Date : / /
Page:

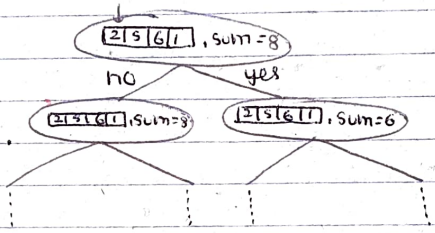
Day - 88

Recursion - 11 (Perfect Sum Problem)

2	5	6	1

, sum = 8

=) Find out the no. of subsets that sum is equal to required sum.



Code

=) int subset (int arr[], int index, int sum, int n){

```
if (sum == 0)
    return 1;
if (sum < 0 || index == n)
    return 0;
return subset(arr, index+1, sum, n) +
       subset(arr, index+1, sum + arr[index], n);
}
```

=) This logic will not work for negative numbers.

=) So, to handle -ve, we will traverse all the cases then we will check to include or to not include.

```
if (index == N)
    return sum == 0;
```

T.C → $O(2^n)$

Date: / /
Page:

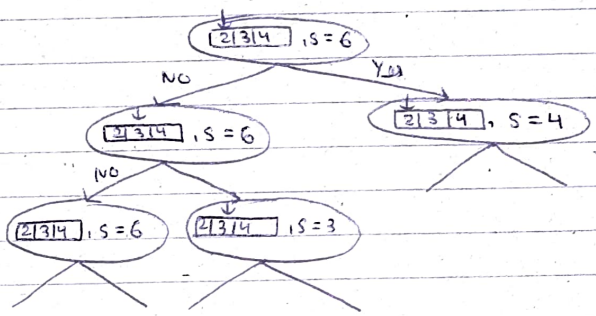
* Target Sum Repeatition:

2	3	4

Sum = 6

⇒ {2, 3} → 5 ✗
{2, 4} → 6 ✓
{2, 2, 3} → 7 ✗
{2, 2, 2} → 6 ✓
{3, 3} ✓

⇒



⇒ If we use any number then in the next call use it again, don't go to the next index.

⇒ code

```
int subsum(int arr[], int index, int n, int sum){
    if(sum == 0)
        return 1;
    if(index == N || sum<0)
        return 0;
    return subsum(arr, index+1, n, sum) +
    subsum(arr, index, n, sum+arr[index]);
}
```