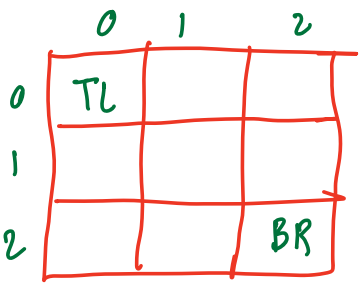
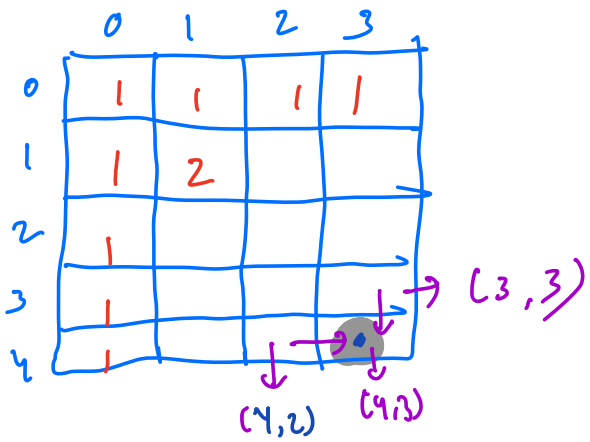
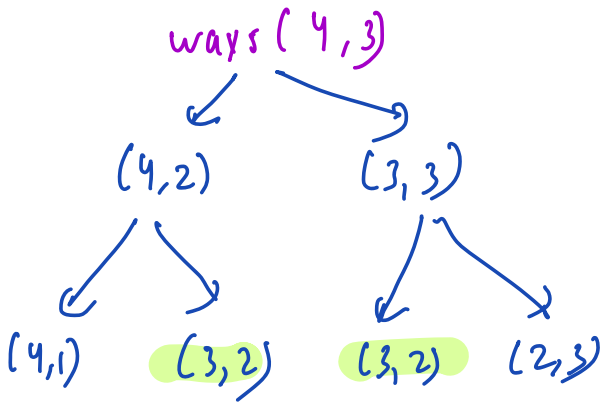
Q1⇒ No of ways to go from (0,0) → (BR cell)
TL BR



(0,0), (0,1), (0,2), (1,2), (2,2)
(0,0), (0,1), (1,1), (1,2), (2,2)
(0,0), (0,1), (1,1), (2,1), (2,2)
(0,0), (1,0), (1,1), (1,2), (2,2)
(0,0), (1,0), (1,1), (2,1), (2,2)
(0,0), (1,0), (2,0), (2,1), (2,2)



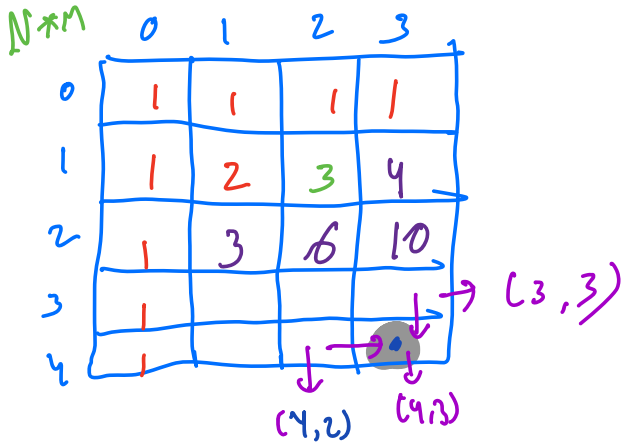


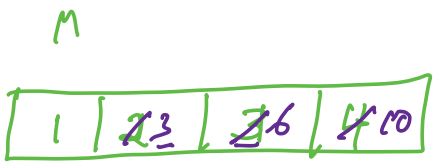
$$dp[i][j] = dp[i-1][j] + dp[i][j-1]$$

```
int ways (int N, int M)
{
    int dp[N][M];
    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: $O(N*M)$

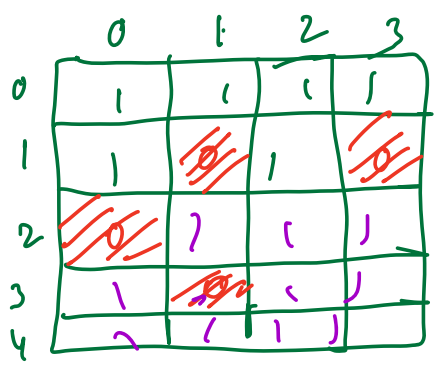
SC: $O(N*M)$

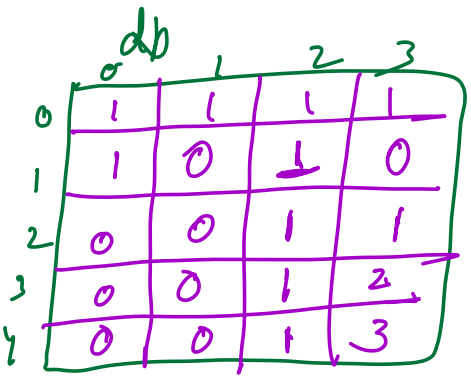




Q2 ⇒ Number of ways to go from (0,0) → (BR)

mat[i][j] = 0 → it means blocked





$$dp[i][j] = \begin{cases} \text{if}(mat[i][j] == 0) \ \{dp[i][j] = 0\} \\ \text{else} \ \{dp[i-1][j] + dp[i][j-1]\} \end{cases}$$

