DP2 : 2 Dimentional

Question-1

Given an array where each element represents amount of money in a house. Find the maximum amount of money you can rob without triggering an alarm.

Alarm trigger when 2 adjacent houses get robbed ~~together~~.

$(i-1) \leftarrow i^{th} \rightarrow (i+1)$

$$A = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ \hline 9 & 4 & 13 & 24 \end{array}$$ ans = 33

$$A = \begin{array}{ccc} 0 & 1 & 2 \\ \hline 10 & 15 & 10 \end{array}$$ ans = 20

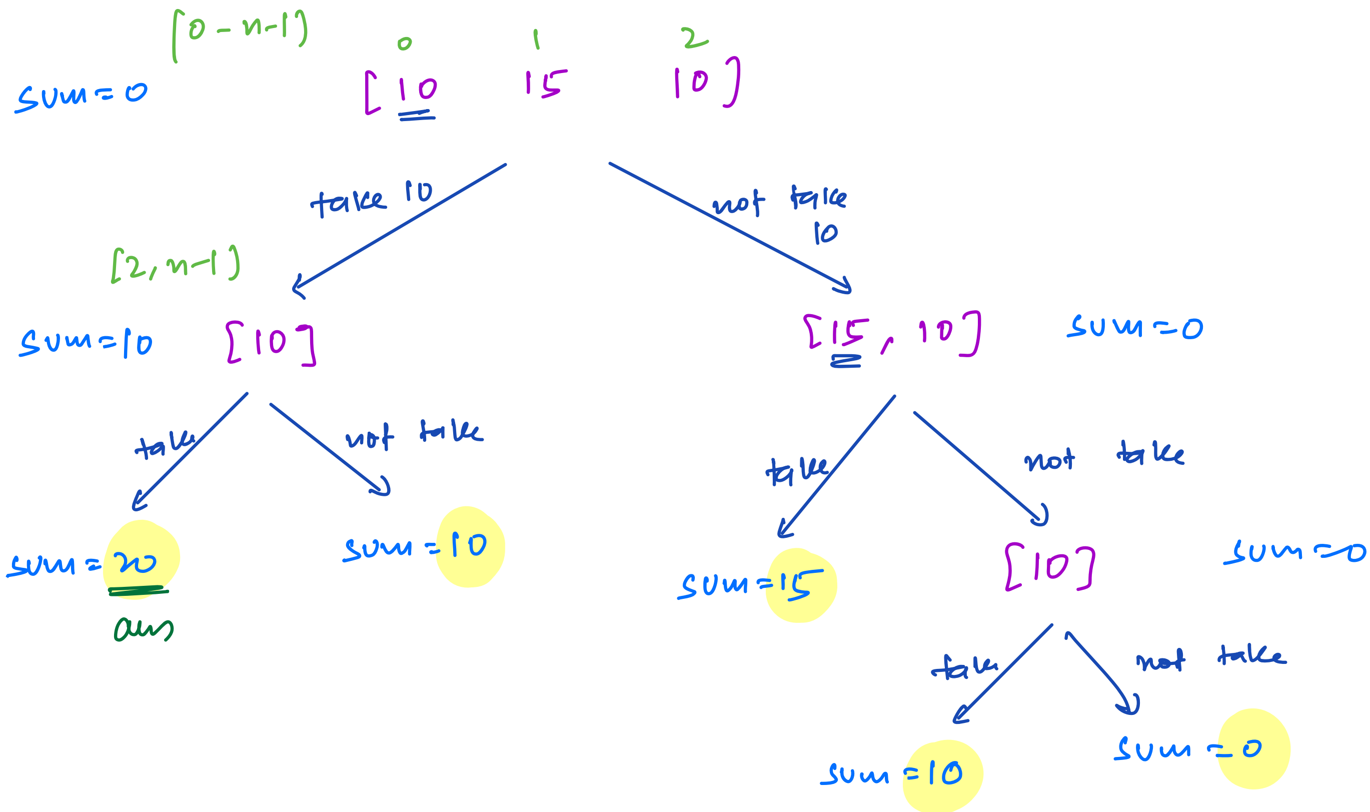
$$A = \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline 2 & 7 & 9 & 3 & 1 \end{array}$$ ans = 12

