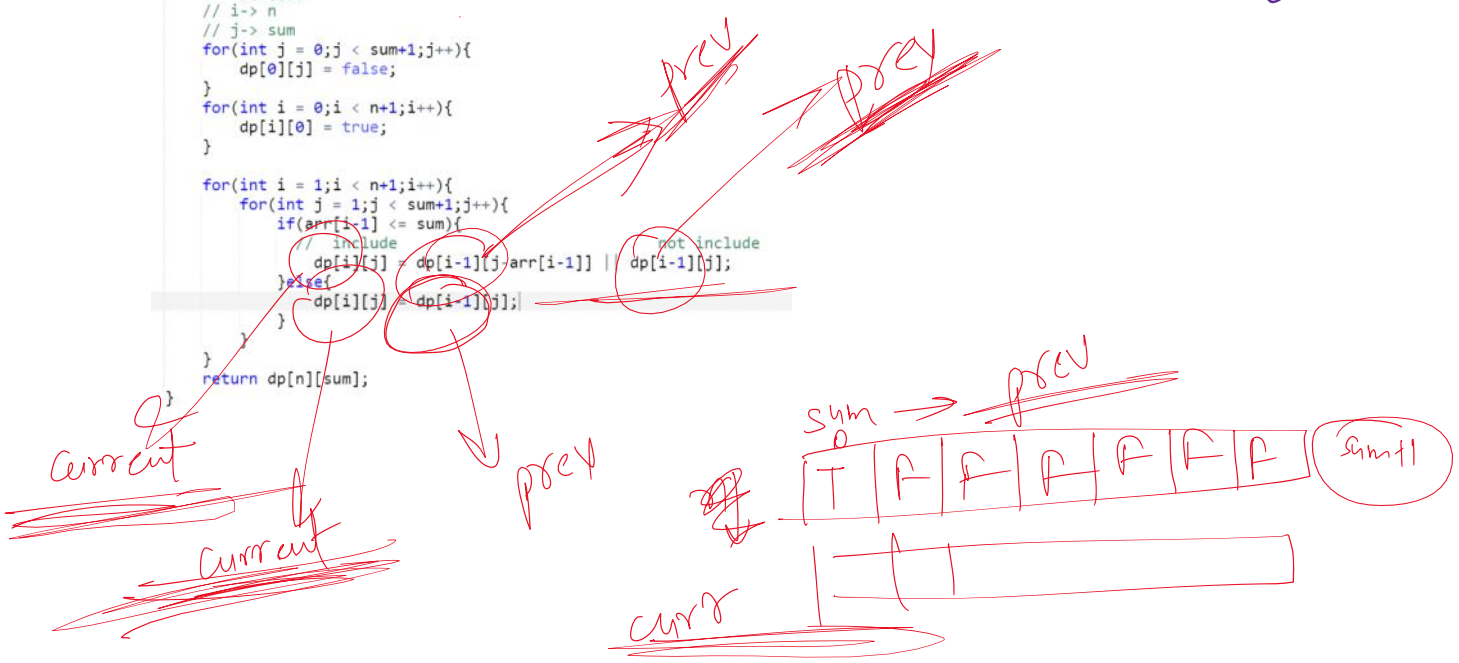
T.C → O(n x sum)

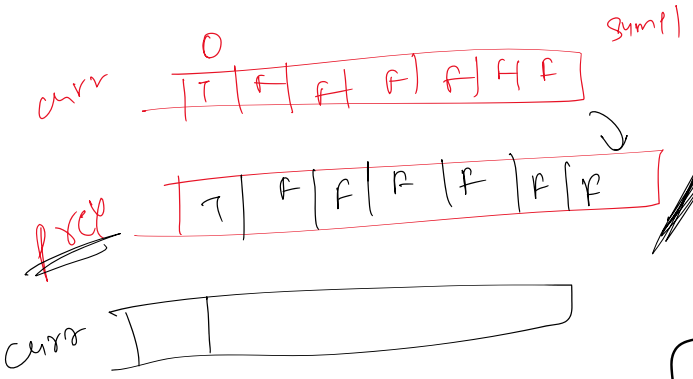
S.C → O(n x sum)

Recursion (80%) → Memoization (10%) → Tabulation (10%) → Space Optimized

```cpp
bool isSubsetSum(vector<int>arr, int sum){
    int n = arr.size();
    vector<vector<int>> dp(n+1,vector<int>(sum+1));
    // return solve(arr,n,sum,dp);
    // dp[i][j]
    // i-> n
    // j-> sum
    for(int j = 0;j < sum+1;j++){
        dp[0][j] = false;
    }
    for(int i = 0;i < n+1;i++){
        dp[i][0] = true;
    }

    for(int i = 1;i < n+1;i++){
        for(int j = 1;j < sum+1;j++){
            if(arr[i-1] <= sum){
                // include                          not include
                dp[i][j] = dp[i-1][j-arr[i-1]] || dp[i-1][j];
            }else{
                dp[i][j] = dp[i-1][j];
            }
        }
    }
    return dp[n][sum];
}
```



