DP Concepts & Questions

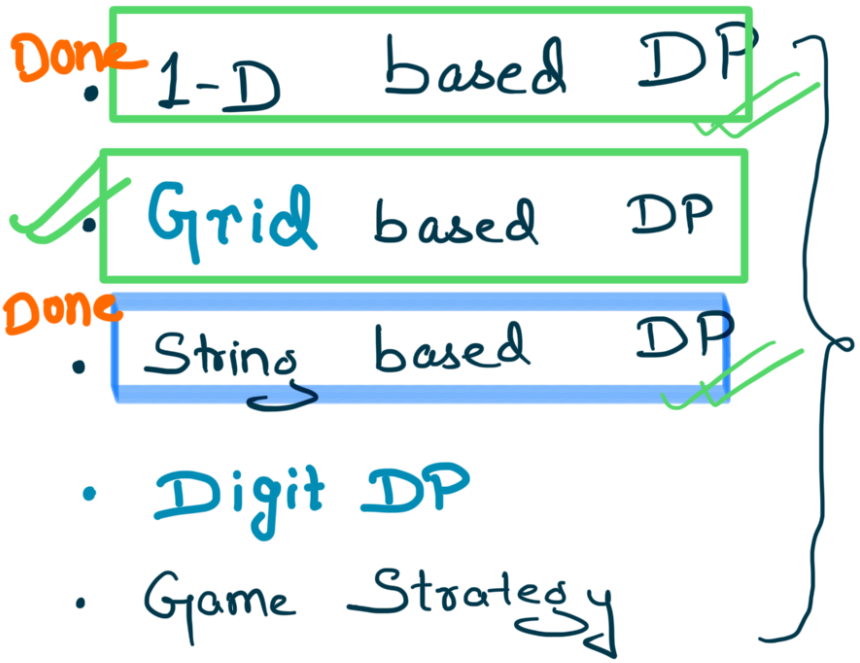
Video 34

Growth is a steady rise,
unseen but unstoppable...

MIK...

भाषण
(Motivation)

cswithMIK → Twitter
Facebook, Instagram → codestorywithMIK
whatsapp → codestorywithMIK



DP on Grids

174. Dungeon Game

Hard | Topics | Companies

The demons had captured the princess and imprisoned her in **the bottom-right corner** of a `dungeon`. The `dungeon` consists of `m x n` rooms laid out in a 2D grid. Our valiant knight was initially positioned in **the top-left room** and must fight his way through `dungeon` to rescue the princess.

The knight has an initial health point represented by a positive integer. If at any point his health point drops to `0` or below, he dies immediately.

Some of the rooms are guarded by demons (represented by negative integers), so the knight loses health upon entering these rooms; other rooms are either empty (represented as 0) or contain magic orbs that increase the knight's health (represented by positive integers).

To reach the princess as quickly as possible, the knight decides to move only **rightward** or **downward** in each step.

Return *the knight's minimum initial health so that he can rescue the princess*.

Note that any room can contain threats or power-ups, even the first room the knight enters and the

bottom-right room where the princess is imprisoned.

Example :-

-2	-3	3
-5	-10	1
10	30	-5

Output : 7

Thought Process

(Brute Force)

-2	-3	3
-5	-10	1
10	30	-5

mid

health =

l = 1 r = 10

mid = l + (r - l)/2

"Binary search on answer"

```
if (canSurvive(mid, i, j)) {
    result = mid;
    r = mid - 1;
} else {
    l = mid + 1;
}
```

(i+1, j)
(i, j+1)

Time Complexity...

l = 1 , r = $4 * 10^7$ (maxHealth)

T.C = log(maxHealth) * m * n * maxHealth

S.C = m * n * maxHealth.

