Welcome ☺

Agenda: 2D
2-3 problems.

Q Maximum Subsequence Sum / House Robbery Question

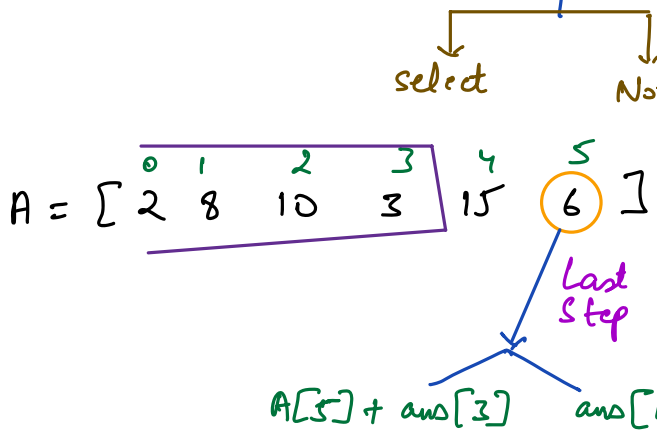
Find max. subsequence sum from a given array, where selecting adjacent elements is not allowed.

eg: [9 1 8 10] Ans = 19

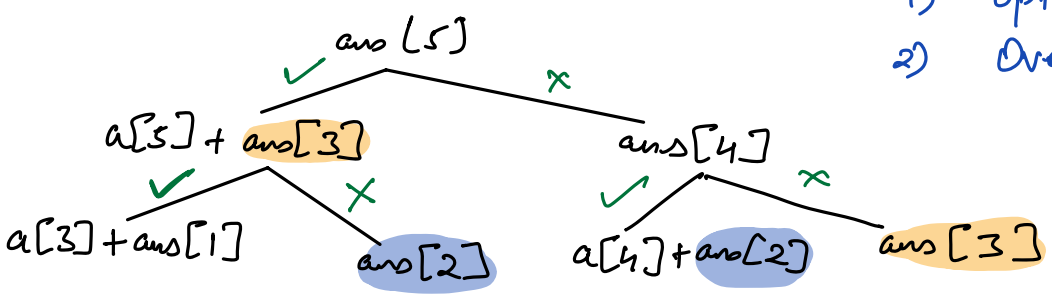
eg: [2 8 10 3 15 6] Ans = 27

Brute force

∀ valid subsequence, check and store



ans[i] ⇒ max. sum from 0 to i



1) Optimal sub ✓
2) Overlapping subprob. ✓

Code

```
ans[0] = max(A[0], 0)
ans[1] = max(A[1], ans[0])
for(i -> 2 to N-1)
{
    ans[i] = max( A[i] + ans[i-2] , ans[i-1] )
}
return ans[N-1]
```

selecting. not selecting.

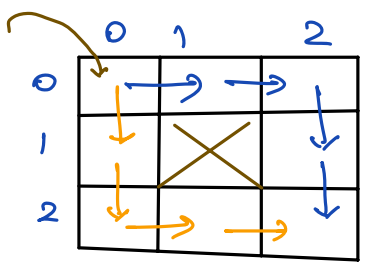
T.C → O(N)
S.C → O(N)

⇒ You can improve space from O(N) ⟶ O(1) (4 LOC)

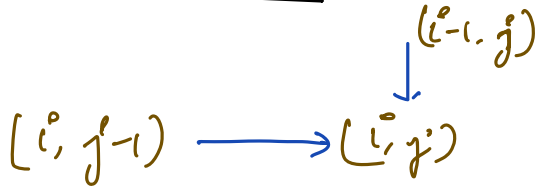
[9 ✓ 1 8 10 ✓]

ans [9 9 17 19]

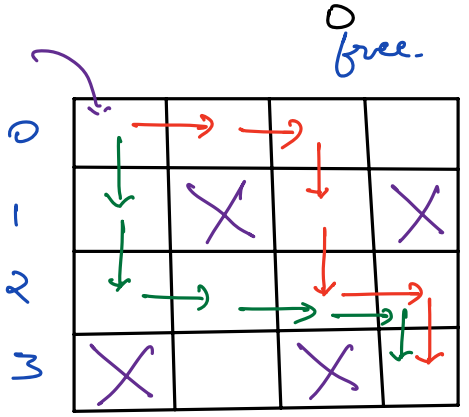
Q Given a 2D matrix.
Start → from top-left. (0,0)
End → Bottom-right.
Find # ways to move from top left to bottom right, given there are some blocked cells.



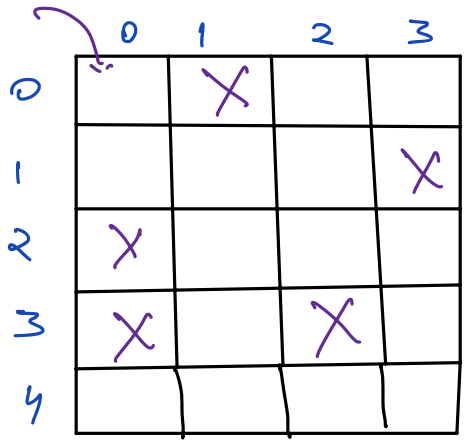
Ans = 2

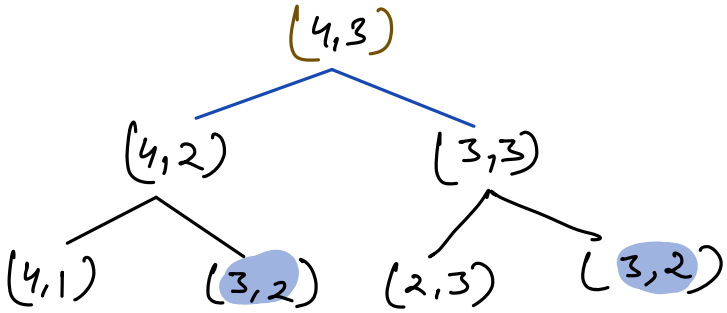


blocked
A[i][j] = 1
0 free.



