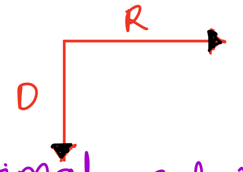
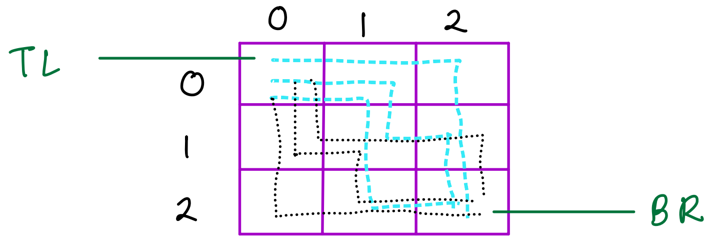
Q ▷ Find total no. of ways TL → BR

movements allowed





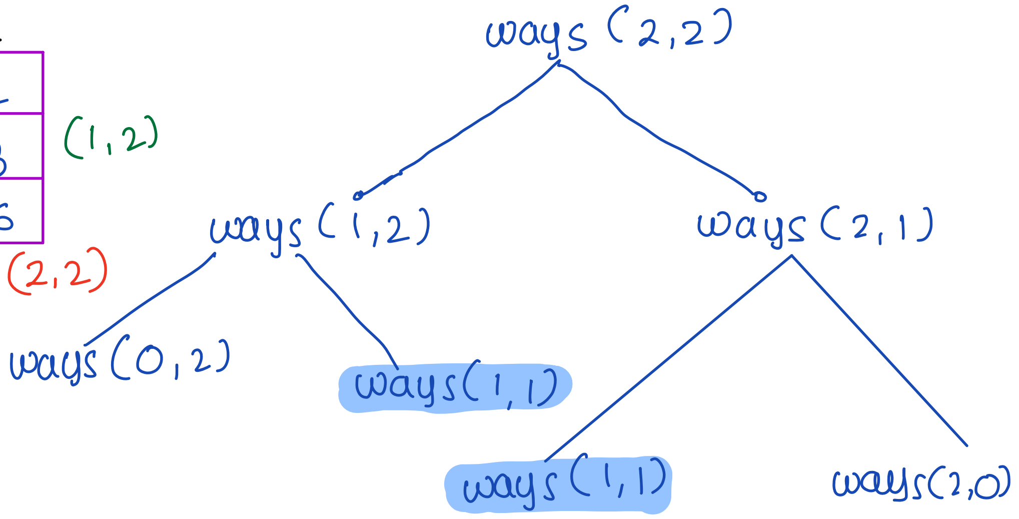
Optimal substructure

|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 1 | 2 | 3 |
| 2 | 1 | 3 | 6 |

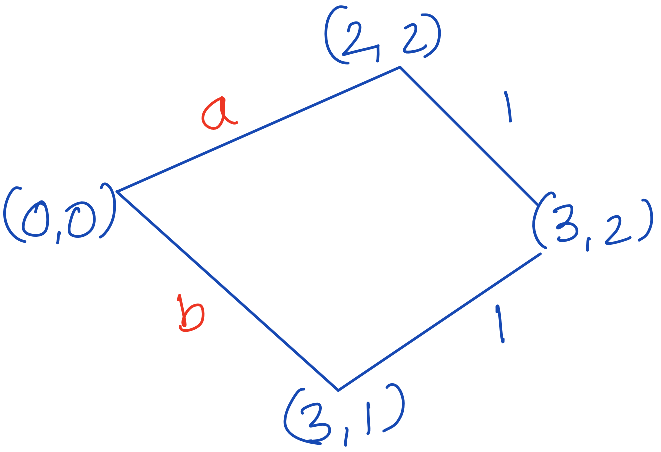
(1,2)

(2,1)

(2,2)

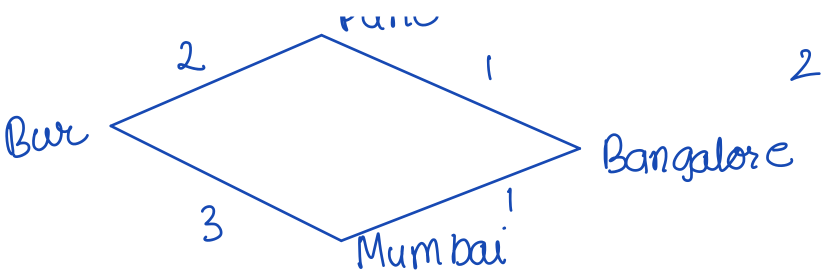


|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 1 | 2 | 3 |
| 2 | 1 | 3 | 6 |
| 3 | 1 | 4 | 10 |



$\boxed{\text{Total no. of ways to reach (3,2) = a+b}}$

Dune.



