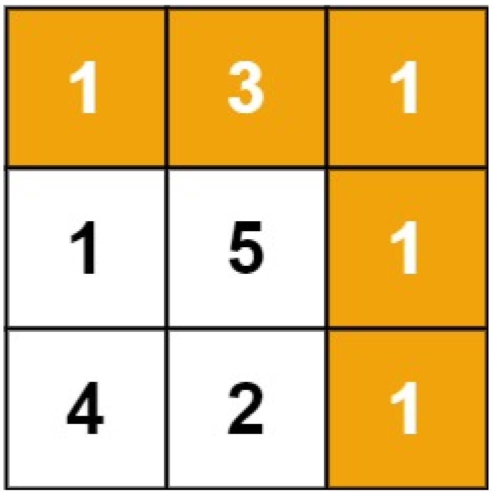
Get minimum cost to reach from one position to another.

Given a `m x n` `grid` filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all numbers along its path.

Note: You can only move either down or right at any point in time.



Input: grid = [[1,3,1],[1,5,1],[4,2,1]]
Output: 7
Explanation: Because the path 1 → 3 → 1 → 1 → 1 minimizes the sum.

```
def minPathSum(self, grid: List[List[int]]) -> int:
        n = len(grid)
        m = len(grid[0])

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

        for i in range(n-1,-1,-1):
            for j in range(m-1,-1,-1):
                if (i == n-1 and j == m-1):
                    dp[i][j] = grid[i][j]
                elif (i==n-1):
                    dp[i][j] = dp[i][j+1]+grid[i][j]
                elif (j == m-1):
                    dp[i][j] = dp[i+1][j]+grid[i][j]
                else:
                    dp[i][j] = min(dp[i][j+1],dp[i+1][j])+grid[i][j]
        return dp[0][0]
```