Bruteforce → Explore all subset of houses by either robbing or skipping each house.

for each house :

1. Rob the house & skip the next one.

2. Skip the house & move to next one.



Code

```
int maxSum ( A[], i) {
    if(i > n-1)      return 0;
    int s1 = A[i] + maxSum(A, i+2);
    int s2 = maxSum(A, i+1);
    return max(s1, s2);
}
```

$TC = O(2^N)$

$SC = O(N)$

optimal substructure ✓
overlapping subproblems ✓ } use DP

```
int dp[n];
∀i, dp[i] = -1;
int maxSum( A[], i ) {
    if(i > n-1)    return 0;
    if(dp[i] != -1)  return dp[i];
    int s1 = A[i] + maxSum(A, i+2);
    int s2 = maxSum(A, i+1);
    dp[i] = max(s1, s2);
    return dp[i];
}
```

TC = O(N)

SC = O(N)

for <u>iterative code</u>

smallest subproblem → dp[n-1] ⇒ [n-1, n-1)

→ dp[n-2] = [n-2, n-1]

$$dp[n-1] = \max(\overset{\text{rob}}{a[n-1]}, \overset{\text{not rob}}{0});$$

$$dp[n-2] = \max(\underset{\text{rob}}{a[n-2]}, \underset{\text{not rob}}{dp[n-1]});$$

```
for( i = n-3  to  0) {
    dp[i] = max( a[i] + dp[i+2] , dp[i+1]);
}
return dp[0];
```

TC = O(N)

SC = O(N) ⇒ O(1) ?

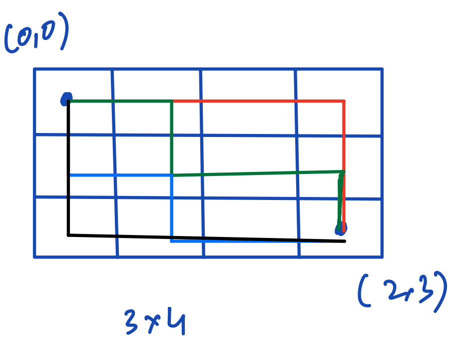
	0	1	2	3
A =	9	4	13	24
dp	33	28	24	24

(9+24, 28)   (4+24, 24)

Question 2

Given mat[n][m], find total no. of ways to go from (0,0) to (n-1, m-1).

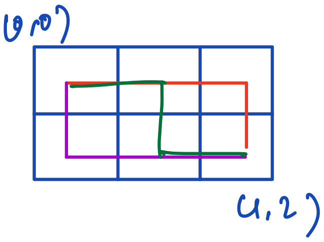
In 1 step, you can go down / right.