Ans = 2





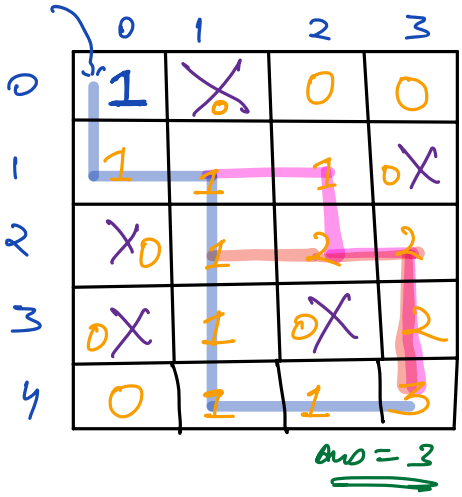
1) Optimal sub ✓
2) Overlapping subprob. ✓

ways[i][j] → # ways to reach (i,j)

ways[0][0] → 1

Code

```
for( i -> 0 to N-1)
{
  for( j -> 0 to M-1)
  {
    if( A[i][j] == 1) ways[i][j] = 0      // blocked cells
    else if( i==0 && j==0) ways[i][j] = 1
    else if( i == 0) // first row.
        ways[i][j] = ways[i][j-1]
    else if( j==0)  ways[i][j] = ways[i-1][j]
    else
        ways[i][j] = ways[i-1][j] + ways[i][j-1]
  }
}
return ways[N-1][M-1]
```



T.C → O(N*M)
S.C → O(N*M)

only store curr & prev rows
S.C → O(2M) → O(M)

H.W

Q Dungeon & Princess

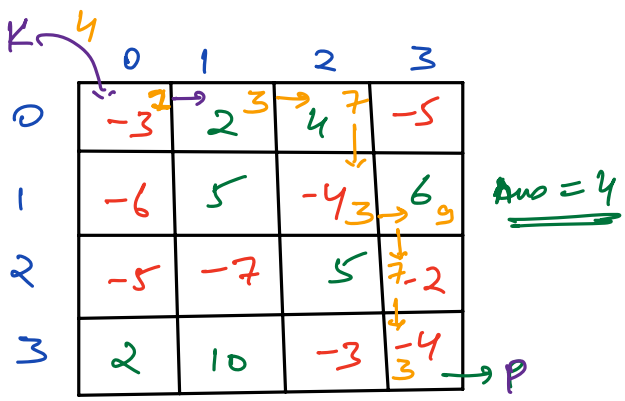
Given a matrix where A[i][j] represents a room s.t

1) A[i][j] < 0 → There is a gaurd in a room that can reduce health by |A[i][j]|

2) A[i][j] = 0 → Empty room.

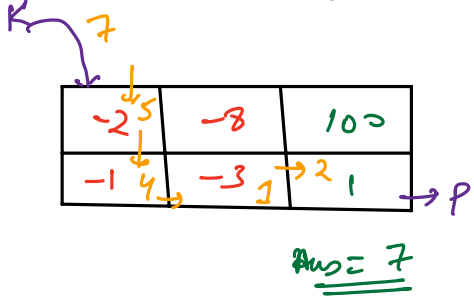
3) A[i][j] > 0 → There are some magic herbs which can increase the health by A[i][j]

Find min. initial health of knight s.t the knight can reach princess alive. If at any point health ≤ 0 ; the knight is dead.

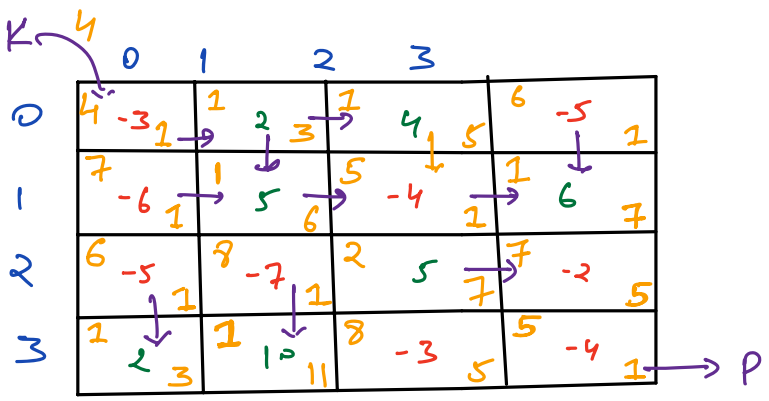


Ans = 4

Moves →



Ans = 7



ans[i][j] ⇒ min. health req. by knight to enter this cell.

Code

```
for ( i -> (N-1) to 0)
{
    for ( j -> (M-1) to 0)
    {
        if ( (i==(N-1)) && (j==(M-1)) )
            ans[i][j] = max( 1, 1 - A[i][j])
        else if ( i == N-1 ) {  // last row
            x = ans[i][j+1]
            ans[i][j] = max( x - A[i][j], 1)   // in case. X - A[i][j] becomes -ve.
        }
        else if ( j == M-1){  // last column.
            x = ans[i+1][j]
            ans[i][j] = max( x - A[i][j], 1)
        }
        else {  // all other cells
            x = min( ans[i+1][j], ans[i][j+1])
            ans[i][j] = max( x - A[i][j], 1)
        }
    }
}
return ans[0][0]
```

T.C → O(N*M)

S.C → O(N*M)

optimize. S.C → O(2m) → O(m)