# 4.16 UNIQUE PATHS IN A GRID USING COMBINATORICS

**Question:**

There is a robot on an m x n grid. The robot is initially located at the top-left corner (i.e., grid[0][0]). The robot tries to move to the bottom-right corner (i.e., grid[m - 1][n - 1]). The robot can only move either down or right at any point in time. Given the two integers m and n, return the number of possible unique paths that the robot can take to reach the bottom-right corner. The test cases are generated so that the answer will be less than or equal to 2 * 10 9.

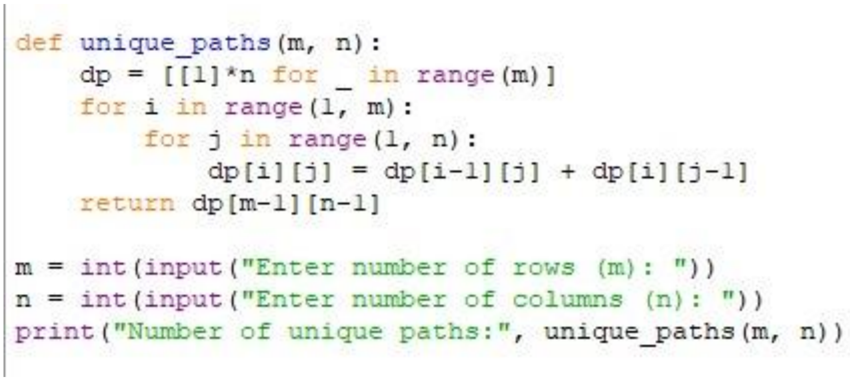
**AIM**

To compute the total number of unique paths from the top-left to the bottom-right corner of an m × n grid using combinatorics.

**ALGORITHM**

1. Create a 2D DP table dp[m][n] where dp[i][j] represents the number of ways to reach cell (i, j).
2. Initialize the first row and first column with 1, since there is only one way to move straight along them.
3. For each cell (i, j) with i > 0 and j > 0:
    - The number of ways = dp[i-1][j] + dp[i][j-1] (from top or left).
4. The result will be stored in dp[m-1][n-1].

**PROGRAM**



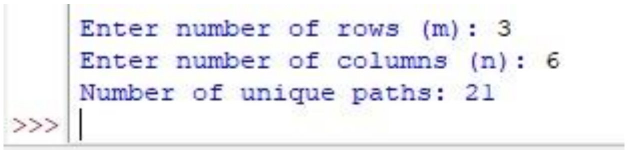
```python
def unique_paths(m, n):
    dp = [[1]*n for _ in range(m)]
    for i in range(1, m):
        for j in range(1, n):
            dp[i][j] = dp[i-1][j] + dp[i][j-1]
    return dp[m-1][n-1]

m = int(input("Enter number of rows (m): "))
n = int(input("Enter number of columns (n): "))
print("Number of unique paths:", unique_paths(m, n))
```

Input:

Enter number of rows (m): 3

Enter number of columns (n): 6

Output:



```
Enter number of rows (m): 3
Enter number of columns (n): 6
Number of unique paths: 21
>>> |
```

**RESULT:**

Thus the program is successfully executed and the output is verified.

**PERFORMANCE ANALYSIS:**

- Time Complexity: O(m x n)
- Space Complexity: O(m x n)