DP 2 : Two Dimentional

Question

Scenario

In the **Scaler educational program**, students are encouraged to balance between attending structured lectures and dedicating time to self-study, rest, or personal projects. Scaler understands the importance of self-directed learning and well-being, hence introduces a **flexible attendance policy**.

According to this policy :

- A student is required not to miss lectures on **two consecutive days**, ensuring a balanced engagement with the curriculum while also providing students time with their own learning.

Given the number of hours allocated for classes each day, the task is to determine the **MAX NUMBER OF HOURS** a student can allocate to self-study, rest, or personal projects by being absent, without violating the policy of missing two consecutive lectures.

SIMPLIFIED

Find maximum Subsequence sum for a given array, where selecting adjacent element is not allowed.

arr[] → 2 (0), -4 (1), 5 (2), 3 (3), -8 (4), 1 (5)

This Ex is for sub-sequence understanding

Ex 1 :- 9, 14, 13

↳ Ans => 22

Ex 2 :- 9, 4, 13, 24

↳ Ans -> 33

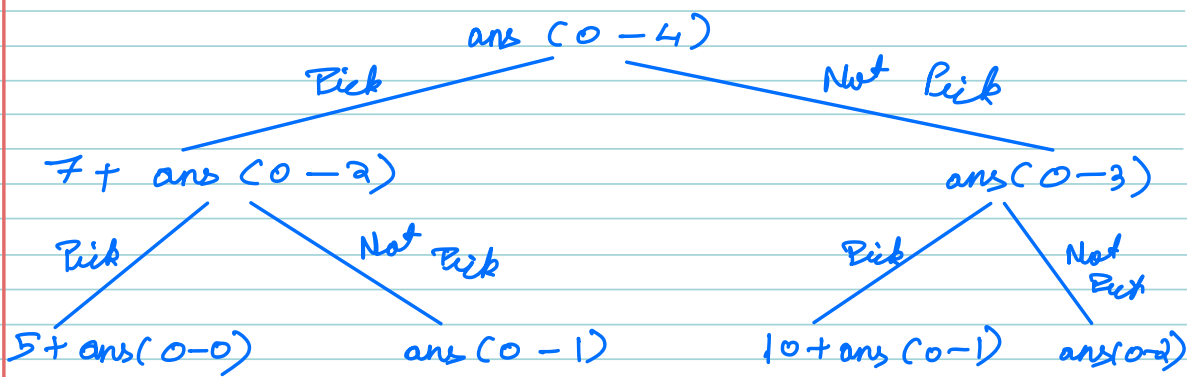
Quiz 1 :- Find maximum Subsequence sum where selecting adjacent element is not allowed.

[10, 20, 30, 40]

↳ Ans => 60

IDEA :- Pick & Not Pick

arr[] → 3 (0), 2 (1), 5 (2), 10 (3), 7 (4)



Q Here we will find the Final ans?

↳ max (Left, Right)

PSEUDO CODE

This can be skipped.

```
int MaxSum ( arr, s, e) {
    if ( e == 0) { return arr[e]; }
    if ( e < 0) { return 0; }
    include = arr[e] + MaxSum (arr, s, e-2);
    exclude = MaxSum ( arr, s, e-1);
    return Max ( include , exclude);
```

Q Can we optimize using DP ?

1.> optimal substructure

2.> overlapping sub-problem.

↳ yes

Q what Dimention of DP array is required ?

Pro tip :- No. of Variables in recursion.

↳ For us only 1 variable is changing Hence 1D array is required.

Suggestion :- Try to use minimum argument in the function

PSEUDO CODE [MEMOIZATION]

```
int dP[N] = {-1};
int MaxSum( arr, s, e, dP) {
    if (e==0) { return arr[e]; }
    if (e<0) { return 0; }
    if ( dP[e] != -1) { return dP[e]; }
    include = arr[e] + MaxSum(arr, s, e-2, dP);
    exclude = MaxSum( arr, s, e-1, dP);
    dP[e] = Max( include, exclude);
    return dP[e];
```

}

TC - O(N*1) ≈ O(N)
SC - O(N)

Q why we use Recursion → memoization → TABULATION Path ?

↳ Almost All the things required for Tabular Code is computed.

Eg:- Dimention of the dp array → dp[n]

Q what we need to store in dp[i] ?

