# Documentation: Unique Paths

## Overview

This algorithm calculates the number of unique paths a robot can take to move from the top-left corner to the bottom-right corner of an `m x n` grid. The robot is only allowed to move either down or right at any given point.

## Problem Statement

Given two integers `m` and `n`, representing the dimensions of the grid, the task is to determine the number of unique paths the robot can take to reach the bottom-right corner.

## Example

### Input:

- `m = 3`
- `n = 7`

### Output:

- `28`

**Explanation:** There are 28 unique paths for the robot to move from the top-left corner to the bottom-right corner in a 3x7 grid.

## Solution Approach

The problem can be solved using dynamic programming. The solution initializes a 2D array `dp` of size `m x n` to store the number of unique paths. It then iterates through the grid, filling the `dp` array based on the constraints of the problem.

1. **Initialization:** Set the top-left cell of `dp` to 1, as there is only one way to reach it. Fill the first row and first column of `dp` with 1s since the robot can only move right from the top row and down from the leftmost column.

2. **Dynamic Programming:** Iterate through the remaining cells of `dp`. For each cell `(i, j)`, the number of unique paths to reach it is the sum of the number of unique paths to reach the cell above it `(i-1, j)` and the cell to its left `(i, j-1)`.

3. **Return Result:** Finally, return the value stored in the bottom-right corner of `dp`, which represents the total number of unique paths.

## Complexity Analysis

- **Time Complexity:** The time complexity of this solution is O(m * n), where `m` and `n` are the dimensions of the grid.
- **Space Complexity:** The space complexity is also O(m * n) since we use a 2D array to store the number of unique paths.