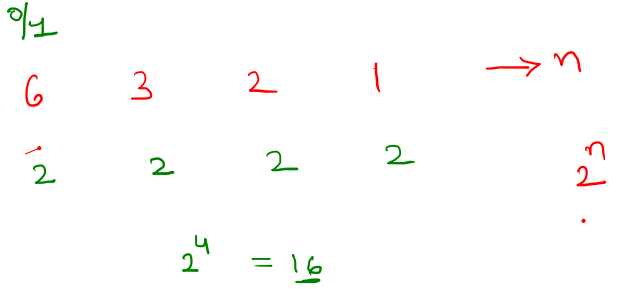
DP-3

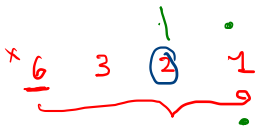
1) Subset Sum Problem

s={6,2,3,1} sum=5

i/p:-
- arr[]
- n
- sum

o/p:-
- yes/no





sum = 5 3

⇒ true

{ 2,

optimal ss:

ss(i , sum) = { true ; sum == 0 → { }
false ; i == 0
ss(i-1, sum) ; a_i > sum
ss(i-1, sum − a_i) || ss(i-1, sum) ; o/w

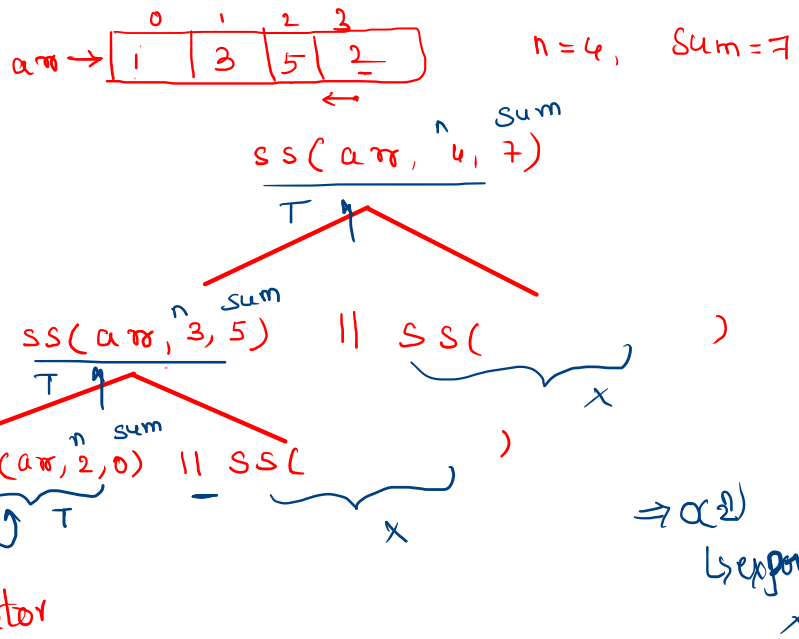
target sum

# of elements in set/arr

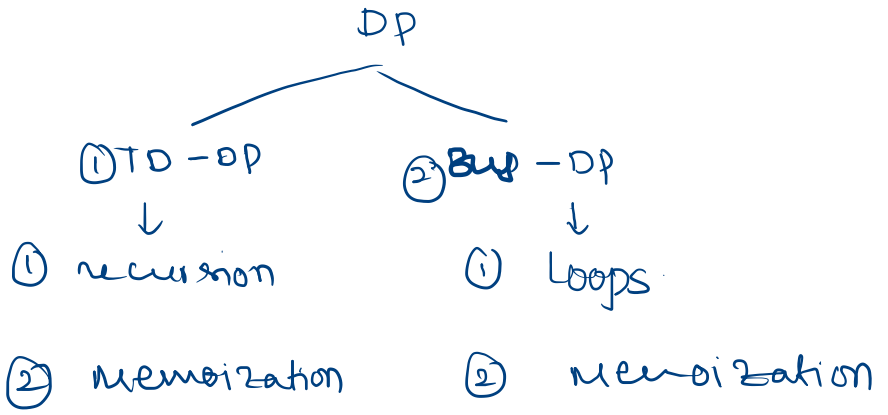
```
// SubSet sum problem
1) Recursive code

function ss(arr[],n,sum)
{
    ✓ if(sum==0)
        return true // empty set
      if(n==0)
        return false
      if(arr[n-1]>sum)
        return ss(arr,n-1,sum)
    → else
        return ss(arr,n-1,sum-arr[n-1]) || ss(arr,n-1,sum)
}
```

↳ logical OR operator



A	B	A || B
F	F	F
F	T	T
T	F	T
T	T	T


DP
(1) TD - DP
(2) BU - DP
(1) recursion
(2) Memoization
(1) Loops
(2) Memoization

dp [n+1] [sum+1]

i/p arr[]

n → # of elements (ICS)

sum → weight

2) TD-DP code for Subset Sum problem

```
dp[n+1][sum+1]={-1}//fill all the values with -1
function ss(arr[],n,sum)
{
      if(sum==0)
         return true // empty set
      if(n==0)
         return false
      if(dp[n][sum]!=-1)
           return dp[n][sum]
      if(arr[n-1]>sum)
         return dp[n][sum]=ss(arr,n-1,sum)
      else
         return dp[n][sum]=ss(arr,n-1,sum-arr[n-1]) || ss(arr,n-1,sum)
}
```

