Dynamic Programming

Video - 93

Note :- This playlist is only for explanation of dns & solutions.

See my "DP Concepts & dns" playlist for understanding DP from scratch...

Facebook, Instagram → codestorywithMIK

Twitter → cswithMIK

→ codestorywithMIK

Leetcode - 1289 Hard

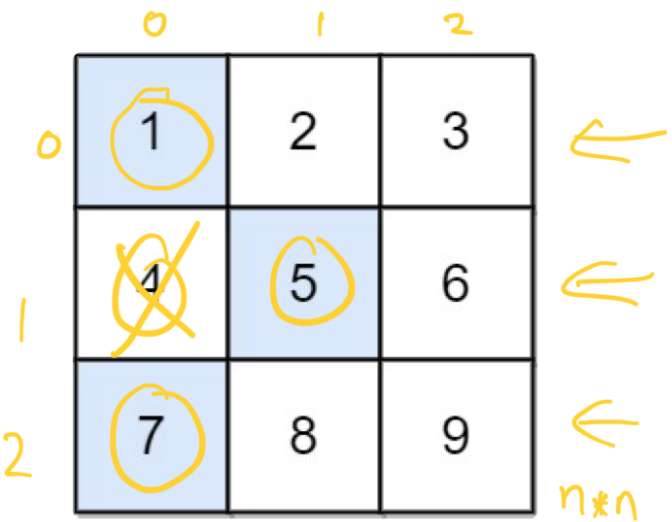
1289. Minimum Falling Path Sum II

Hard 1738 97 Add to List Share

Given an `n x n` integer matrix `grid`, return *the minimum sum of a* ***falling path with non-zero shifts***.

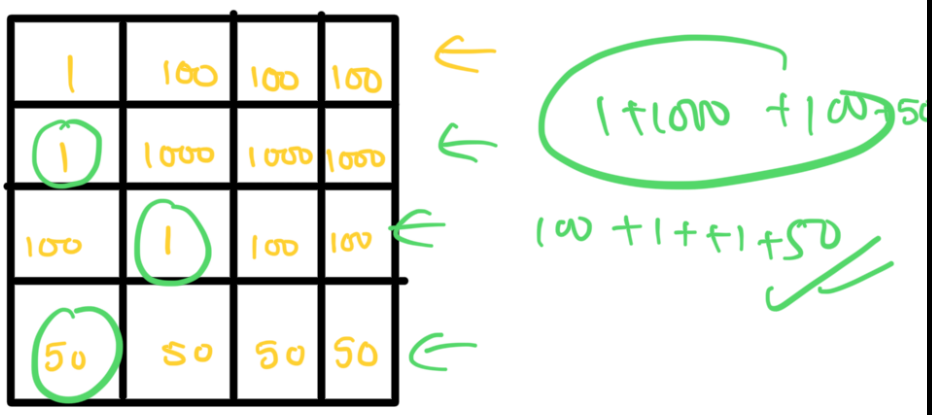
A **falling path with non-zero shifts** is a choice of exactly one element from each row of `grid` such that no two elements chosen in adjacent rows are in the same column.

Example :-



Output = 13

Why Greedy will Fail ??

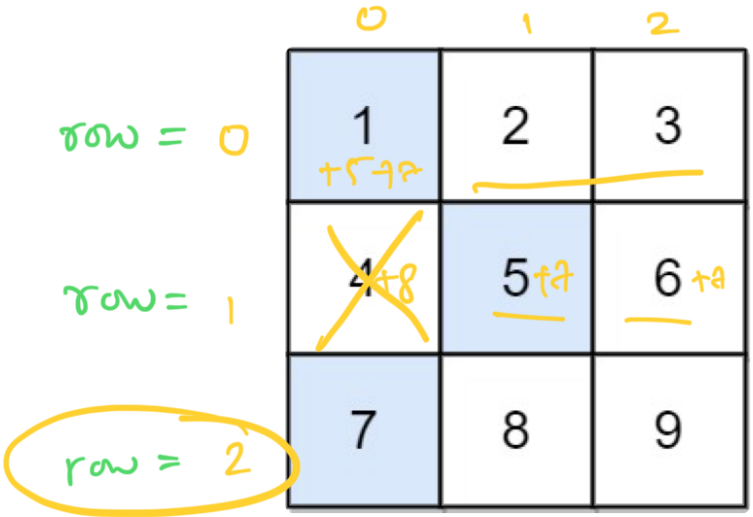


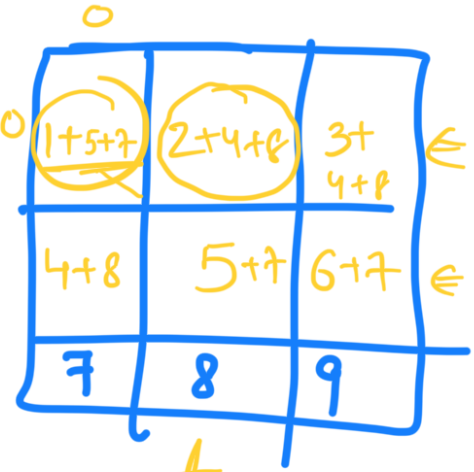
Intuition ...

