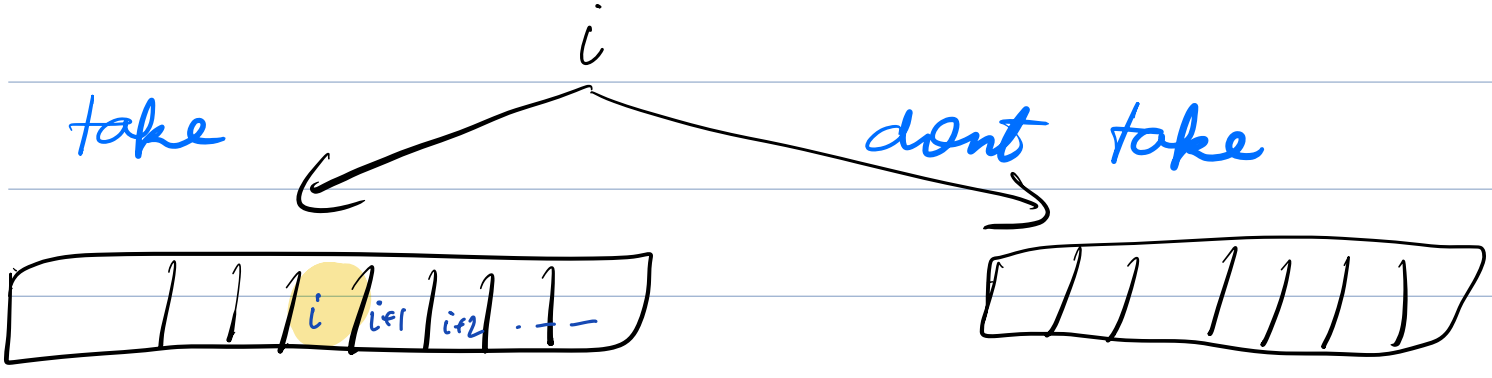
Q1 Given an array, find max sum if you are not allowed to pick adjacent (side by side) elements

House Robber

Eg - 9 4 13 ans = 22

9 4 13 24 ans = 33

Brute: Recurse on all possibilities

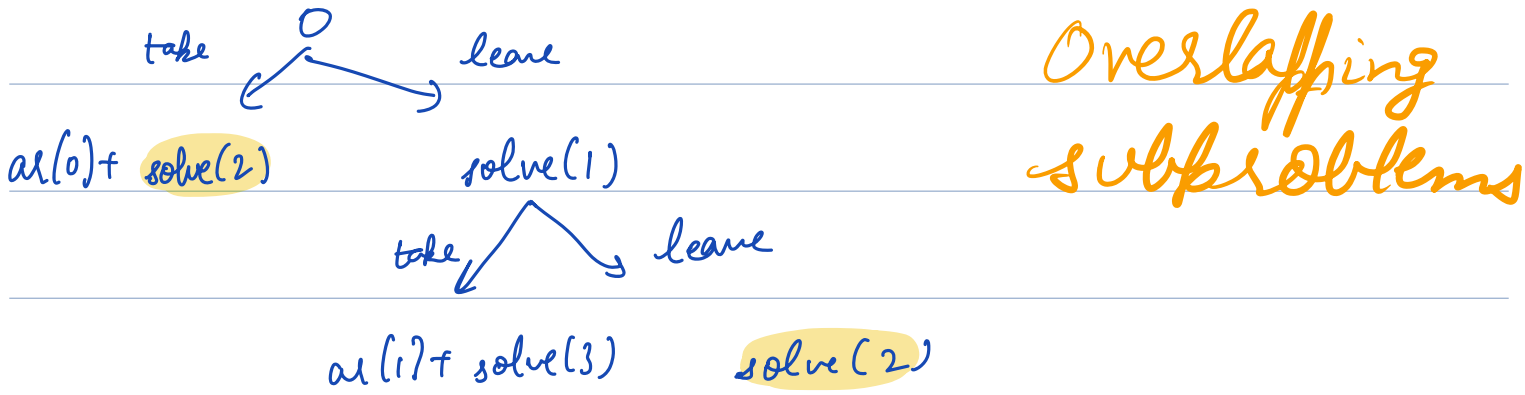


ar[i] + solve(i+2) solve(i+1)