BUP - DP for SS - problem

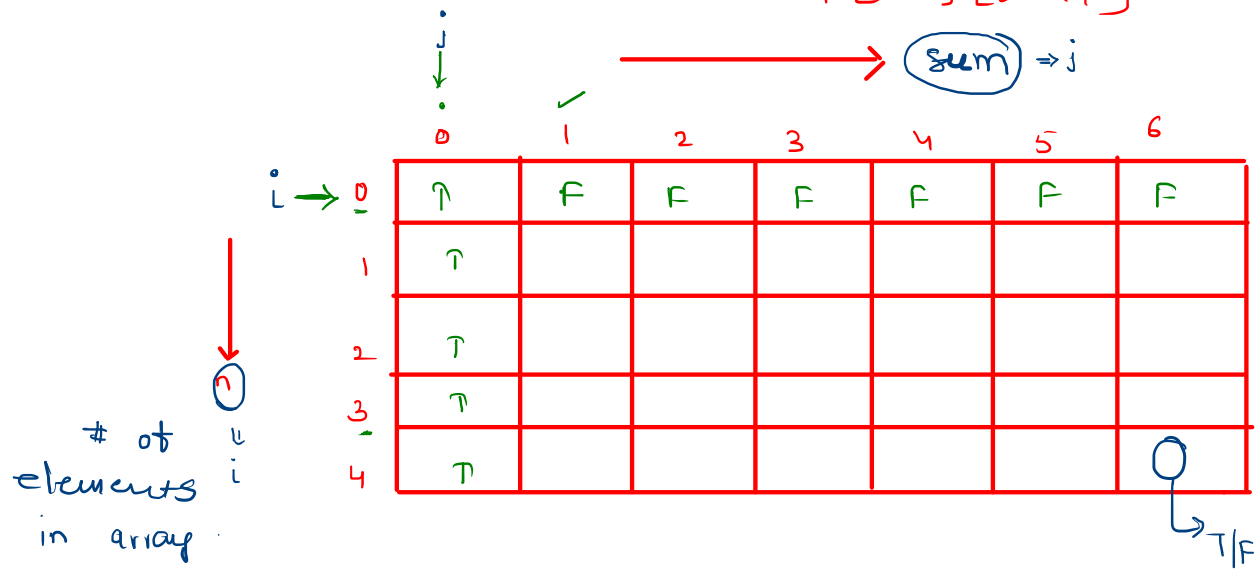
arr

0	1	2	3
5	3	1	2

, sum = 6

n = 4

dp [n+1] [sum+1] 5 X 7



True/false

3) BUP-DP code for Subset Sum problem

```
function ss(arr[],n,sum)
{
      dp[n+1][sum+1];
      for(i=0;i<=sum;i++)//first row
         dp[0][i]=false

       for(i=0;i<=n;i++)//first col
         dp[i][0]=true

       for(i=1;i<=n;i++)
       {
            for(j=1;j<=sum;j++)
            {
                if(arr[i-1]>j)
                    dp[i][j]=dp[i-1][j]
                else
                    dp[i][j]=dp[i-1][j-arr[i-1]]|| dp[i-1][j]
            }
        }
        return dp[n][sum]
}
```

n → i

sum → j

S = { 2, 3, 1, 5 } sum = 5

✓ 2) Total number of subsets with the given sum

{2, 3} ⇒ 2+3=5 ✓

{5} ⇒ ② op

	0	1	2	3	4	5	6
0	T 1	F 0	F 0	F 0	F 0	F 0	F 0
1	T 1						
2	T 1						
3	T 1						
4	T 1						

5 X 7

final Count

4) BUP-DP code for number of subsets whose sum is given sum

```
function ss(arr[],n,sum)
{
      dp[n+1][sum+1];
      for(i=0;i<=sum;i++)//first row
         dp[0][i]=0

       for(i=0;i<=n;i++)//first col
         dp[i][0]=1

       for(i=1;i<=n;i++)
       {
            for(j=1;j<=sum;j++)
            {
                if(arr[i-1]>j)
                    dp[i][j]=dp[i-1][j]
                else
                    dp[i][j]=dp[i-1][j-arr[i-1]]+ dp[i-1][j]
            }
        }
        return dp[n][sum]
}
```

n → i

sum → j

# of subsets

3) Un Bounded KS

Rod-cutting

```
Input : W = 8
        val[] = {10, 40, 50, 70}
        wt[]  = {1, 3, 4, 5}
Output : 110
We get maximum value with one unit of
weight 5 and one unit of weight 3.
```

Bounded KS → 0/1

UB KS

	O_1	O_2	O_3	O_4
P_i	5	7	9	10
W_i	2	1	3	4

C = 7

22 = 10 + 5 + 7

UB → tq

```
function ks(p[],wt[],w,n)
{
      dp[n+1][w+1];

      for(i=0;i<=w;i++)// 1st row all zeros
         dp[0][i]=0
      for(i=0;i<=n;i++)
         dp[i][0]=0

      for(i=1;i<=n;i++)
      {
           for(j=1;j<=w;j++)
           {
               if(wt[i-1]>j)
                   dp[i][j]=dp[i-1][j]
               else
                  dp[i][j]=max( p[i-1]+ dp[i][w-wt[i-1]], dp[i-1][j] )
           }
      }

     return dp[n][w]
}
```