Thought Process ...

If you notice, we have **options** for every row.

→ Recursion

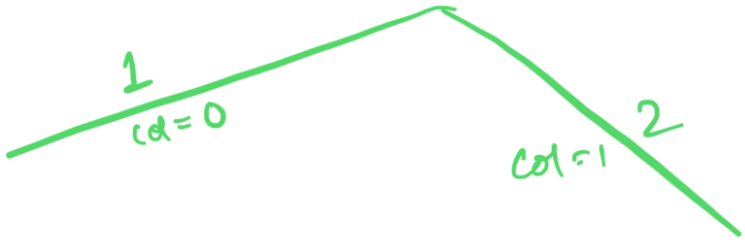


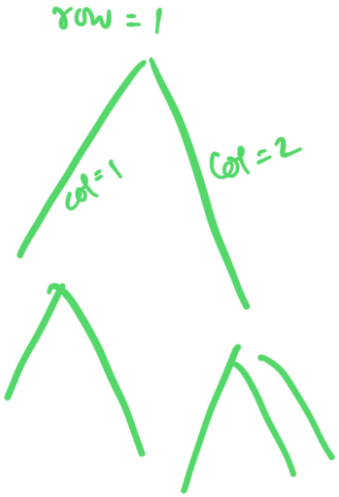


t

t[0][1] = 14

t[0][0] = 11

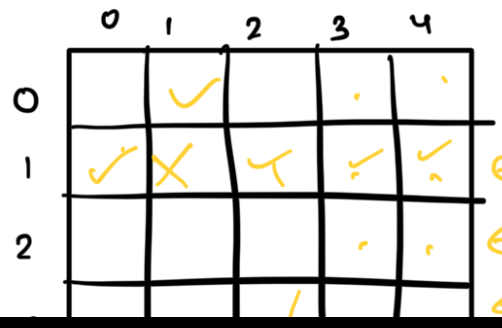





```
for (col=0; col<n; col++) {

    result = min(result, Solve(col, row 0, grid));

}
```

```
Solve(col, row, grid) {
    if (row == n-1) {
        return grid[row][col];
    }

    ans = INT_MAX;

    for(nextCol = 0; nextCol<n; nextCol++) {(n)
        if(nextCol != col) {
            ans = min(ans, solve(nextCol, row+1,g))
        }
    }

    return grid[row][col] + ans;
}
```

Time & Space ...



n without Memo :- n^n possibilities

$\leftarrow (n-1)$ to n

$\leftarrow n$

T.C = $(n^n * n)$

S.C = O(1)

$\leftarrow n$ with Memo : ( )

←n
(n*n)

O((n*n)*n)

S.C = O(n²).

Approach-2

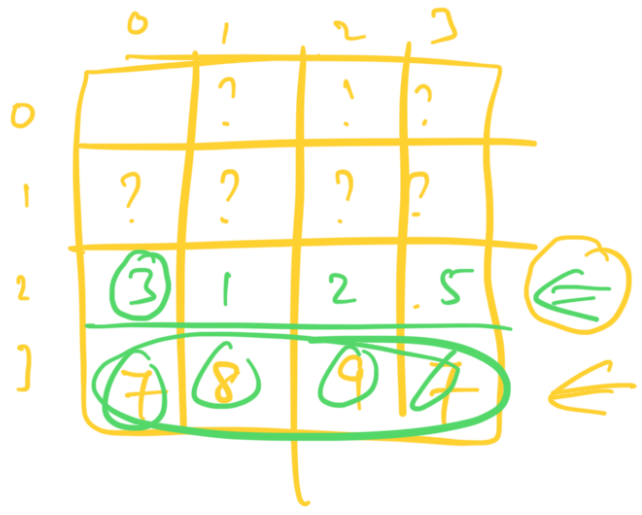
Bottom UP...

```
int solve(int row, int col, vector<vector<int>>& grid) {
    if (row == n - 1) {
        return grid[row][col];
    }

    if(t[row][col] != -1) {
        return t[row][col];
    }

    int ans = INT_MAX;
    for (int nextCol = 0; nextCol < n; nextCol++) {
        if (nextCol != col) {
            ans = min(ans, solve(row + 1, nextCol, grid));
        }
    }

    return t[row][col] = grid[row][col] + ans;
}
```

(row, col)

t

t[i][j] = minFPaSum from
⇒ (row = i, col = j) to row n-1




```
for(row = n-2 ; row >=0 ; row-) {
    for(col = 0; col<n; col++) {
```