take max

9 4 13 24

dp⇒ 33 28 24 24



Overlapping subproblems

Code

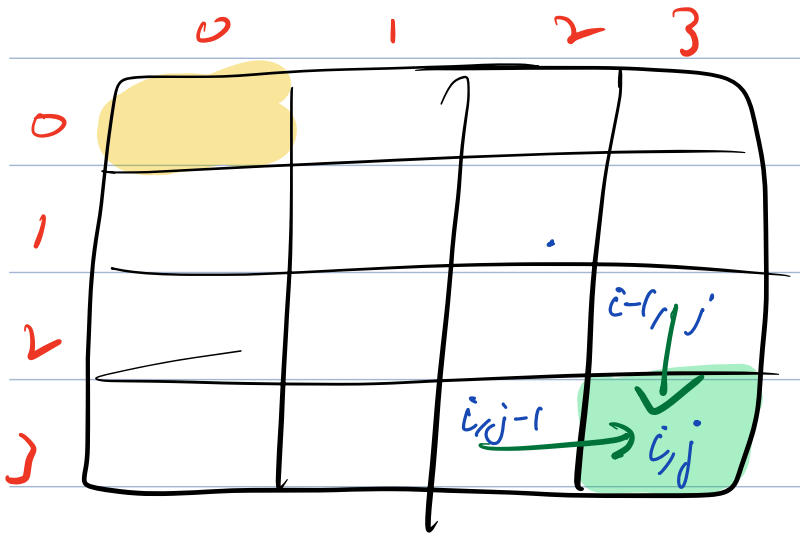
```
int dp[n]          //Initialize -1
int maxSum(int i){
{ if( i >= n )
      return 0
( if (dp[i] != -1)
      return dp[i]
( ans = max( arr[i] + maxSum(i+2),
                maxSum(i+1) )
( dp[i] = ans
( return ans
}
```

TC: } O(n)
SC: }

final answer =

maxSum(0)

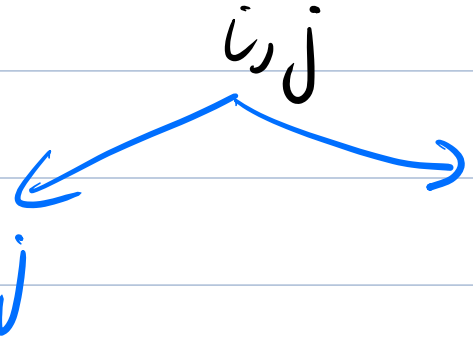
Q2 Number of ways of reaching $n-1, m-1$. Can only move down or right.



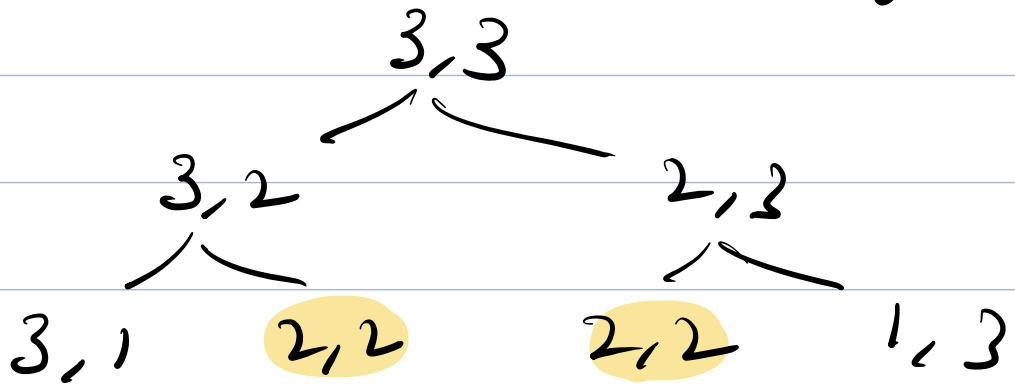
(1,2) ⇒ R R D

D R R

R D R



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



dp[n][m]