```
int ways (int mat[][], int N, int M)
{
    // col
```

```
for (i=0 ; i<n ; i++)
  {
    if( mat[i][0] == 1 )   dp[i][0]=1
    else
       break;
    }
// row
    j= 0; j<m ;j++
     if( mat[0][j] ==1)   dp[0][j]=1;
    else  break;
}
 for(i=1 ; i< N;i++)
  {
    for (j=1; j<m; j++)
     }
       if ( mat[i][j] ==0)
                dp[i][j]=0;
       else
         dp[i][j] = dp[i-1][j] +
                              dp[i][j-1]
       }
  }
  return  dp[n-1][m-1];
}
```

TC: O(N*M)

```
int ways( int mat[][], int N, int M)
{
    int dp[N][M] = {-1};
    if( mat[0][0] == 0 ) return 0;

    func ( mat, dp, N-1, M-1)
}
```

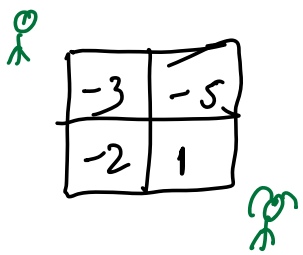
TODO

(0,0) -> 1

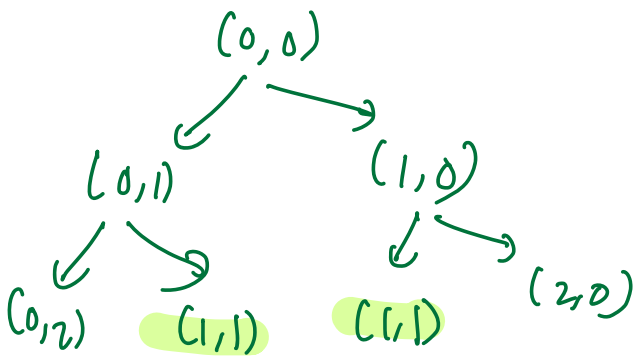
```
int func(int mat[][]; int dp[][], int i, int j)
{
    if ( i < 0 || j < 0) return 0;
    if (mat[i][j] == 0 ) return 0;

    if (dp[i][j] == -1)
    {

dp[i][j] = { func(mat, dp, i-1, j)
                 +
             func(mat, dp, i, j-1)

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

Q3 ⇒ Min Cost to reach (0, 0) → BR

// Given mat[N][M], where each cell indicates health gained, min. health required at 0,0 such that we can reach [N-1][M-1]



H = 4 (X) H = 5 (X) H = 6



Case - I

x → [-5]

x - 5 = 1

x = 6

Case 2

x → [-2] [-5] → 6

x + (-2) = 6

x = 8

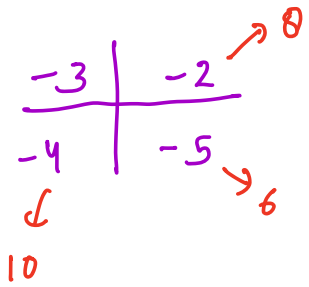
Case 3

x → [-4] [-5] → 6

x + (-4) = 6

x = 10

case 4

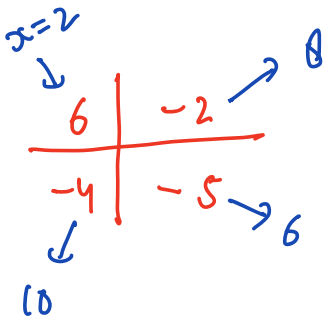


$x + (-3) = \min(8, 10)$

$x = 8 + 3$

$= 11$

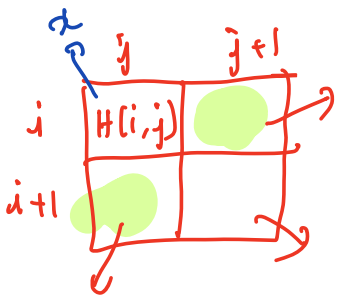
case 5



$x + 6 = \min(8, 10)$

$x + 6 = 8$

$x = 2$



→ $dp(i, j+1)$

$dp(i+1, j)$

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

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

dp[i][j] = min health req [i,j] → [N-1, M-1]

dp[i][j] = max(1, min[dp[i, j+1], dp[i+1, j]) - H[i][j])

```
func (int H[][], int N, int M, int dp[][],
                                int i , int j)
 {


 if( dp[i][j] == -1)
    {
  int a = fun( H, N, M, dp, i, j+1)
  int b = fun( H, N, M, dp, i+1; j)
  .
 dp[i][j] = max( 1, min( a,b) - H[i][j]),
 }
```