RRRDD   DDRRR

RDRRD   RDDRR

DRDRR   ⋮

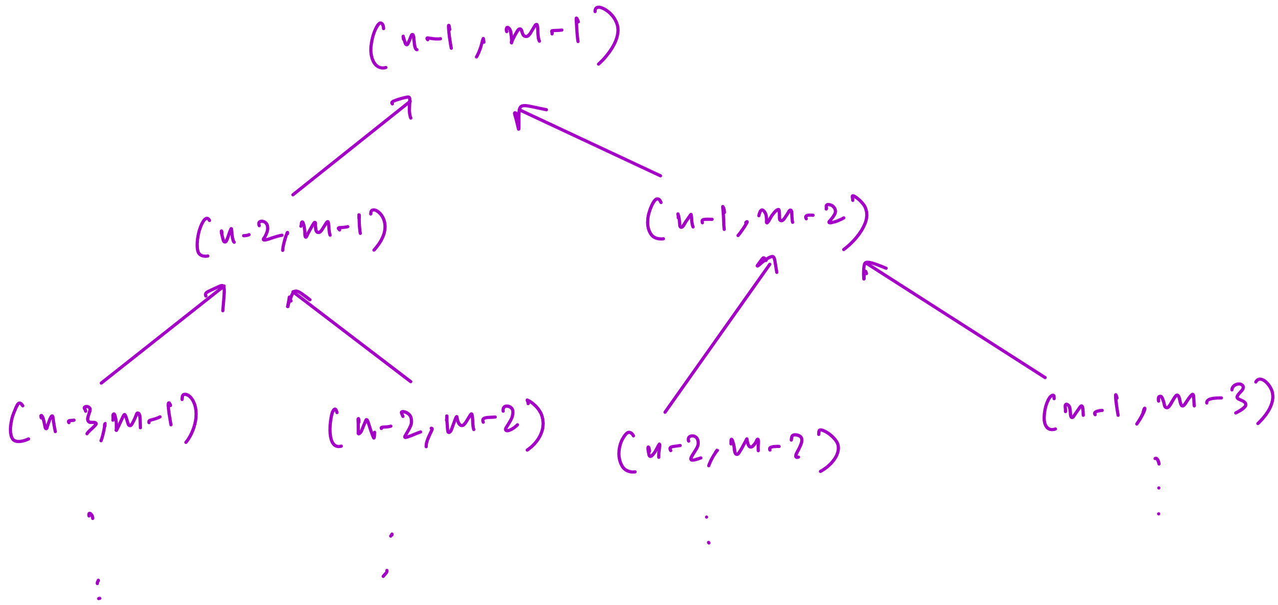


RRD RDR

DRR

ans = 3

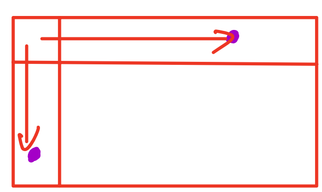
Brute force



$$ways(i,j) = ways(i-1,j) + ways(i,j-1)$$

Code

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



$TC = O(2^{N*M})$

$SC = O(N*M)$

}

when i=0, you can only go right

j=0, you can only go down

```
int dp[N][M];
∀ i,j   dp[i][j] = -1

int ways (i, j) {
    if (i==0 || j==0) {
        return 1;
    }
    if (dp[i][j] != -1)   return dp[i][j];

    dp[i][j] = ways(i-1, j) + ways(i, j-1);

    return dp[i][j];
}
```

$TC = O(N \times M)$

$SC = O(N \times M)$

Iterative dp

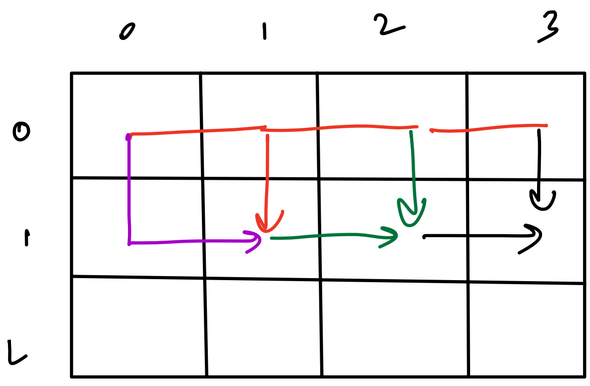
```
for (i=0 to n-1) {
    for (j=0 to m-1) {
        if (i==0 || j==0)   dp[i][j] = 1;
        else   dp[i][j] = dp[i-1][j] + dp[i][j-1];
```

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

TC = O(N*M)

SC = O(N*M)

	0	1	2	3
0	1	1	1	1
1	1	2	3	4
2	1	3	6	10



3x4

We can further optimize SC by using only 2 rows.

current row

previous row

$\Rightarrow SC = O(2M)$

$= O(M)$

Question 3

Find how many A-digit positive numbers exist, s.t. the sum of their digits = B.