↳ dp[i] : max Sub Sequence Sum from (0-i) by not picking adjacent elements

TABLE

	0	1	2	3	4
arr →	3	2	5	10	7
dp →	3	3	8	13	15
elements contributing to sum →	3	3	5, 3	10, 3	7, 5, 3

Q DP Expression ?

↳ dp[i] = max(arr[i] + dp[i-2] , dp[i-1])

PSEUDO CODE [TABULAR]

```
dp[0] = arr[0];
dp[1] = max( arr[0], arr[1] );
for ( i = 2; i < n; i++ ) {
    dp[i] = max( arr[i] + dp[i-2]
                 , dp[i-1] )
}
return dp[n-1];
```

TC — O(N)
SC — O(N)

Q Is Space optimization Possible.

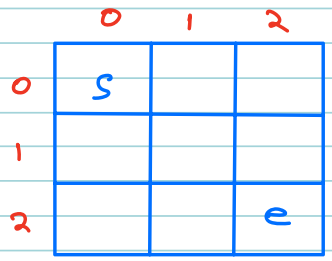
↳ yes

↳ (H.W.)

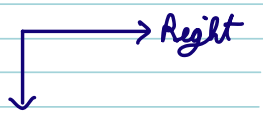
CONCLUSION :- Correct Way to solve DP Problem.

Recursion → memoization → Tabulation

Question:- Given Mat[n][m], find total No. of ways to go from (0,0) to (n-1, m-1).



Allowed Steps

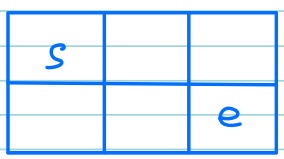


Possible Steps :-

RRDD
DDRR
RDRD
RDDR
DRRD
DRDR

6 ways

Quiz 2 :- Find total No of ways to go from (0,0) to (1,2)



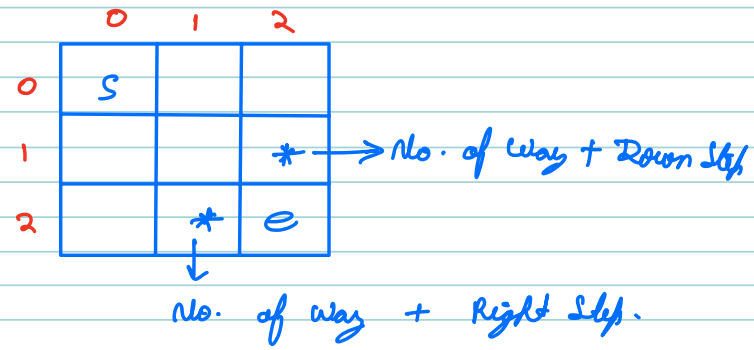
RRD
DRR
RDR

3 ways

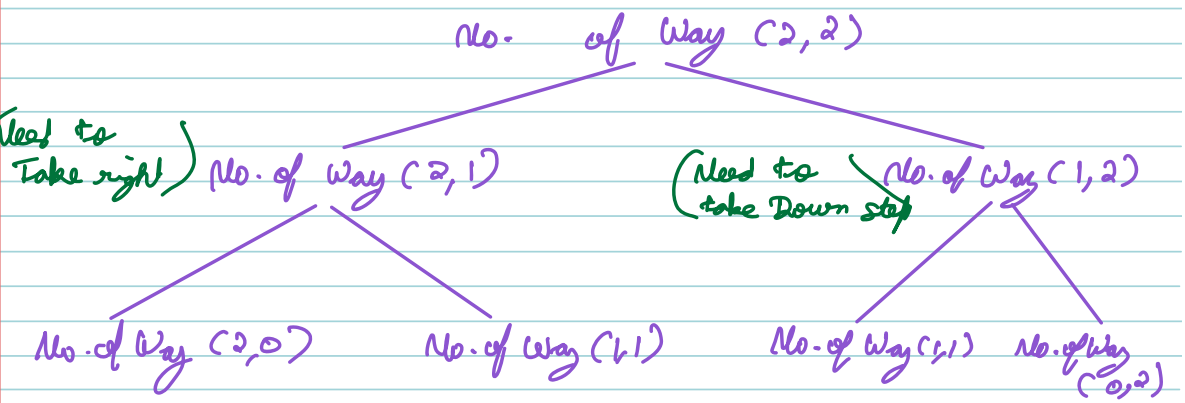
Q This question is similar to which question?