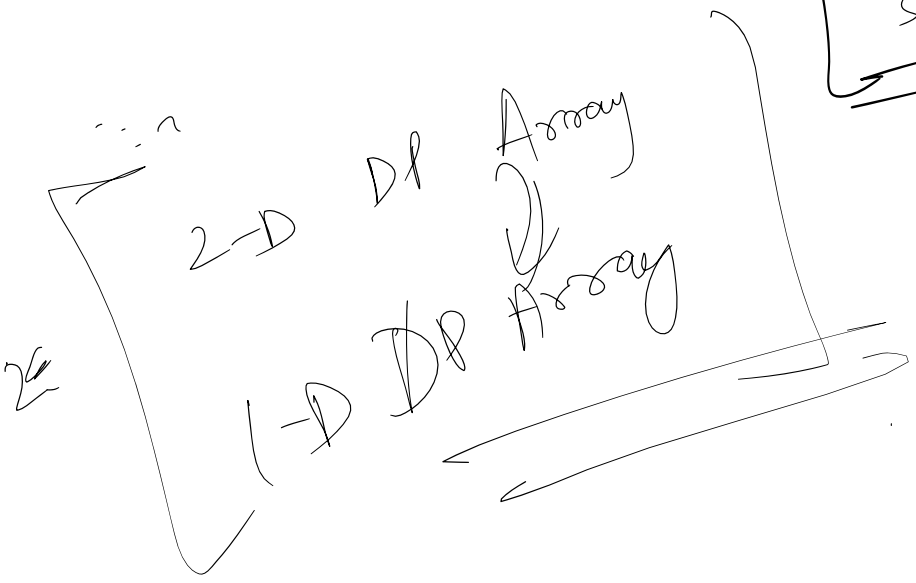



```cpp
vector<int> curr(sum+1,0);
curr[0] = 1;

for(int i = 1;i < n+1;i++){
    auto prev = curr;
    for(int j = 1;j < sum+1;j++){
        if(arr[i-1] <= sum){
            // include                          not include
            curr[j] = prev[j-arr[i-1]] || prev[j];
        }else{
            curr[j] = prev[j];
        }
    }
}
return curr[sum];
```

T.C- O(n x sum)

S.C- O(2 x sum)



Partition Equal Subset Sum

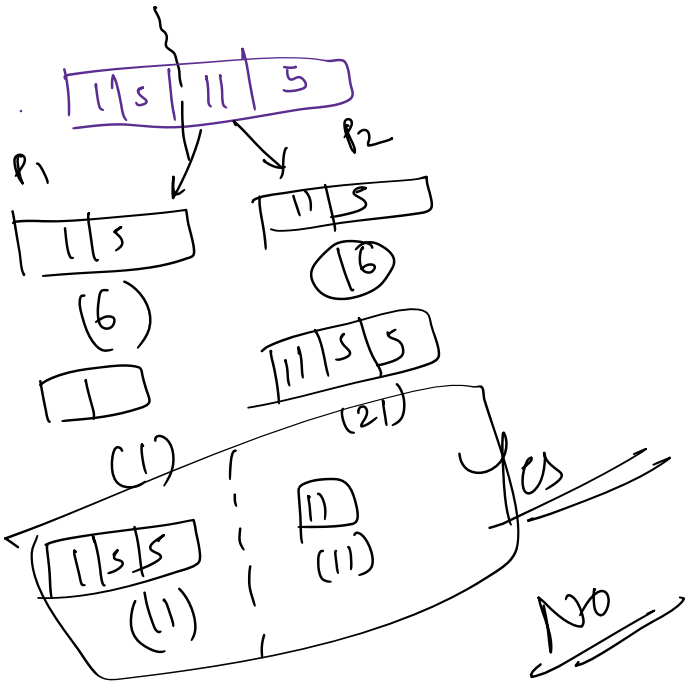
Medium Accuracy: **30.24%** Submissions: **149K+** Points: **4**