A valid no. does not have leading zeros.

Since the answer can be large, return ans % (10^9+7).

$A=2$, $B=4$

ans = 4

$\Rightarrow$ 22, 31, 13, 40

$A=1$, $B=5$

ans = 1

$\Rightarrow$ 5

$A=1$, $B=10$

ans = 0

Brute force: Generate all possible A-digit number & check their sum.

$\Rightarrow 9^A$ or 10^A

Iterative dp

dp[i][j] → count of valid i-digit numbers whose digit sum = j.

dp[A][B] ← goal

$$\underbrace{\underbrace{[\cdot\ \cdot\ \cdots\ ]}_{i-1 \text{ digits}}\ d}_{i \text{ digits}} \quad d \to [0-9]$$

dp[i][j] += dp[i-1][j-d]

Base case :

dp[1][j] = 1   0 <= j <= 9

dp[1][j] = 0   j > 9

Code

int dp[A+1][B+1]

∀ i,j   dp[i][j] = 0

// Base case

for (d = 1 to 9) {

```
        if ( d <= B)
            dp[1][d] = 1
    }
    int MOD = 10^9 + 7;
    for ( i = 2  to  A ) {
        for ( j = 0  to  B ) {
            for ( d = 0  to  9 ) {
                if ( j >= d )
                    dp[i][j] = ( dp[i][j] + dp[i-1][j-d] ) %
                                                        MOD;
            }
        }
    }

    return  dp[A][B];
```

$TC = O(A \times B \times 10)$
$= O(A \times B)$

$SC = O(A \times B)$

$A = 2, \; B = 4$ $22, 31, 13, 40$

$dp[1][0] = 0$
$dp[1][1] = 1$
$dp[1][2] = 1$
$[3] = 1$
$[4] = 1$

$dp[2][1] = dp[1][1]$ $d = 0$
$+ \; dp[1][0]$ $d = 1$

$dp[2][1] = 1$ (10)

$dp[2][2] = dp[1][2]$ $\quad d = 0$

$+$

$dp[1][1]$ $\quad d = 1$

$+$

$dp[1][0]$ $\quad d = 2$

$= 2$ $\quad (20, 11)$

Code → Bruteforce

```
int  count( int i, int j) {
     if (i==1) {
         if(j==B)  return 1;
         return 0;
     }
     int ans=0;
     for(d = 0  to 9)  {
         if( j >=d ) {
             ans += count(i-1, j-d);
         }
     }
     return  ans;
}
```

Catalan Numbers

→ count of valid parantheses

N=2

()(), (())

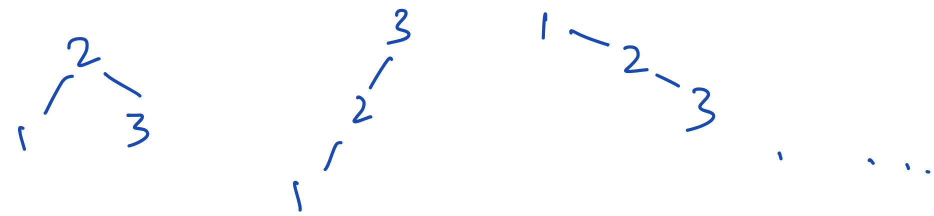
N=3

()()(), (())(), ((())),

()(())

→ no. of distinct BST with N nodes

N=3



Catalan no.