↳ STAIR – CASE

IDEA



OR



PSEUDO CODE

```
int Way (i, j) {
if (i<0 || j<0) { return 0; }
if (i==0 && j==0) { return 1; }
return ways(i-1, j) + ways(i, j-1);
}
```

ways(i-1, j) — Left

ways(i, j-1) — up

↳ OPTIMIZATION

Q Can we use DP ?

↳ yes

Q How many variables are changing ?

↳ 2D , DP array is needed.

PSEUDO CODE [MEMOIZATION]

```
int dp[N][M] = {-1};
int Ways(i, j, dp) {
    if (i<0 || j<0) { return 0; }
    if (i==0 && j==0) { return 1; }
    if (dp[i][j] != -1) { return dp[i][j]; }
    dp[i][j] = Ways(i-1, j, dp) + Ways(i, j-1, dp);
    return dp[i][j];
}
```

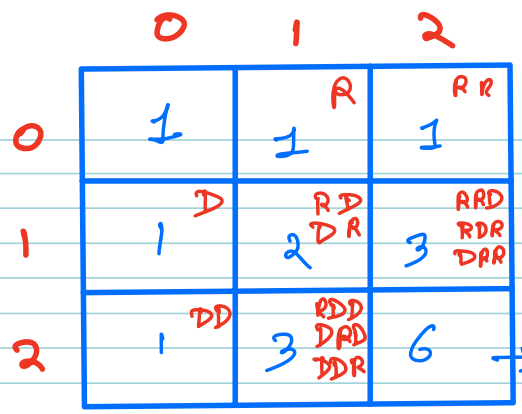
TC - O(N*M)

SC - O(N*M)

=> TABULATION

Q what we will be Storing at dp[i][j] ?

↳ dp[i][j] = Total No. of Way from (0,0) to (i,j)



Q DP Expression?

↳ $dp[i][j] = dp[i-1][j] + dp[i][j-1]$

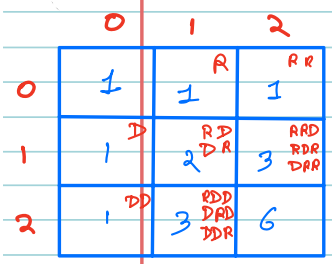
PSEUDO CODE [→ TABULAR]

```
for (i=0; i<n; i++){
    for (j=0; j<m; j++){
        if (i==0 || j==0){
            dp[i][j] = 1;
        } else {
            dp[i][j] = dp[i-1][j] + dp[i][j-1];
        }
    }
}
return dp[n-1][m-1];
```

TC — O(N*m)
SC — O(N*m)

Q Can we optimize over space?

↳ yes.



DP → | 1 | 1 | 1 | → Row 0

Reuse → This is an Ans for (0,1)

| 1 | ~~1~~ | ~~1~~ | → Row 1

2 3

New Value (1,1) = Current Value (0,1) + Left Value (1,0)

DRY RUN

DP → | 1 | ~~1~~ 2 | ~~1~~ 3 | → ~~Row 0~~ ~~Row 1~~ Row 2

3 6

Hence, Revamped DP Expression:

$$dp[j] = dp[j] + dp[j-1]$$

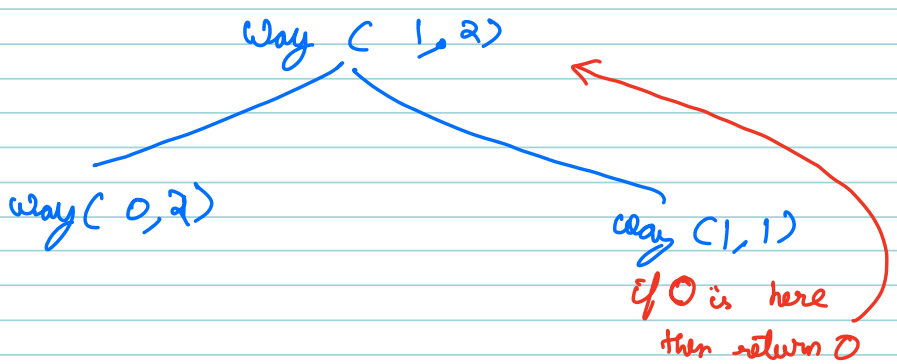
PSEUDO CODE

```
dp[m] = {-1};
for (i=0; i<n; i++){
    for (j=0; j<m; j++){
        if (i==0 || j==0){
            dp[j] = 1;
        } else {
            dp[j] = dp[j] + dp[j-1];
        }
    }
}
```

```
    }
  }
}
return dp[n-1];
```