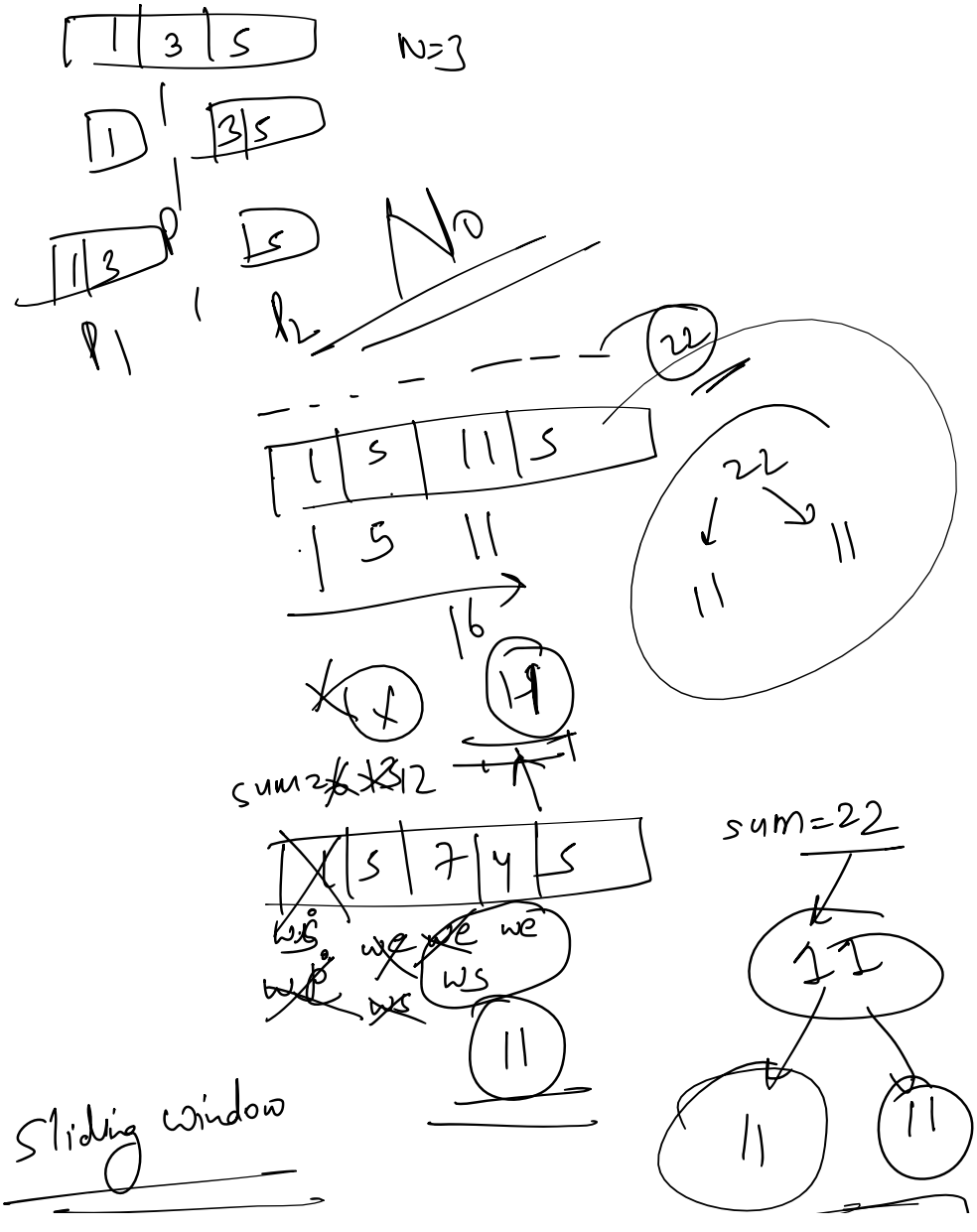
This problem is part of GFG SDE Sheet. Click here to view more.

Given an array **arr[]** of size **N**, check if it can be partitioned into two parts such that the sum of elements in both parts is the same.