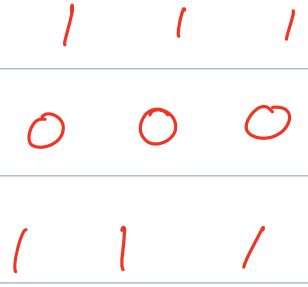
```
int ways(int i, int j){
  if(i==0 || j==0)
      return 1
  if(dp[i][j]!=-1)
      return dp[i][j]
  ans = ways(i-1,j) + ways(i,j-1)
  dp[i][j] = ans
  return ans
}
```

final ans

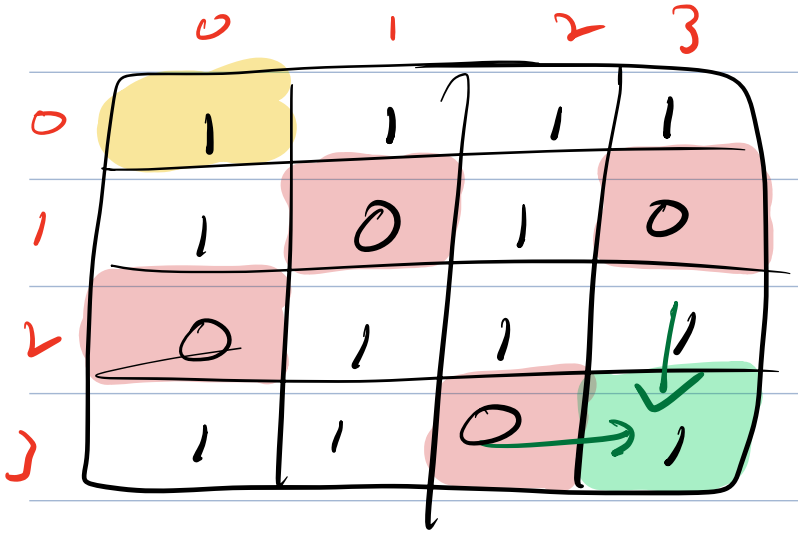
= ways(n-1, m-1)

TC: } O(n m)

SC: }



Q3 Same ques, but now some cells can be blocked.



mat[i][j] = 0
blocked

```
int ways(int i, int j){
    if(mat[i][j]==0)
        return 0
    if(i==0 && j==0)
        return 1
    if(i==0)
        return ways(0,j-1)
    if(j==0)
        return ways(i-1,0)
```

} blocked

```
if(dp[i][j] != -1)
    return dp[i][j]
```

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

}

dp[n]

dp[0] = as[0]

for(i=1; i<n; i++){

| dp[i] = max(dp[i-1], as[i] + dp[i-2])

}

Uber, GS, LinkedIn

Q4 Dragons & princess

initial

	0	1	2	3
0	-3	2	4	-5
1	-6	5	-4	6
2	-15	-7	5	-2
3	2	10	-3	-4

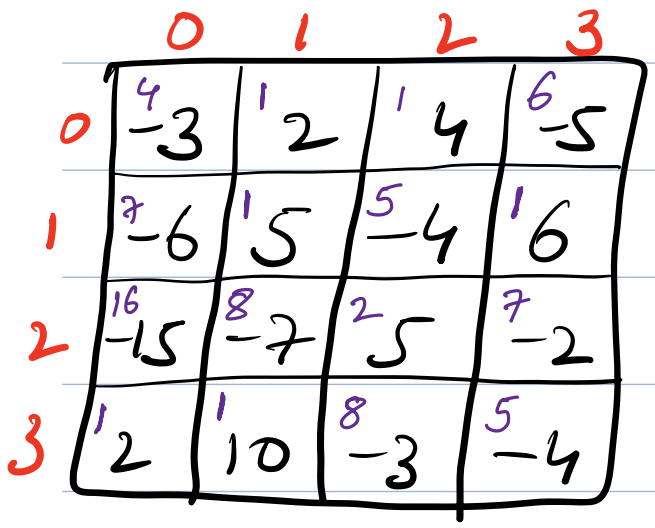
↳ princess

each cell is a chamber

dragon → decrease strength

potion → increase strength

If at any point, health ≤ 0, you die. What is the min energy that you need if you move optimally



$x + mat[i][j] = 1$

$x = 1 - mat[i][j]$

energy (i, j)

= max (1, min (energy (i+1, j) ,

energy (i, j+1))

− arr[i][j])

energy (i, j) =

max (1,

min (energy (i-1, j), energy (i, j-1))

− mat [i, j])

```
int energy ( i , j ){
    if( i == n-1 && j == m-1 ){
     ret max ( 1, 1 - arr[i][j] )
    if ( i >= n || j >= m )
        return INT_MAX
    if ( dp[i][j] != -1 )
        return dp[i][j]
    ans = max ( 1, min (energy (i+1, j) ,
                        energy (i, j+1) )
                  - arr[i][j] )
    dp[i][j] = ans
    return ans
}
```

TC: } O(NM)
SC: }

{done}