Question:- Given mat[n][m], find total No. of ways from (0,0) to (n-1, m-1). Here 1 represents non blocked cell & 0 represents blocked cell.

	0	1	2	3
0	1	1	1	1
1	1	0	1	0
2	0	1	1	1
3	1	0	1	1
4	1	1	1	1

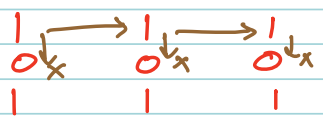


PSEUDO CODE

```
int Ways (i, j) {
    if (i < 0 || j < 0) { return 0; }
    if (i == 0 && j == 0) { return 1; }

    if (mat[i][j] != 0) {
        return Ways(i-1, j) + Ways(i, j-1);
    } else {
        return 0;
    }
}
```

Quiz :- How Many Unique paths in grid from (0,0) to (2,2) ?



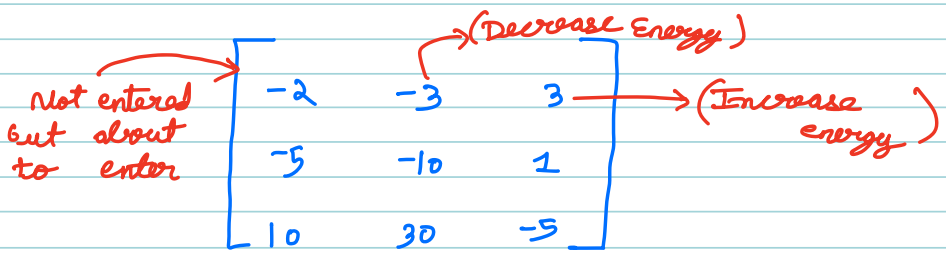
⇒ Ans = 0

Question [Leet Code Hard Problem]

Find the minimum health level of the prince to start with to save the princess, where the negative numbers denote a dragon and positive numbers denote red bull.

Redbull will increase the health whereas the dragons will decrease the health.

The prince can move either in horizontal right direction or vertical down direction.
If health level <= 0, it means prince is dead.