Memoize:- i, j, health = i * j * h
= m * n * maxHealth

Improving Our Approach :-

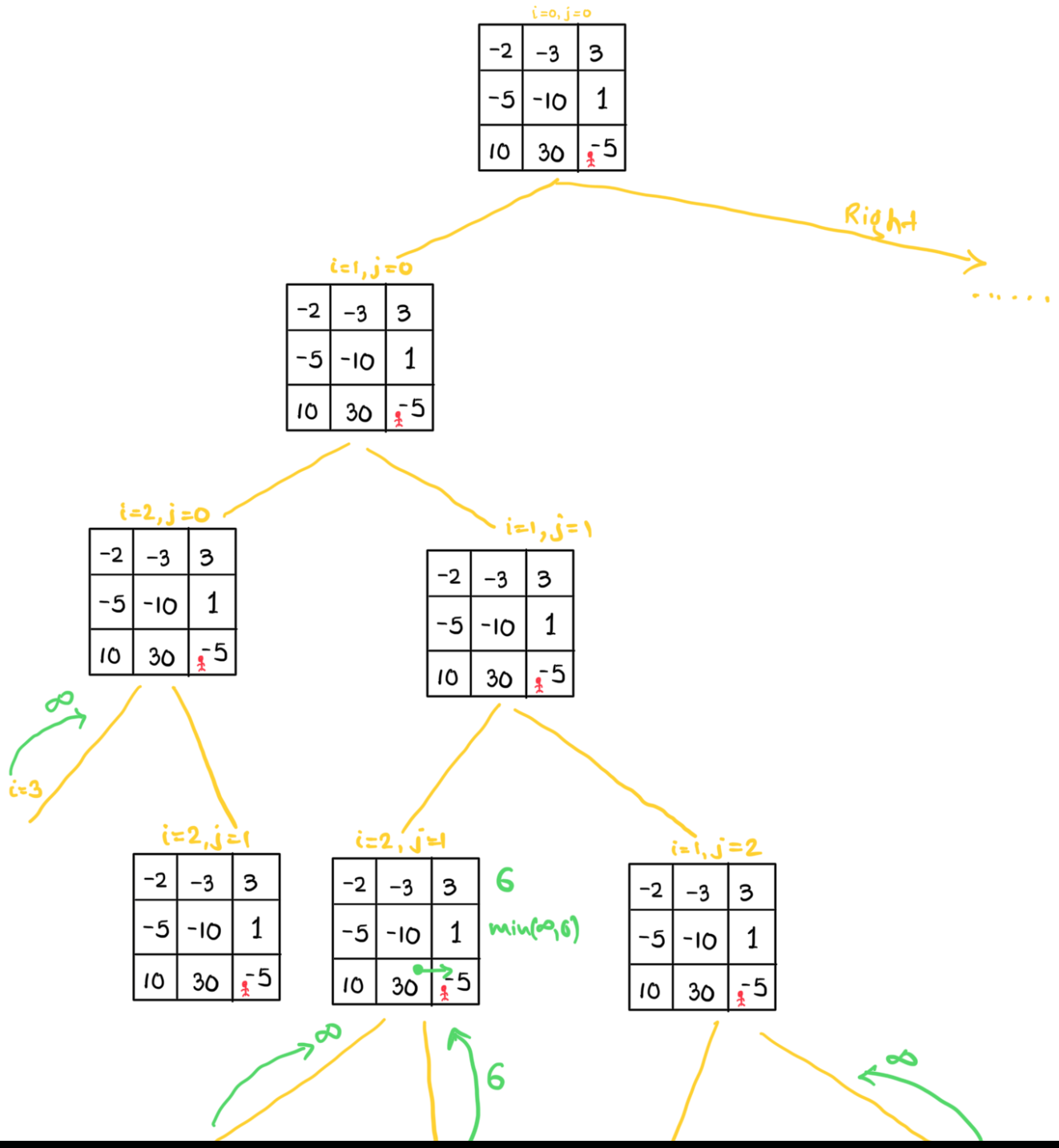
(Grid (2-d array))

↓

Grid DP. = Recursion + Memoize

options from each cell [i][j]

- → Right i.e [i][j+1]
- → Down i.e [i+1][j]



i=3, j=4

i=2, j=2

-2	-3	3
-5	-10	1
10	30	-5

i=2, j=2

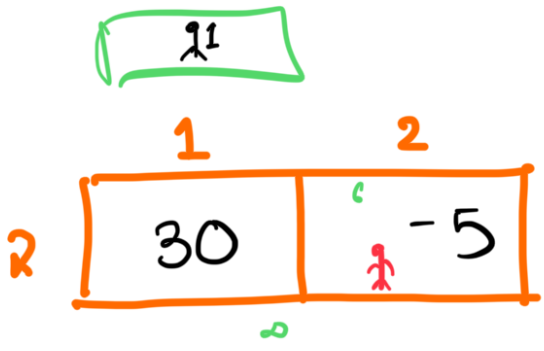
-2	-3	3
-5	-10	1
10	30	-5

i=1, j=3

6 - 5 = 1

abs(dungeon[2][2]) + 1

```
if (i == m-1 && j == n-1) {
    if (dungeon[i][j] <= 0) {
        return abs(dungeon[i][j]) + 1;
    }
    else {
        return 1;   // dun[i][j] > 0
    }
}
```



6 — dungeon[2][1]

6 — 30 = -ve

int right = 6 ;

int down = ∞ ;

