Theory

Count the Number of subset with given Sum

I/p : arr[] = [2, 3, 5, 6, 8, 10]
sum = 10

o/p : 3

< 2, 3, 5 >
< 10 >
< 2, 8 >

] 3 ways possible

Previously, we have seen that, is it possible to find a subset with given sum

↳ output

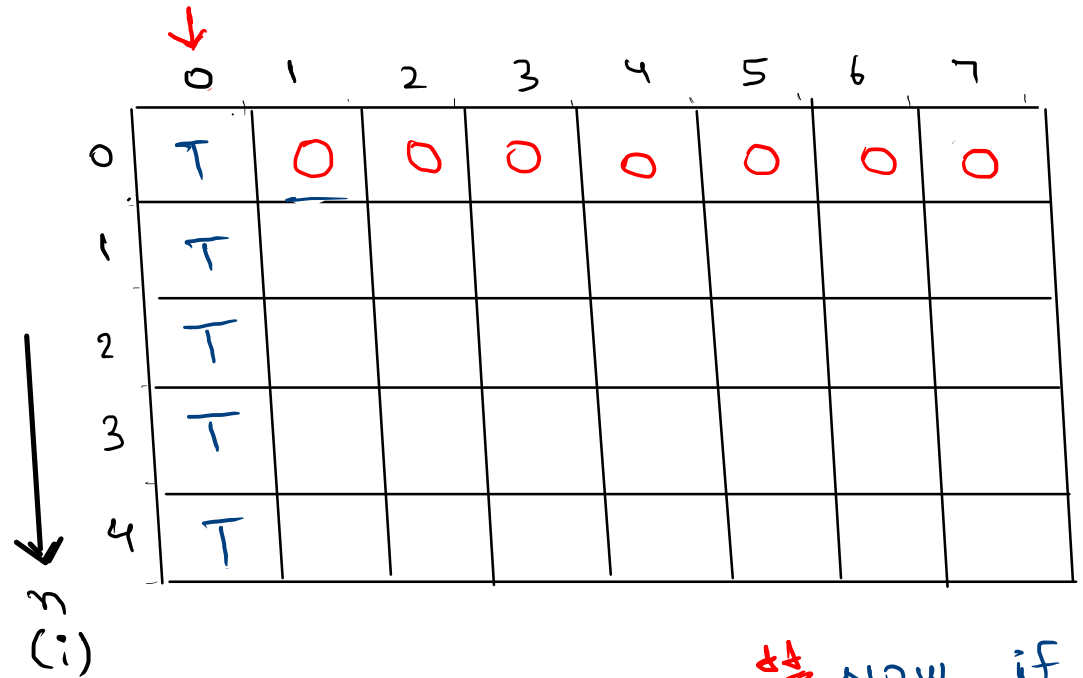
Count

False → 0

True → >0

How, we have created a 2D Vector dp[n+1][sum+1] →

2D Vector → int → Count



which represent whether is it possible to Create a subset with given sum !

False → 0

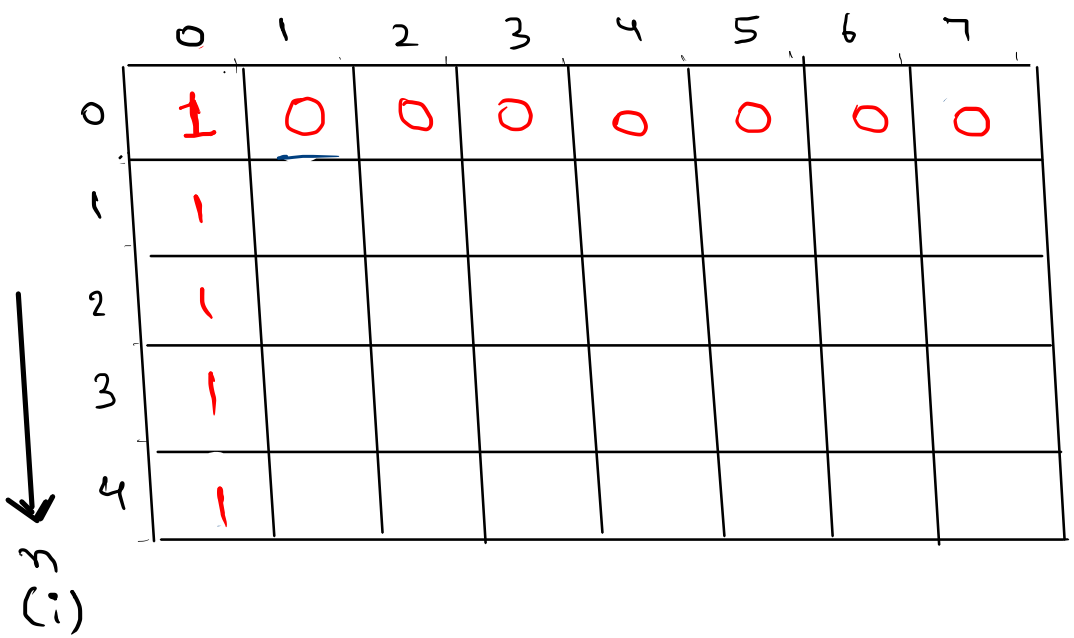
Now, if (sum == 0)

