Welcome ☺

Agenda : 2 D D.P

Q Find maximum subsequence sum from a given array, where selecting adjacent element is not allowed

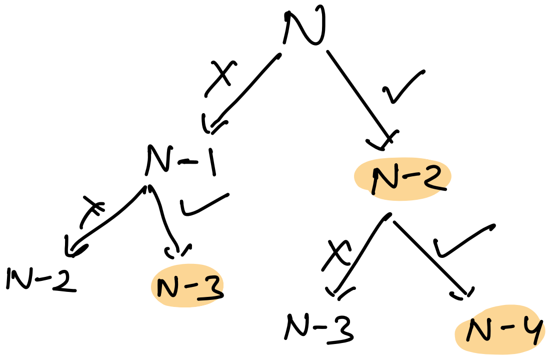
eg: A : [5, 4, 8] ⇒ 5 + 8 ⇒ 13

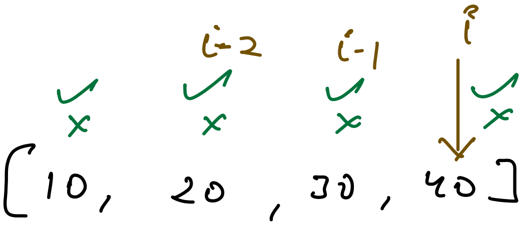
Quiz [10, 20, 30, 40]

10 + ~~(20)~~ + 40 = 50

20 + ~~(30)~~ + 40 = 60

Bruteforce





maxSum [i-1]

maxSum [i-2]

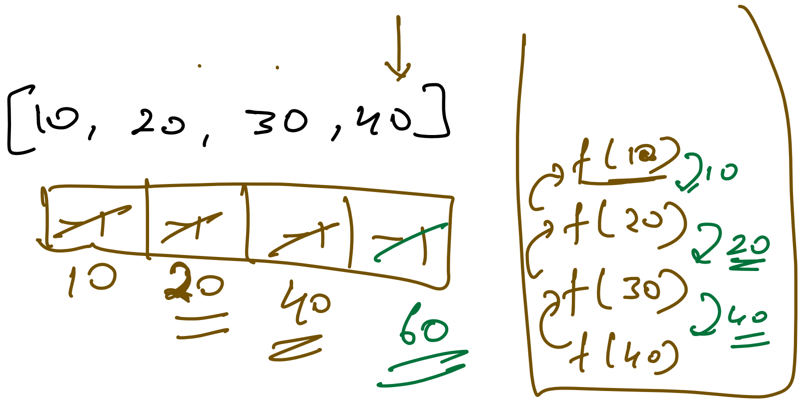
max (maxSum[i-1] , maxSum[i-2] + arr[i])

Recursive D.P

```
int dp[N] = {-1}
int maxSum ( arr[ ] , ~~index~~ N-1
{
    if( index < 0 ) return 0 ;
    if( dp[index] != -1 ) return dp[index]

  f1 = maxSum ( arr , index - 1 )

  f2 = arr[index] + maxSum ( arr , index - 2 )

  ans = max ( f1 , f2 )

  dp[index] = ans

  return ans

}
```

T.C ⇒ ~~O(2^N)~~ ⇒ O(N)

S.C = O(N+N)



[20, 10 , 30, 40]

```
dp[0] = a[0]
dp[1] = max(a[0], a[1])
for( i → 2 to N-1 )
{
    dp[i] = max ( arr[i] + dp[i-2] , dp[i-1] )
}
return dp[N-1]
```

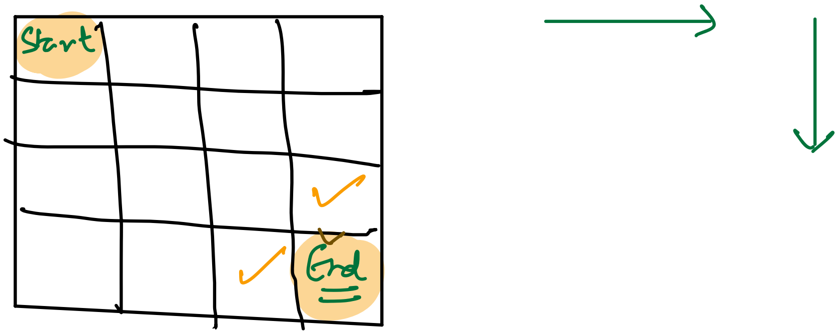
T.C ⇒ O(N)

S.C ⇒ O(N)

can reduce to

O(1)

Q Count Unique Paths.



Bruteforce

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

Recursive

```
int dp[N][m] = {-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]
}
```

$T.C = O(\cancel{2^{(N*m)}}) \rightarrow O(N*m$

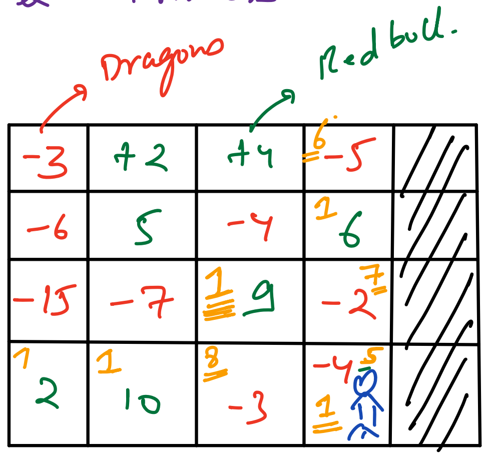
$S.C = O(N*m)$

Q No. of ways to reach bottom right with the given constraints.

	0	1	2
0	1	▨ 0	1
1	1	1	0
2	0	1	1

```
if( mat[i][j] != 0 )
        ways[i][j] = ways[i-1][j] + ways[i][j-1]
else
        ways[i][j] = 0
```

Q Dungeons & Princess



−3	+2	+4	6 −5	▨
−6	5	−4	1 6	▨
−15	−7	1 9	−2 7	▨
1 2	1 10	8 −3	−4 5 1	▨

Find min. health with which Prince should start so that he can reach to princess without dying.

Bruteforce

$$X + arr[i][j] = \min\left(dp[i+1][j] , dp[i][j+1] \right)$$

Code

```
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])
}

// Fill last row & last column

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

T.C = O(N*m)

SC = O(N*m)