Allowed Dir
Right.
Down

Min Health

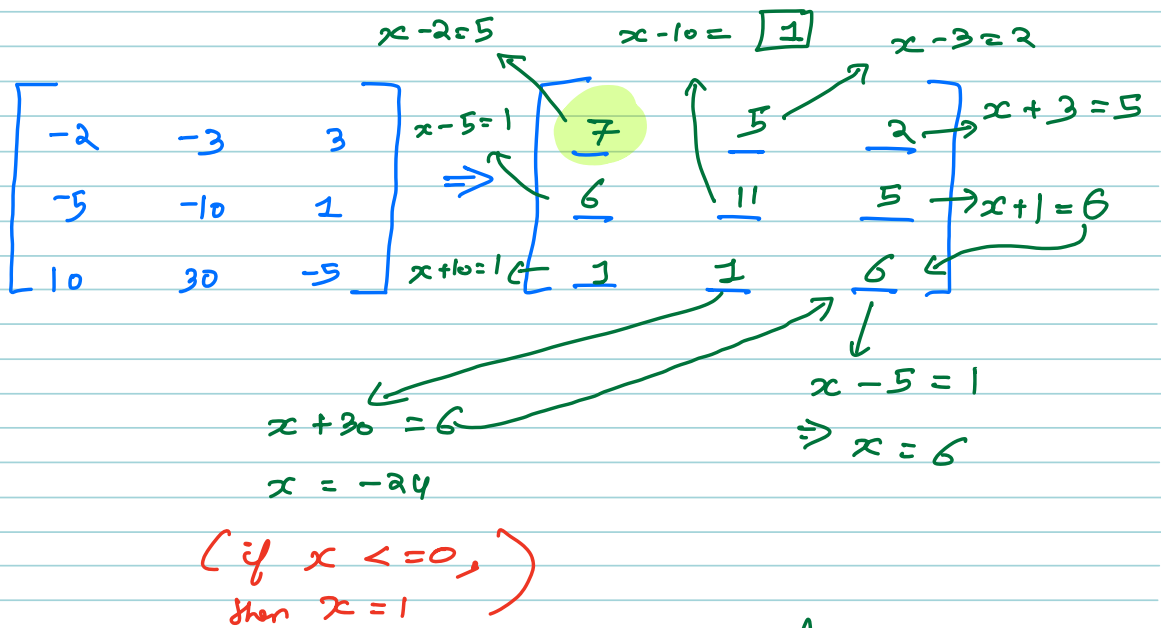
① E = ✗ ✗ 0 ✗

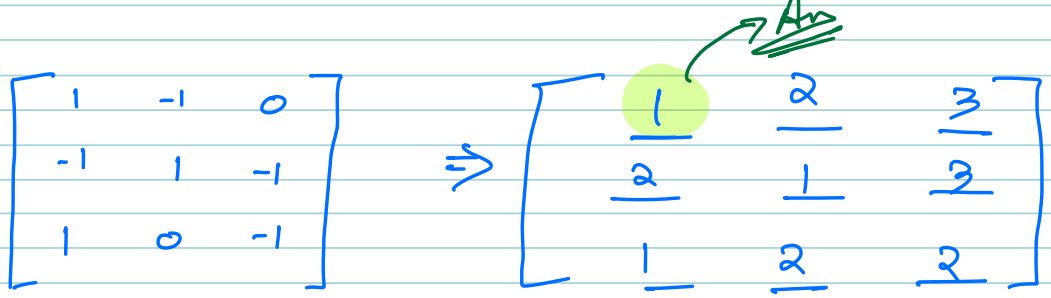
② E = ✗ ✗ ✗ ✗ ✗ 0 ✗

③ E = ✗ ✗ ✗ ✗ ✗ 1 ✓

Conclusion :- So the question is to start with least possible health but still able to reach princess.

Imp :- This is Not Min / Max path. Here we want to meet princess alive i.e. energy never goes 0 or negative.





DP Expression ?

$x + arr[i][j] = \min(dp[i+1][j], dp[i][j+1])$

$\Rightarrow dp[i][j] = \min(dp[i+1][j], dp[i][j+1]) - arr[i][j]$

if (x or $dp[i][j] <= 0$){

$dp[i][j] = 1;$

}

PSEUDO CODE

```
dp[N][M] = {-1};
if ( arr[N-1][M-1] > 0) {
        dp[N-1][M-1] = 1;
} else {
        dp[N-1][M-1] = 1 + abs(arr[N-1][M-1]);
}
for( i = N-2 ; i >= 0 ; i--){
    for (j = M-2 ; j >= 0 ; j--){
        x = max( 1, min(dp[i+1][j], dp[i][j+1])
                    - arr[i][j]);
        dp[i][j] = x;
    }
}
return dp[0][0];
```

TC → O(N * M)
SC → O(N * M)

CATALAN NUMBER

The Catalan numbers form a sequence of natural numbers that have numerous applications in combinatorial mathematics. Each number in the sequence is a solution to a variety of counting problems. The Nth Catalan number, denoted as Cn, can be used to determine:

- The number of correct combinations of N pairs of parentheses.
- The number of distinct binary search trees with N nodes, etc.