arr[] = {1, 3, 5, 6} →

No. of ways to get Sum=0

{ } → 1

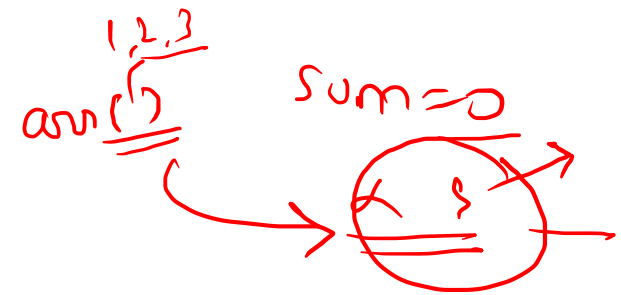
How, we have created a 2D Vector dp[n+1][sum+1]



which represent whether is it possible to Create a subset with given sum !

False → 0

(True → 1)



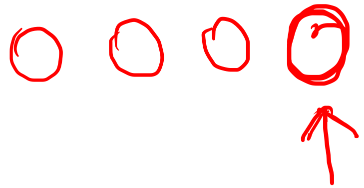
{ Recursive solution }

Logic

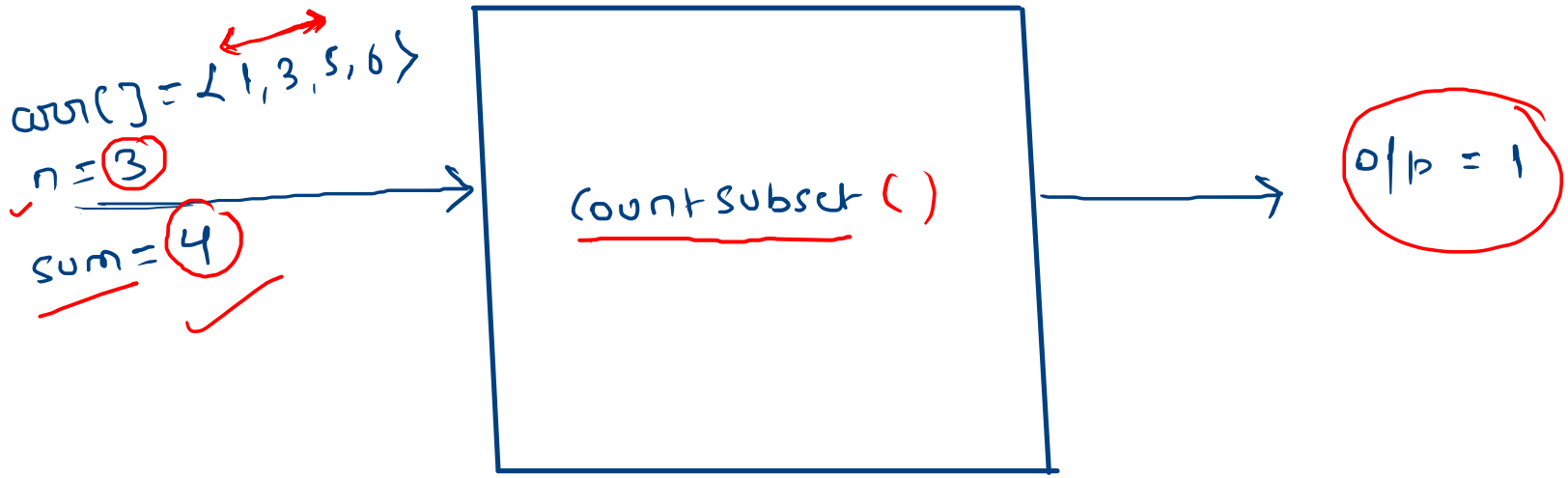
PREVIOUS QUESTION CODE LOGIC

Base condition

```
bool SubsetSum (vector<int> arr, int n, int sum)
{
    if (n==0) return false
    if (sum==0) return true;
    if (arr[n-1] <= sum)
    { return
            SubsetSum(arr, n-1, sum - arr[n-1]) ||
            SubsetSum (arr, n-1, sum);
    }
    else
        return SubsetSum (arr, n-1, sum);
}
```