2X1 + 3X1

ways (i, j) = Total ways to reach (i,j)

DP table    NxM    {2D Matrix}

DP expresion / Recurrence Relation.

$$\boxed{ways(i, j) = ways(i-1, j) + ways(i, j-1)}$$

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

      int a = ways(i-1, j);
      int b = ways(i, j-1);
      dp[i][j] = a+b;
      return a+b;
}
```

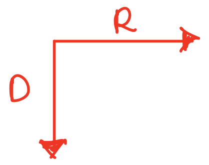
TC: O(NM)

SC: O(NM)

```
int[][] dp = new int[N][M];
    for i ->
          j ->
             dp[i][j] = -1
```

Q2> Find total no. of ways to reach BR from TL with obstacles

movement allowed



|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 |   |   |   |   |
| 1 |   | ▨ |   |   |
| 2 |   |   | ▨ |   |
| 3 |   |   |   |   |

M[i][j] = -1

obstacle

DP

|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 2 |
| 2 | 1 | 1 | 0 | 2 |
| 3 | 1 | 2 | 2 | 4 |

M = Matrix

```
int ways ( int i , int j ) {
        // Base condition
        if( i<0 || j<0 ) return 0;
        if( M[i][j] == -1 ) return 0;
        if( dp[i][j] != -1 ) return dp[i][j];

      int a = ways(i-1, j);
      int b = ways(i, j-1);
      dp[i][j] = a+b;
}     return a+b;
```

if M[0][0] != -1
dp[0][0] = 1
else
dp[0][0] = 0

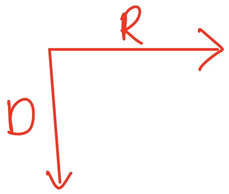
dp[0][0] = 0

|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | ~~0~~ | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 2 | ~~0~~ | 1 |
| 3 | 1 | 3 | 3 | 4 |

8 : 15 → 8 : 25

Break

Q3> Given a 2D matrix, filled with positive numbers find path from TL ⟶ BR to minimize the cost.



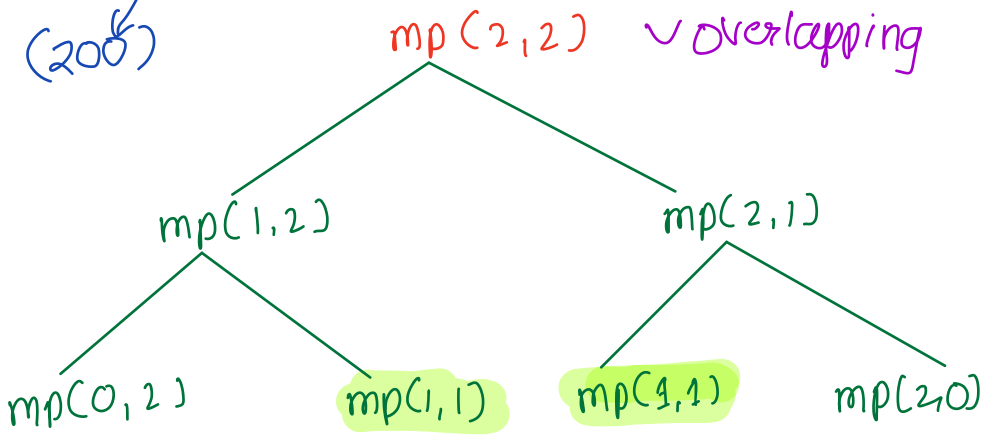
|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 2 | 3 | 5 |
| 1 | 1 | 6 | 4 |
| 2 | 9 | 2 | 7 |

(100)

(200)

✓ optimal substructure

✓ overlapping



|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 2 | 10 | 1 |
| 1 | 100 | 100 | 4 |
| 2 | 100 | 2 | 7 |

mp(i,j) = min cost path ending at (i,j)
reach (i,j)

DP table = 2D matrix

DP expression
recurrence

$$mp(i,j) = M[i][j] + \min \begin{cases} mp(i-1,j) \\ mp(i,j-1) \end{cases}$$

(1,2)
(i-1, j)
↑
(2,1) ⟵ (2,2)

M

DP

(i, j-1)

|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 2 | 3 | 5 |
| 1 | 1 | 6 | 4 |
| 2 | 9 | 2 | 7 |

|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 2 | 5 | 10 |
| 1 | 3 | 9 | 13 |
| 2 | 12 | 11 | 18 |

→ Init DP table

→ Take Prefix sum of first row &

→ Take Prefix sum of first col