$C_0 = 1$

$C_1 = 1$

$C_2 = C_0 \times C_1 + C_1 \times C_0 = 2$

$C_3 = C_0 \times C_2 + C_1 \times C_1 + C_2 \times C_0 = 5$

$C_4 = C_0 \times C_3 + C_1 \times C_2 + C_2 \times C_1 + C_3 \times C_0 = 14$

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

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

code

```
int C[n+1];

C[0] = 1 , C[1] = 1;

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

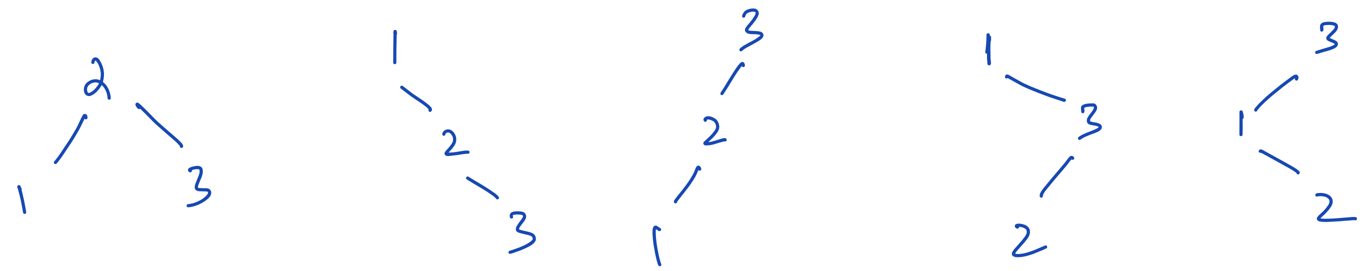
$TC = O(N^2)$

$SC = O(N)$

Question

Given a number N, count total unique BSTs that can be formed by N distinct no. .

$N = 3$ $\{1, 2, 3\}$



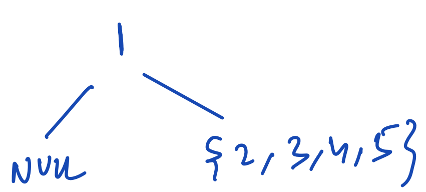
ans = 5

$N = 2$ $\{1, 2\}$

ans = 2

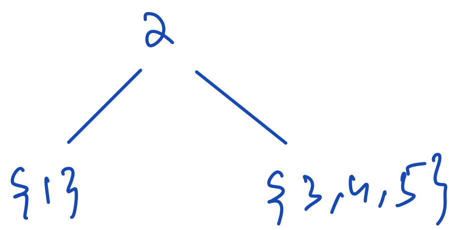
$N = 5$ $\{1, 2, 3, 4, 5\}$ $C_5 = ?$

1 as root



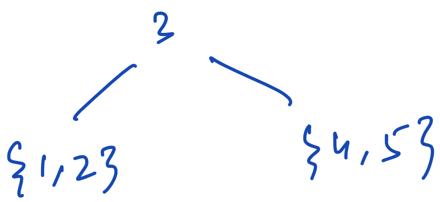
$\xrightarrow[\text{BSTs}]{\text{total}} C_0 \times C_4$

2 as root



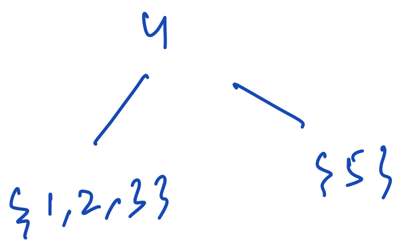
$\xrightarrow{\text{total BSTs}} C_1 \times C_3$

3 as root

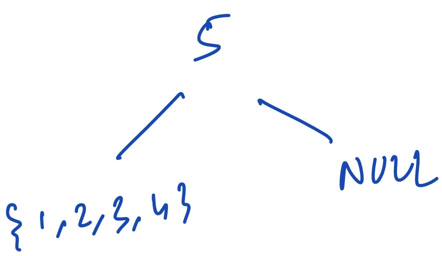


$\xrightarrow{\text{total BSTs}} C_2 \times C_2$

4 as root



$\longrightarrow C_3 \times C_1$



$\longrightarrow C_4 \times C_0$

$$C_5 = C_0 \times C_4 + C_1 \times C_3 + C_2 \times C_2 + C_3 \times C_1 + C_4 \times C_0$$

ans = $C[N]$

$TC = O(N^2)$

$SC = O(N)$

$C_i \Rightarrow$ total no. of unique BSTs with i nodes.