

Minimum Path Sum

Grid DP for minimum cost path ■

● MEDIUM

■ Hook

Same grid, but now each cell has a cost. Find the path from top-left to bottom-right that minimizes total cost. Move only right or down. Variation of Unique Paths — instead of counting, we're optimizing.

■ DP Grid

```
grid:      dp (min cost to reach):
[1][3][1]  [1][4][5]
[1][5][1]  [2][7][6]
[4][2][1]  [6][8][7] ← answer=7
dp[i][j] = grid[i][j] + min(dp[i-1][j], dp[i][j-1])
First row: prefix sums of row 0
First col: prefix sums of col 0
```

■ Code

```
public int minPathSum(int[][] grid) {
    int m=grid.length, n=grid[0].length;
    int[][] dp = new int[m][n];
    dp[0][0] = grid[0][0];
    for (int i=1; i<m; i++) dp[i][0] = dp[i-1][0]+grid[i][0];
    for (int j=1; j<n; j++) dp[0][j] = dp[0][j-1]+grid[0][j];
    for (int i=1; i<m; i++)
        for (int j=1; j<n; j++)
            dp[i][j] = grid[i][j] + Math.min(dp[i-1][j], dp[i][j-1]);
    return dp[m-1][n-1];
}
```

■ TIME

$O(m \times n)$

■ SPACE

$O(m \times n) \rightarrow O(n)$

- ✓ $dp[i][j] = grid[i][j] + \min(\text{from above, from left})$
- ✓ Base: prefix sum first row and col
- ✓ Can modify grid in-place to save space
- ✓ Time: $O(mn)$ | Space: $O(1)$ in-place

■ MENTAL MODEL — Remember This Forever

At each cell, the cheapest path to it = cell's own cost + cheaper of the two entry paths.
Greedy build optimal cost table. The robot 'remembers' the cheapest way to arrive at each intersection.