**Minimum Path Sum**

**Medium**

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:**

Calendar

Description automatically generated

**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

class Solution {

public:

    int dp[209][206];

    int minCost(int i, int j, vector<vector<int>> &grid) {

        if (i==grid.size()-1 and j==grid[0].size()-1) return grid[i][j];

        if (dp[i][j]!=-1) return dp[i][j];

        int a=INT\_MAX, b=INT\_MAX;

        if (i+1<grid.size()) a=minCost(i+1, j, grid);

        if (j+1<grid[0].size()) b=minCost(i, j+1, grid);

        return dp[i][j]=grid[i][j]+min(a, b);

    }

    int minPathSum(vector<vector<int>>& grid) {

        /\*memset(dp, -1, sizeof(dp));

        return minCost(0, 0, grid);\*/

        int m=grid.size(), n=grid[0].size();

        int dp[m][n];

        memset(dp, 0, sizeof(dp));

        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 i=1; i<n; i++) dp[0][i]=dp[0][i-1]+grid[0][i];

        for (int i=1; i<m; i++) {

            for (int j=1; j<n; j++) {

                dp[i][j]=grid[i][j]+min(dp[i-1][j], dp[i][j-1]);

            }

        }

        return dp[m-1][n-1];

    }

};