logic



give count of subset which having given sum

Logic

I/p

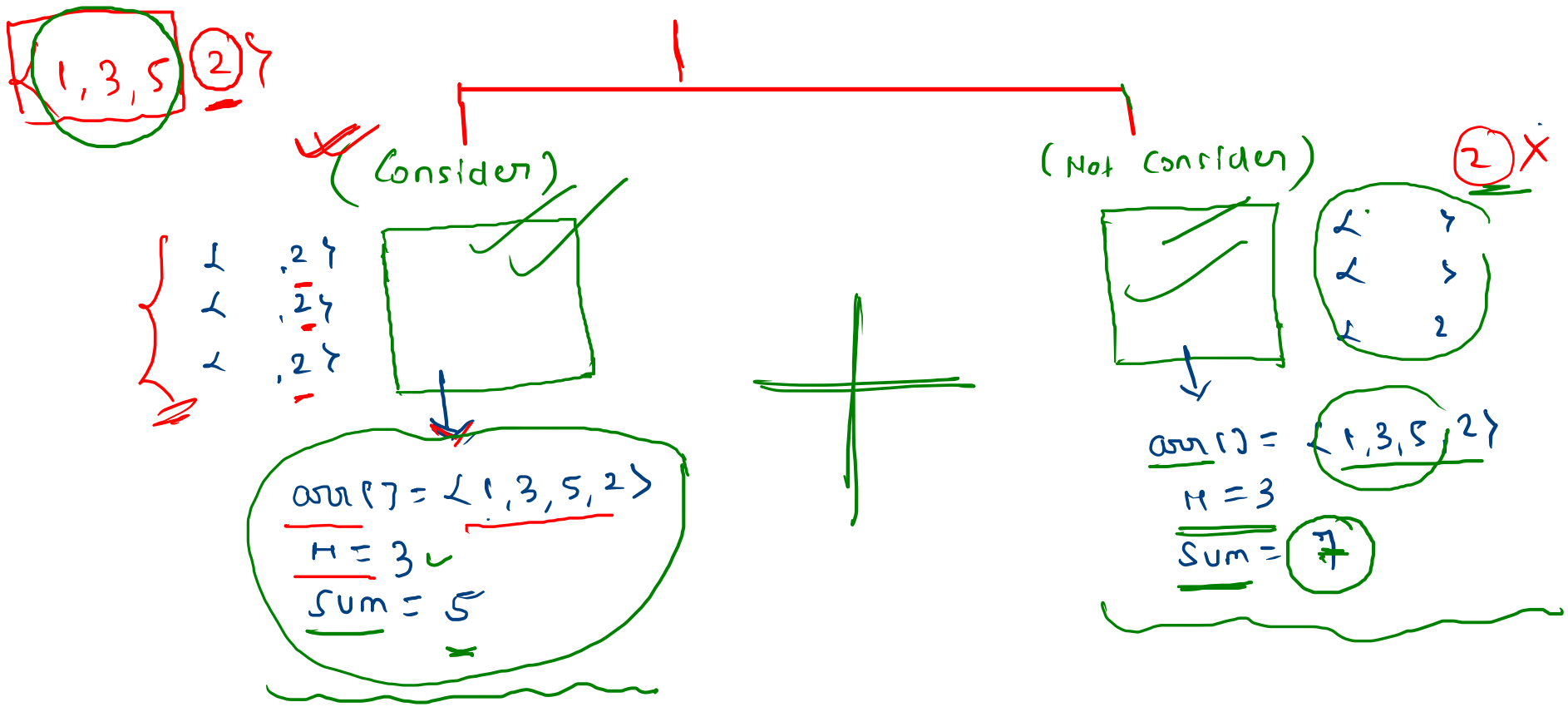
arr[] = {1, 3, 5, 2}

n = 4

sum = 7

we have 2 choices

consider this or not ?



{ 2} = 7

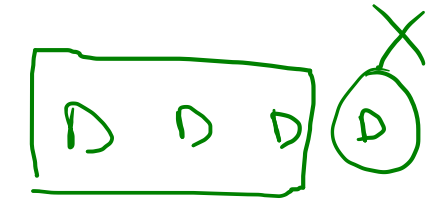
{ } = 7

{ } = 7

Recursive solution

Logic

```
bool CountSubset( vector<int> arr, int n, int sum)
{
    Base condition [ if (n==0) return 0;
                     if (sum==0) return 1;
    if (arr[n-1] <= sum)
    {
        return CountSubset(arr, n-1, sum-arr[n-1]) +
               CountSubset(arr, n-1, sum);
    }
    else
        CountSubset(arr, n-1, sum);
}
```



(Recursive code) ⟶ (DP code)

↳ Memoization

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

n, sum

DP solution

```
bool CountSubset( vector<int> arr, int n, int sum)
{
    if( dp[n][sum] != -1) return dp[n][sum];

    Base condition [ if ( n == 0 ) return 0;
                    if ( sum == 0 ) return 1; ]

    if ( arr[n-1] <= sum)
    {
      return dp[n][sum]= CountSubset ( arr, n-1, sum-arr[n-1]) +
                         CountSubset ( arr, n-1, sum);
    }
    else
    return dp[n][sum] = CountSubset ( arr, n-1, sum);
}
```