```
for ( i = 1 ; i < N ; i++ ) {
        for ( j = 1 ; j < M ; j++ ) {
            dp[i][j] = min( dp[i-1][j] , dp[i][j-1])
                                + M[i][j] ;
        }
}

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

(House Robber)

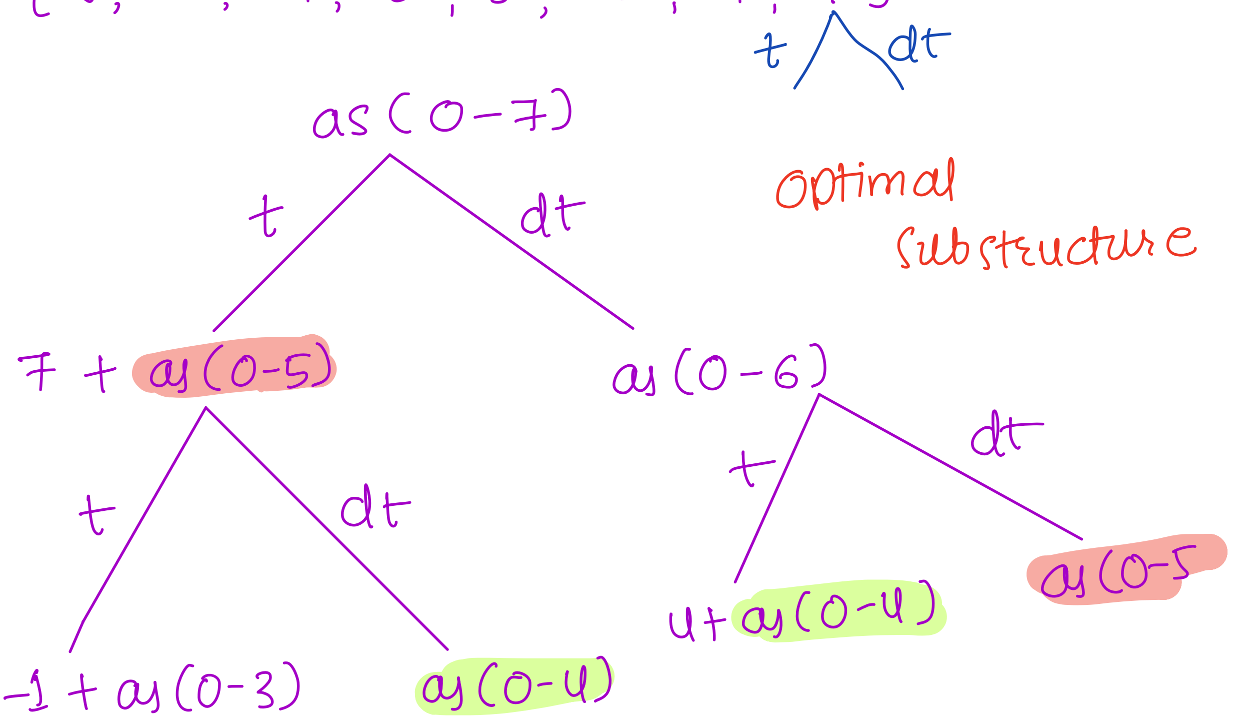
Qu> Given an array find max alternating sum.

cannot take consecutive elements.