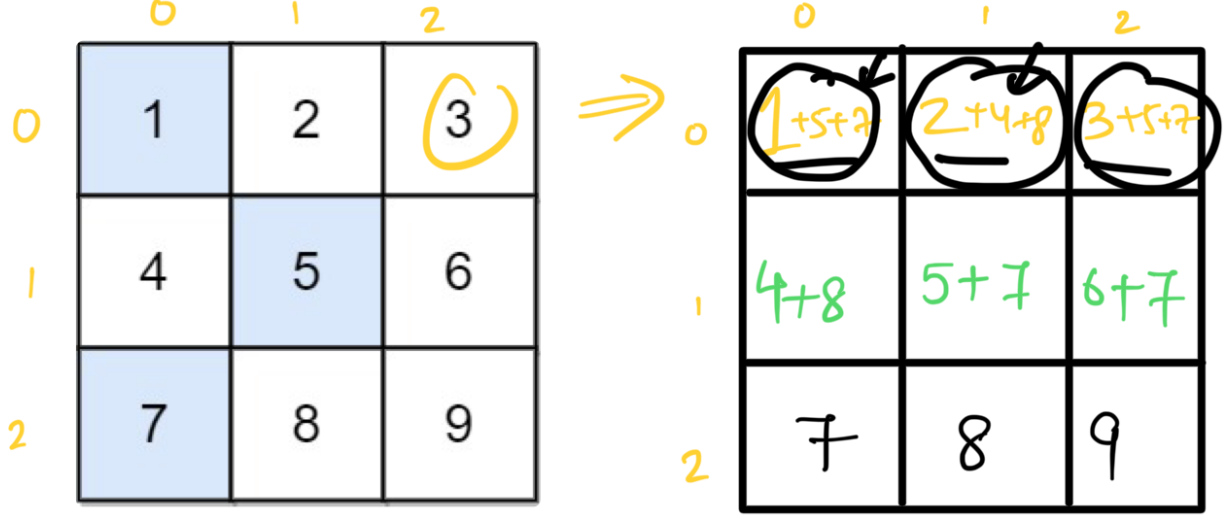
```
int ans = INT_MAX;
for(int nextCol = 0; nextCol < n; nextCol++){
    if (nextCol != col) {
        ans = min(ans, f[row+1][nextCol]);
    }
}
f[row][col] = ans + grid[row][col];
```

$T.C = O(n^3)$

$S.C = O(n^2)$

Approach-3

Further optimisation...



grid

t[][]

t[row+1][nextMinCol]

nextMinCol1 → 1st Mini

nextMinCol2 → 2nd Mini

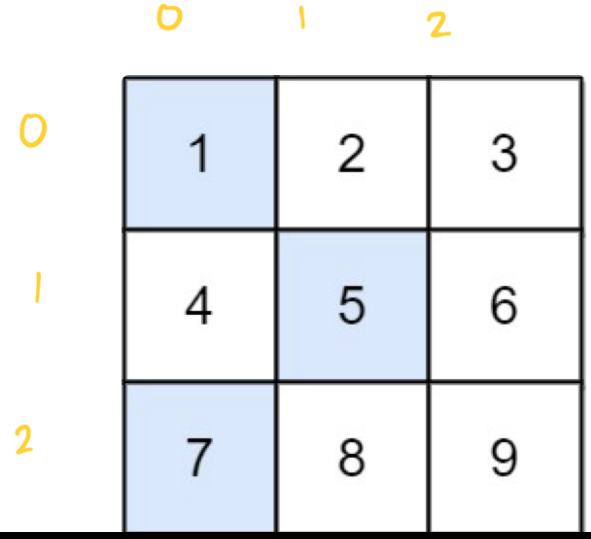
13

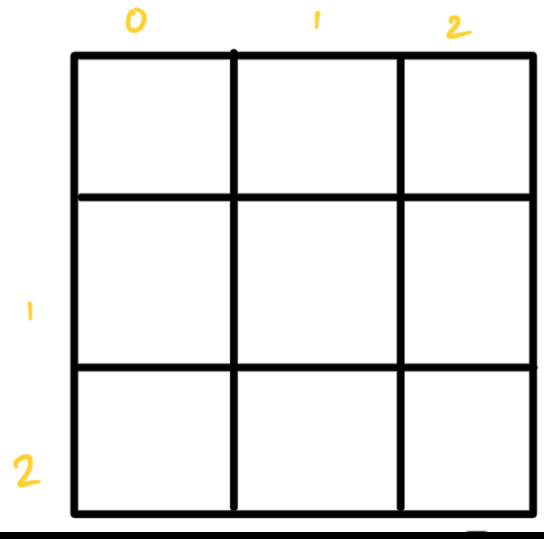
1st Row → t[0][nextMinCol]

t[0][0] = 13

Approach-4

Constant Space...





grid

x[][]