Here is the Sequence

$C_0 = 1$

$C_1 = 1$

$C_2 = C_0 * C_1 + C_1 * C_0 = 2$

$C_3 = C_0 * C_2 + C_1 * C_1 + C_2 * C_0 = 5$

$C_4 = C_0 * C_3 + C_1 * C_2 + C_2 * C_1 + C_3 * C_0 = 14$

Formula

$$C_N = C_0 * C_{N-1} + C_1 * C_{N-2} + C_2 * C_{N-3} \cdots + C_{N-1} * C_0$$

$$C_N = \sum_{i=0}^{N-1} C_i * C_{N-1-i}$$

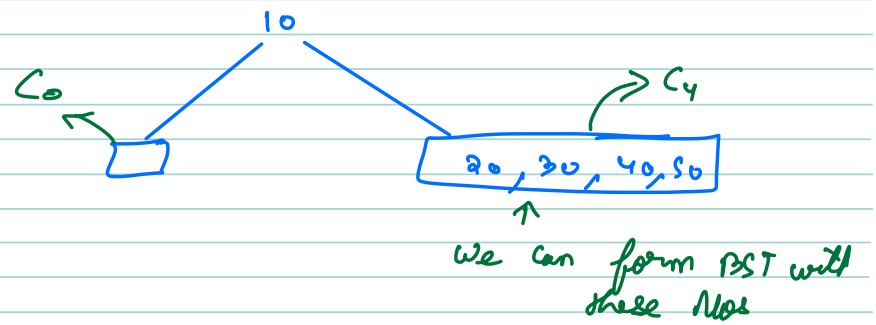
Question:- You are given a Number N, count Total no. of unique Binary Search Trees, that can be formed using N distinct numbers.

Ex N=1 [10], Ans 1 → C_1 10

Ex N=2 [10,20] | Ans 2 → C_2 10 20 and 20 10

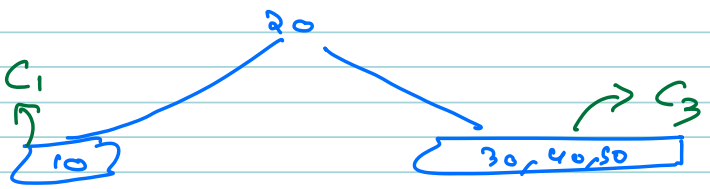
Ex N = 5 ⇒ [10, 20, 30, 40, 50]

⇒ 10 as root



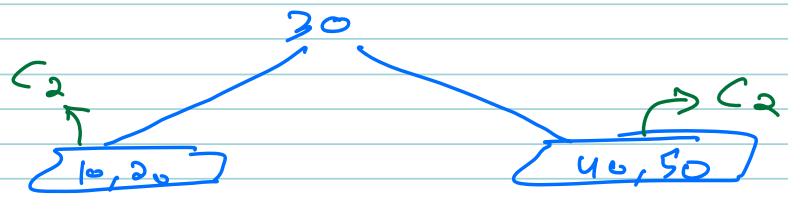
So, total No. of BST will 10 as root = $C_0 * C_4$

⇒ 20 as root



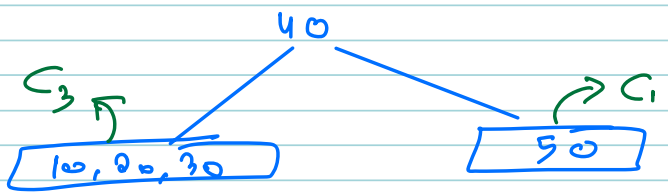
So, total No. of BST will 20 as root = $C_1 * C_3$

=> 30 as root



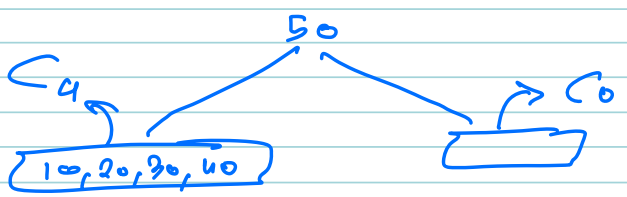
So, total No. of BST will 30 as root = $C_2 * C_2$

=> 40 as root



So, total No. of BST will 40 as root = $C_3 * C_1$

=> 50 as root



So, total No. of BST will 50 as root = $C_4 * C_0$

Total No. of Ways = $C_0 * C_4 + C_1 * C_3 + C_2 * C_2 + C_3 * C_1 + C_4 * C_0$

$[C_I]$

Quiz:- Count Total No. of Unique BST, that can be formed using 2 distinct No.

↳ $C_2 = C_0 * C_1 + C_1 * C_0$

$C_0 = 1$, $C_1 = 1$; $C_1 = 1$, $C_0 = 1$

⇒ 2

PSEUDO CODE [H.W.]

```
int C[N+1];
C[0] = 1;
C[1] = 1;
for ( i = 2; i <= N; i++){
    for (int j = 0; j < i ; j++){
        C[i] += C[j] * C[N-1-j];
    }
}
return C[N];
```