{9 4 13 24}

{9+13 4+24} 9+24 = 33

$$\left\{\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2, & -1, & -4, & 5, & 3, & -1, & 4, & 7 \end{array}\right\}$$



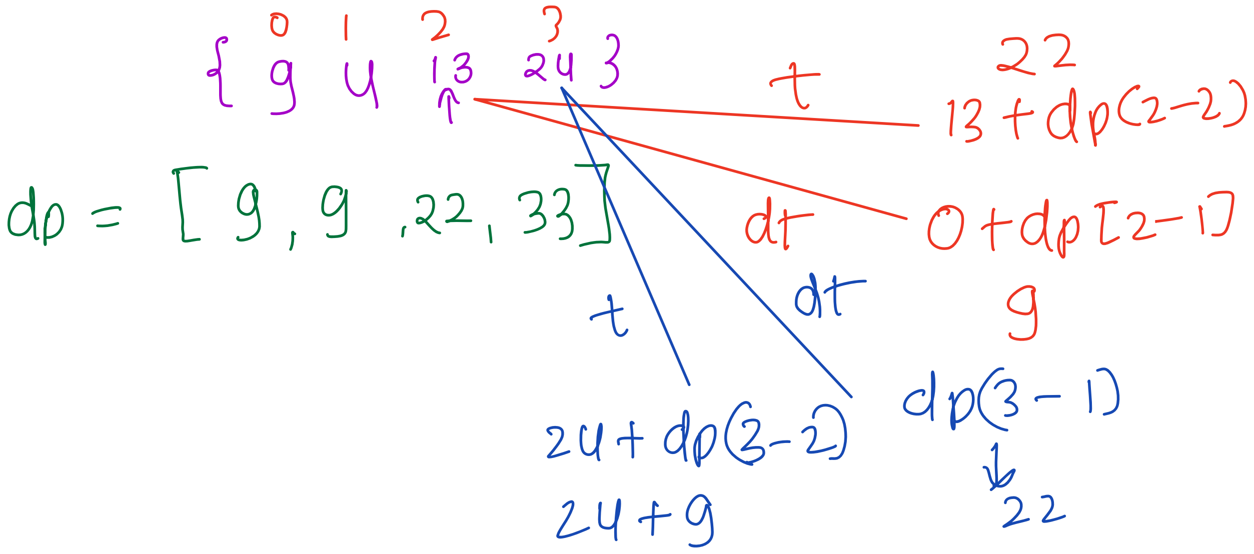
optimal substructure

as(i) = max sum from 0-i

DP table = 1D array

DP expression/ recurrence

$$as(i) = \max \begin{cases} t & A[i] + as(i-2) \\ dt & as(i-1) \end{cases}$$

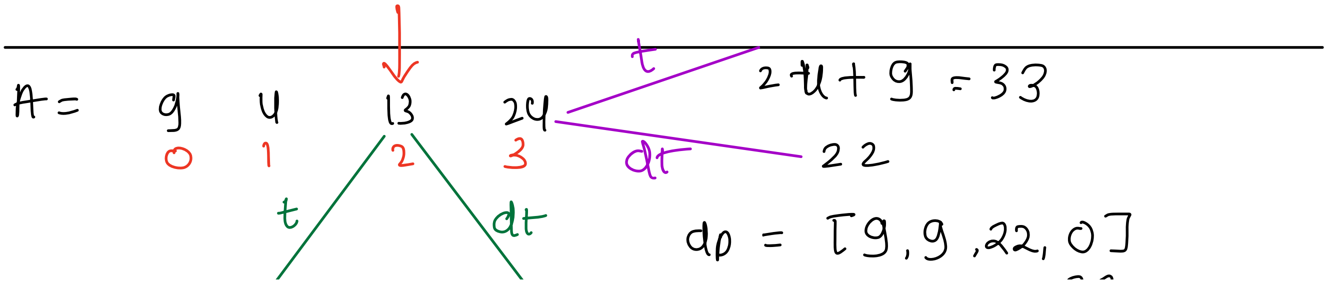


```
int dp = new int[n];
    dp[0] = A[0];
    dp[1] = max(A[0], A[1]);

    for (i = 2 ; i<n ; i++)
        int t = A[i] + dp[i-2];
        int dt = dp[i-1];
        dp[i] = max(t, dt);

    return dp[n-1];
```

TC : O(n)      SC : O(n)



13+9 9 33

$O(n)$

$$A_i + \left.\begin{matrix} \overset{t}{dp_{i-2}} \\ \underset{dt}{dp_{i-1}} \end{matrix}\right\} \max$$