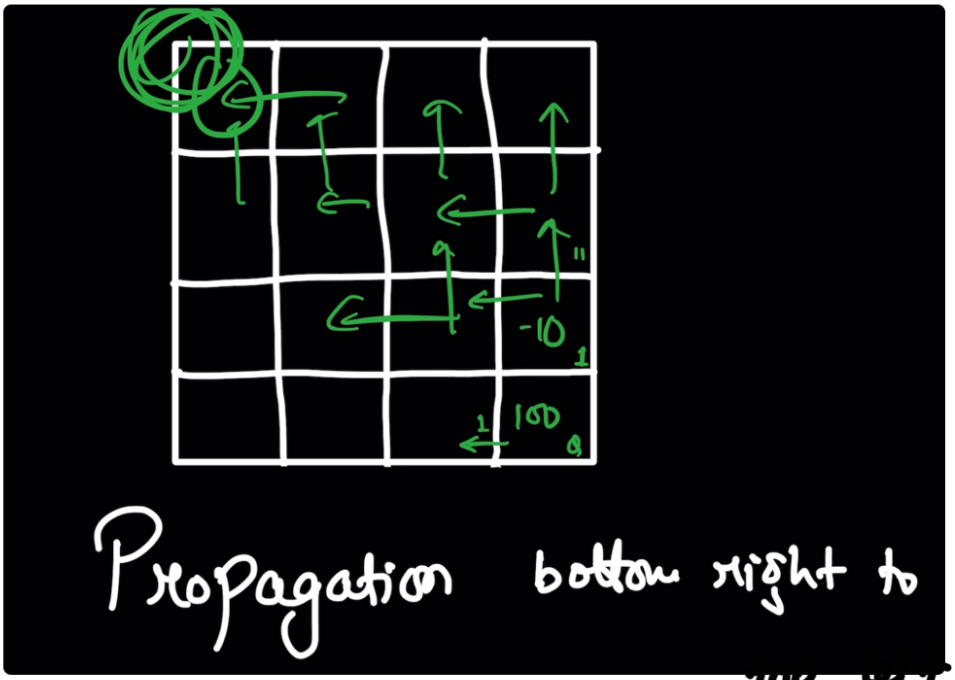
6 = min(right, down)

min(right, down) − dungeon[i][j]

result = min(6, ∞) − 30

= 6 − 30

return result < 0 ? 1 :



Story To Code ...

Bottom Up :-

Recur + Memo

```cpp
int solve(int i, int j, vector<vector<int>>& dungeon) {
    if(i >= m || j >= n) {
        return 1e9;
    }

    if(t[i][j] != -1) {
        return t[i][j];
    }

    if(i == m-1 && j == n-1) {
        if(dungeon[i][j] > 0) {
            return 1;
        }
        return abs(dungeon[i][j]) + 1;
    }

    int right = solve(i, j+1, dungeon);
    int down  = solve(i+1, j, dungeon);

    int result = min(right, down) - dungeon[i][j];

    return t[i][j] = (result > 0) ? result : 1;

}
```

State Definition:-

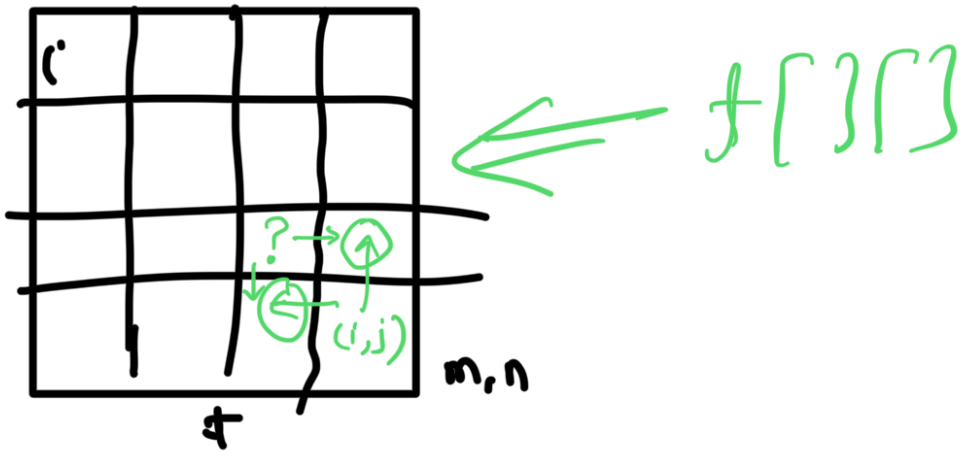
t[i][j] = min health required to reach [m-1][n-1] from [i][j]

(i,j) → (m-1, n-1)

t[i'][j'] =

(0,0) ⟶ (m-1)(n-1)

return t[0][0];



t[ ][ ]

```cpp
int solve(int i, int j, vector<vector<int>>& dungeon) {
```

```
    if(i >= m || j >= n) {
        return 1e9;
    }

    if(t[i][j] != -1) {
        return t[i][j];
    }

    if(i == m-1 && j == n-1) {
        if(dungeon[i][j] > 0) {
            return 1;
        }
        return abs(dungeon[i][j]) + 1;
    }

    int right = solve(i, j+1, dungeon);
    int down  = solve(i+1, j, dungeon);

    int result = min(right, down) - dungeon[i][j];

    return t[i][j] = (result > 0) ? result : 1;
}
```

```
for(i = m-1 ; i>=0; i--) {
    for(j = n-1; j >=0 ;j--) {
        if (i==m-1 && j==n-1) {
            if(dung[i][j] > 0)
                t[i][j] = 1
            else t[i][j] = abs(dun[i][j])
                           +1;
        }
        else {
            right = (j+1>=n)?1e9 : t[i][j+1];
            down = (i+1>=m)?1e9 : t[i+1][j];
            result = min(right, down) - dung[i][j];
            t[i][j] = result>0 ? result : 1 ;
        }
    return t[0][0];
```