DAA-DAY2

1.Given an m x n grid and a ball at a starting cell, find the number of ways to move the ball out of the grid boundary in exactly N steps.

Example:

Input: m=2,n=2,N=2,i=0,j=0 ·

Output: 6 ·

CODE:

def findWays(m, n, N, i, j):

from functools import lru\_cache

@lru\_cache(None)

def dfs(x, y, steps):

if x < 0 or x >= m or y < 0 or y >= n:

return 1

if steps == 0:

return 0

return (dfs(x + 1, y, steps - 1) +

dfs(x - 1, y, steps - 1) +

dfs(x, y + 1, steps - 1) +

dfs(x, y - 1, steps - 1)+

return dfs(i, j, N)

2. You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed. All houses at this place are arranged in a circle. That means the first house is the neighbor of the last one. Meanwhile, adjacent houses have security systems connected, and it will automatically contact the police if two adjacent houses were broken into on the same night.

Example:

(i) Input : nums = [2, 3, 2]

Output : The maximum money you can rob without alerting the police is 3(robbing house 1).

CODE:

def rob(nums):

def rob\_linear(houses):

prev, curr = 0, 0

for money in houses:

prev, curr = curr, max(prev + money, curr)

return curr

if len(nums) == 1:

return nums[0]

if len(nums) == 0:

return 0

return max(rob\_linear(nums[:-1]), rob\_linear(nums[1:]))

print(rob([2, 3, 2]))

3.You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top? Examples: (i) Input: n=4 Output: 5

CODE:

def climbStairs(n):

if n <= 2:

return n

a, b = 1, 2

for \_ in range(3, n + 1):

a, b = b, a + b

return b

print(climbStairs(4))

4.A robot is located at the top-left corner of a m×n grid .The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid. How many possible unique paths are there?

Example:

1. Input: m=7,n=3

Output: 28

CODE:

def uniquePaths(m, n):

# Create a 2D array with dimensions m x n

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

# Fill the dp array

for i in range(1, m):

for j in range(1, n):

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

return dp[m - 1][n - 1]

print(uniquePaths(7, 3))

5. In a string S of lowercase letters, these letters form consecutive groups of the same character. For example, a string like s = "abbxxxxzyy" has the groups "a", "bb", "xxxx", "z", and "yy". A group is identified by an interval [start, end], where start and end denote the start and end indices (inclusive) of the group. In the above example, "xxxx" has the interval [3,6]. A group is considered large if it has 3 or more characters. Return the intervals of every large group sorted in increasing order by start index.

Example 1:

Input: s = "abbxxxxzzy"

Output: [[3,6]]

Explanation: "xxxx" is the only large group with start index 3 and end index 6.

CODE:

def largeGroupPositions(s):

result = []

start = 0

for i in range(1, len(s) + 1):

if i == len(s) or s[i] != s[i - 1]:

if i - start >= 3:

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

start = i

return result

print(largeGroupPositions("abbxxxxzzy"))