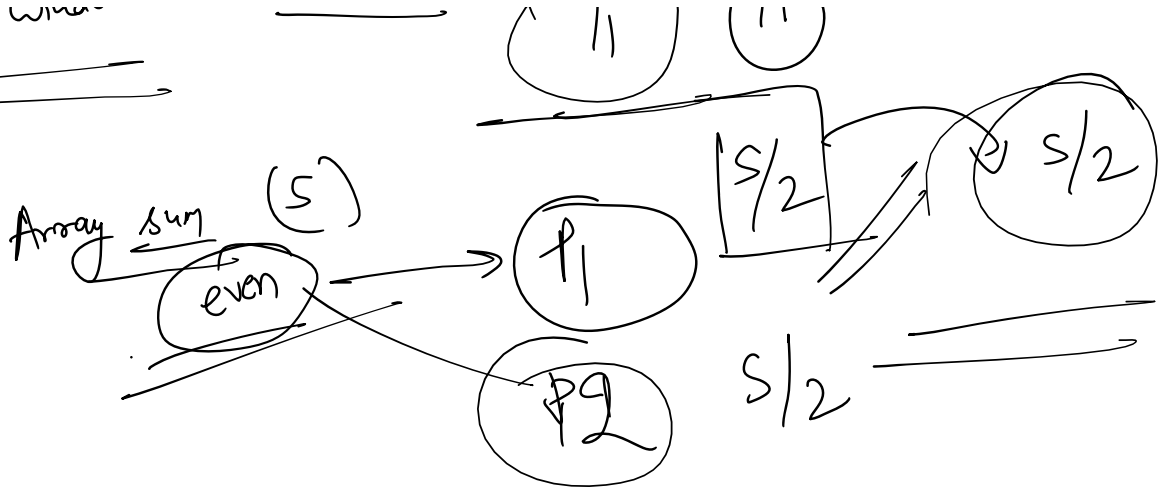
Example 1:

Input: N = 4
arr = {1, 5, 11, 5}
Output: YES
Explanation:
The two parts are {1, 5, 5} and {11}.





Sliding window



(1) Sum of Array

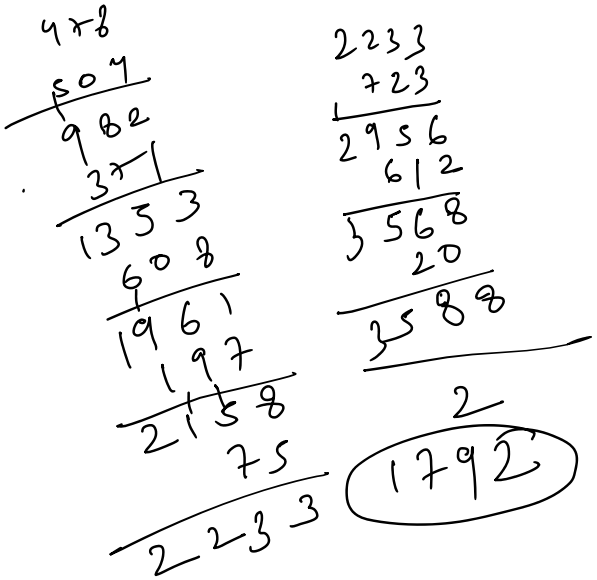
(2) Sum is even

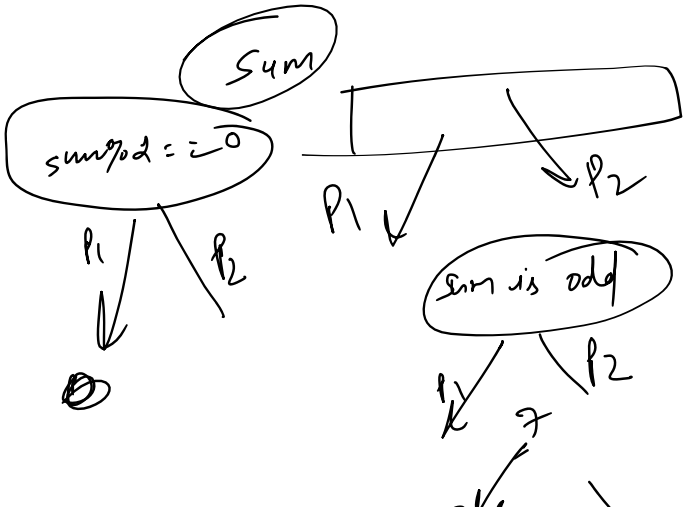
(1) 478 504 371 608 197 75 723 612 20

From <https://practice.geeksforgeeks.org/problems/subset-sum-problem2014/1>

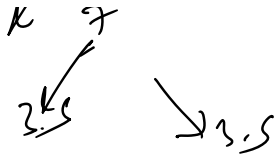
1792

20
612





$P_1 == P_2$



Perfect Sum Problem

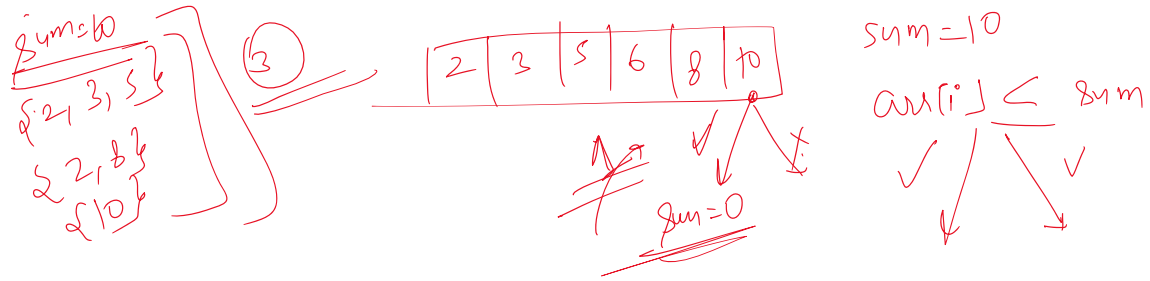
Medium Accuracy: **20.58%** Submissions: **111K+** Points: **4**

