**DAY-2**

**DATE:09-10-2024**

1. Number of ways to move the ball out of the grid boundary in exactly N steps

m, n, N, i, j = 2, 2, 2, 0, 0

dp = [[0] \* n for \_ in range(m)]

dp[i][j] = 1

result = 0

for \_ in range(N):

dp\_new = [[0] \* n for \_ in range(m)]

for x in range(m):

for y in range(n):

if dp[x][y] > 0:

for dx, dy in [(1, 0), (-1, 0), (0, 1), (0, -1)]:

nx, ny = x + dx, y + dy

if 0 <= nx < m and 0 <= ny < n:

dp\_new[nx][ny] += dp[x][y]

else:

result += dp[x][y]

dp = dp\_new

print(result) # Output: 6 for the given example

2. Rob houses arranged in a circle

nums = [2, 3, 2]

if len(nums) == 1:

print(nums[0])

else:

def rob\_line(nums):

rob1, rob2 = 0, 0

for num in nums:

new\_rob = max(rob1 + num, rob2)

rob1 = rob2

rob2 = new\_rob

return rob2

print(max(rob\_line(nums[1:]), rob\_line(nums[:-1]))) # Output: 3

3. Distinct ways to climb a staircase

n = 4

a, b = 1, 1

for \_ in range(n):

a, b = b, a + b

print(a) # Output: 5 for n = 4

4. Unique paths for a robot in a grid

m, n = 7, 3

dp = [[1] \* n for \_ in range(m)]

for i in range(1, m):

for j in range(1, n):

dp[i][j] = dp[i-1][j] + dp[i][j-1]

print(dp[-1][-1]) # Output: 28 for m = 7, n = 3

5. Large groups in a string

s = "abbxxxxzzy"

result = []

i = 0

while i < len(s):

j = i

while j < len(s) and s[j] == s[i]:

j += 1

if j - i >= 3:

result.append([i, j - 1])

i = j

print(result) # Output: [[3, 6]]

6. The Game of Life

board = [[0,1,0],[0,0,1],[1,1,1],[0,0,0]]

m, n = len(board), len(board[0])

dirs = [(1, 0), (-1, 0), (0, 1), (0, -1), (1, 1), (-1, -1), (1, -1), (-1, 1)]

for i in range(m):

for j in range(n):

live\_neighbors = sum(0 <= i+di < m and 0 <= j+dj < n and abs(board[i+di][j+dj]) == 1 for di, dj in dirs)

if board[i][j] == 1 and (live\_neighbors < 2 or live\_neighbors > 3):

board[i][j] = -1

if board[i][j] == 0 and live\_neighbors == 3:

board[i][j] = 2

for i in range(m):

for j in range(n):

board[i][j] = 1 if board[i][j] > 0 else 0

print(board) # Output: [[0,0,0],[1,0,1],[0,1,1],[0,1,0]]

7. Champagne tower

poured, query\_row, query\_glass = 2, 1, 1

dp = [[0] \* (i + 1) for i in range(101)]

dp[0][0] = poured

for r in range(query\_row + 1):

for c in range(r + 1):

overflow = (dp[r][c] - 1.0) / 2.0

if overflow > 0:

dp[r + 1][c] += overflow

dp[r + 1][c + 1] += overflow

print(min(1, dp[query\_row][query\_glass])) # Output: 0.5 for poured = 2