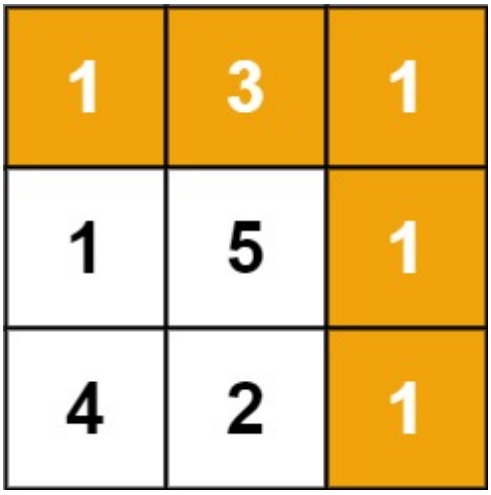
64. Minimum Path Sum

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.

Example 1:




```
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.
```

Example 2:

```
Input: grid = [[1,2,3],[4,5,6]]
Output: 12
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 200`
- `0 <= grid[i][j] <= 100`

```
import heapq
class Solution:
    def minPathSum(self, grid: List[List[int]]) -> int:
        visited = [[False]*len(grid[0]) for i in range(len(grid))]
```

```
            queue = []
            heapq.heappush(queue,(grid[0][0],(0,0)))
            while len(queue)>0:
                cost,(x,y) = heapq.heappop(queue)
                if x==len(grid)-1 and y==len(grid[0])-1:
                    return cost
                if visited[x][y]==True:
                    continue

                visited[x][y] = True
                for dx,dy in ((x+1,y),(x,y+1)):
                    if dx<=len(grid)-1 and dx>=0 and dy<=len(grid[0])-1
and dy>=0  and visited[dx][dy]==False:
                        heapq.heappush(queue,(cost+grid[dx][dy],(dx,dy)))
```

- ```
    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]
    ```
    ```