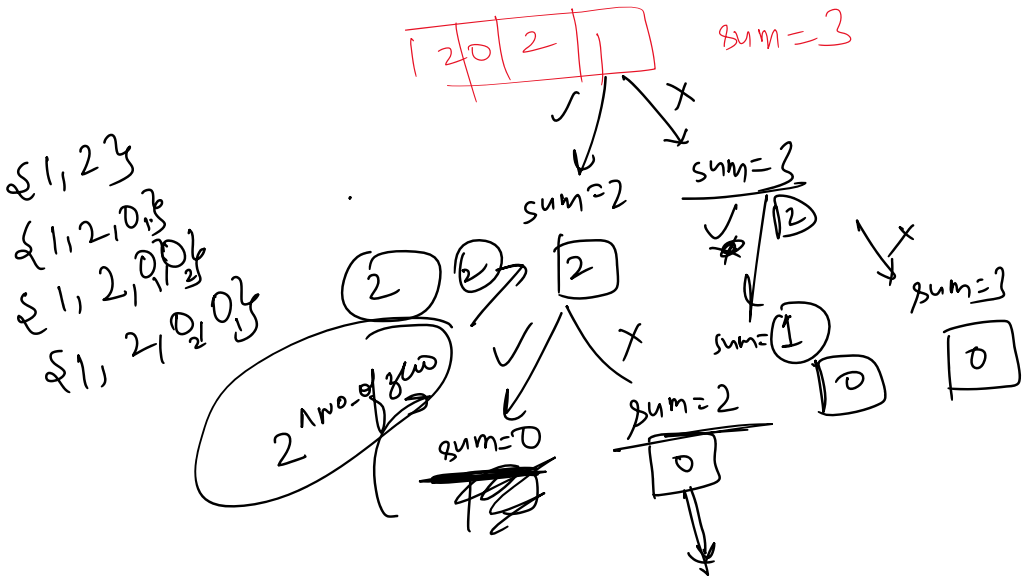
Given an array **arr[]** of non-negative integers and an integer **sum**, the task is to count all subsets of the given array with a sum equal to a given **sum**.

Note: Answer can be very large, so, output answer modulo 10^9+7

Example 1:

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





```
int mod = int(pow(10,9)) + 7;
int solve(int arr[], int n, int sum, int nz){
    // base case
    if(sum == 0) return (1 << nz);
    if(n == 0) return 0;
    if(arr[n-1] == 0){
        return solve(arr,n-1,sum,nz)%mod;
    }
    if(arr[n-1] <= sum){
      //  include                              not include
      return (solve(arr,n-1,sum-arr[n-1],nz) + solve(arr,n-1,sum,nz))%mod;
    }else{

        return solve(arr,n-1,sum,nz)%mod;
    }
}
int perfectSum(int arr[], int n, int sum)
{
    int no_of_zeros = 0;
    for(int i = 0;i< n;i++){
        if(arr[i] == 0){
            no_of_zeros++;
        }
    }
    return solve(arr,n,sum,no_of_zeros);
}
```

N = max(length of array, sum)

T.C - O(2^N)

S.C - O(N)

```
int perfectSum(int arr[], int n, int sum)
{
    int no_of_zeros = 0;
    for(int i = 0;i< n;i++){
        if(arr[i] == 0){
            no_of_zeros++;
        }
    }
    vector<int> curr(sum+1,0);
    curr[0] = (1 << no_of_zeros);

    for(int i = 1;i < n+1;i++){
        auto prev = curr;
        for(int j = 1;j < sum+1;j++){
            if(arr[i-1] == 0){
                curr[j] = prev[j]%mod;

            }
            else if(arr[i-1] <= j){
              //  include                              not include
                curr[j] = (prev[j-arr[i-1]] + prev[j])%mod;
            }else{
                curr[j] = prev[j]%mod;
            }
        }
    }
    return curr[sum]%mod;
}
```

T.C - O(N x sum)

S.C - O(2 x sum)