13-Sept-2022

DSA-501

LECTURE # 7

TOPIC: DP-3

Sprint Evaluation - 2

Subset Sum

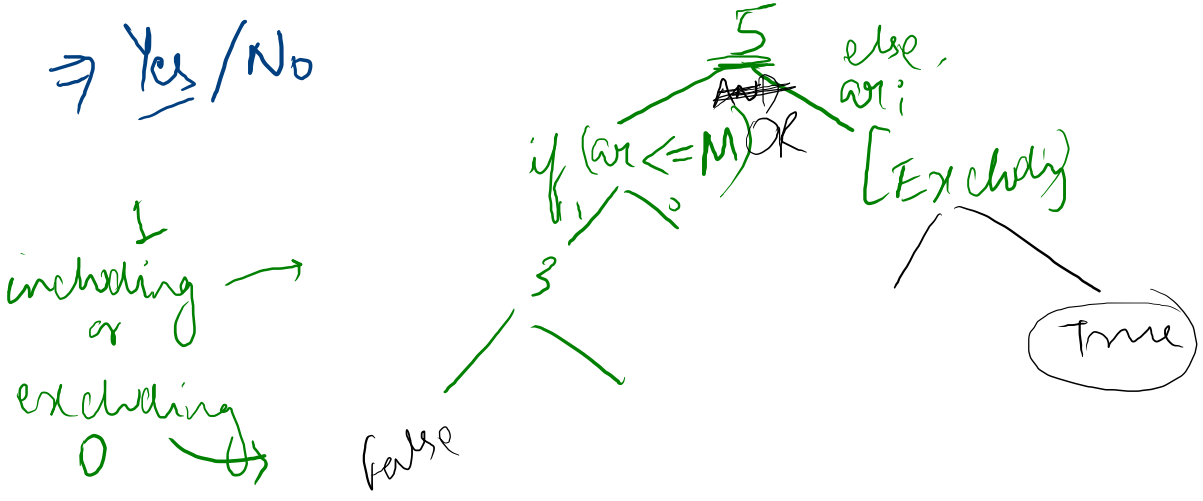
N=4

Arr = [5 3 12 1]

M = 4

Find a subset s.t
sum of its elements == M

⇒ Yes / No



DP → Recursive call

1. base conditions
2. recursive calls

```
{N      SubSumRec(N, M, arr)
{
    if (N == 0)
        ret FALSE
    if (M == 0)
        ret TRUE

    if(arr[N-1] <= M)                    → inc
ret     (SSRec(N-1, M-arr[N], arr)
        OR
         SSRec(N-1, M, arr))
                                         → excl

    else //excluding
        ret SSRec(N-1, M, arr)
}
```

Subset Sum

N = 4

$2 * 2 * 2 * 2 = 2^N$

Arr = [5 3 12 1]

M = 4

```
SubSumRec(N, M, arr)
{
    if (N == 0)
        ret FALSE
    if (M == 0)
        ret TRUE
    if (arr[N-1] <= M)
        ret (SSRec(N-1, M - arr[N], arr)   // inc
             OR
             SSRec(N-1, M, arr))           // excl
    else // excluding
        ret SSRec(N-1, M, arr)
}
```

TC = 2^N

TOP-DOWN Approach

I) create DS → no. of vars ? 2 (N, M)

→ ② dp[N+1][M+1] = -1 // all False

II) fill the data structure

```
function SubsetSumTD(N, M, arr)
{
        if( M ==0) return TRUE

        if(N==0) return FALSE

        if( dp[N][M] != -1) return dp[N][M]

        //recursive calls
        if( arr[N-1] <= M)
        {
                //incl OR excl
                return dp[N][M] = ( SubsetSumTD(N-1, M-arr[N-1], arr)
                                    OR
                                    SubsetSumTD(N-1, M, arr) )
        }
        else
                return dp[N][M] = SubsetSumTD(N-1, M, arr)

}
```

SC = O(N*M)

TC = O(N+M)

BOTTOM - UP

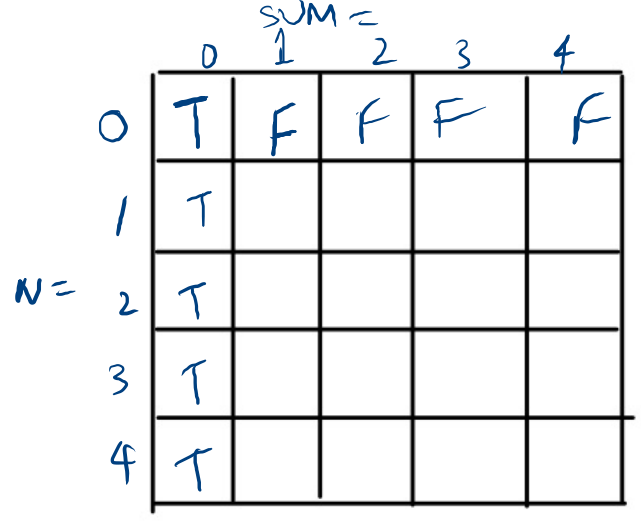
dp[5][5]

I] create data struct ⇒ dp[N+1][M+1]

Arr = [5 3 12 1]

M = 4

II) Fill the data st iteratively




```
function SubsetSumBottomUp(N, Sum, arr)
{
        dp[N+1][Sum+1]
        //initialization
        for(i=0; i<=N; i++) dp[i][0 ]= True

        for(i=1;i<=Sum; i++) dp[0][i ]=True

        //fillng the array
        for( i=1; i<=N; i++){
          for(j=1;j<=Sum; j++)
          {
                if( arr[i-1 ] <= j)
                {
                  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]
}
```

Edit Distance →

autocomplete

str1 = abcde) convert
str2 = abcg

opern's

1. insertn a char
2. deletn " "
3. replace a char

(I)
abcde
2 delete
abc
+1 addn → 3
abcg

(II) abcde
1 delete
abcd
1 replace
abcg

Min edit dist = 2