Min Cost Path in C++

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
    int n = 4; // Number of rows
    int m = 4; // Number of columns
    int grid[4][4] = {
        {8, 2, 1, 6},
        {6, 5, 5, 2},
        {2, 1, 0, 3},
        {7, 2, 2, 4}
    };

    // Initialize dp array
    vector<vector<int>> dp(n, vector<int>(m, 0));

    // Fill dp array from bottom-right to top-left
    for (int i = n - 1; i >= 0; i--) {
        for (int j = m - 1; j >= 0; j--) {
            if (i == n - 1 && j == m - 1) {
                dp[i][j] = grid[i][j];
            } else if (i == n - 1) {
                dp[i][j] = dp[i][j + 1] + grid[i][j];
            } else if (j == m - 1) {
                dp[i][j] = dp[i + 1][j] + grid[i][j];
            } else {
                dp[i][j] = grid[i][j] + min(dp[i][j + 1], dp[i + 1]
[j]);
            }
        }
    }

    // Print the minimum cost path sum
    cout << dp[0][0] << endl;

    return 0;
}
```

Dry Run

Input Grid:

```
grid = {
    {8, 2, 1, 6},
    {6, 5, 5, 2},
    {2, 1, 0, 3},
    {7, 2, 2, 4}
}
```

Steps:

1. **Initialization**:
    - Create a dp table with dimensions' n × m (initialized to 0).
2. **Filling the DP Table**:
    - Start from the bottom-right corner (n-1, m-1) and work backwards.

Filling DP Table:

- **Bottom-right corner (i = 3, j = 3)**:

  dp[3][3] = grid[3][3] = 4

- **Last row (i = 3)**:

  dp[3][2] = grid[3][2] + dp[3][3] = 2 + 4 = 6
  dp[3][1] = grid[3][1] + dp[3][2] = 2 + 6 = 8
  dp[3][0] = grid[3][0] + dp[3][1] = 7 + 8 = 15

- **Last column (j = 3)**:

  dp[2][3] = grid[2][3] + dp[3][3] = 3 + 4 = 7
  dp[1][3] = grid[1][3] + dp[2][3] = 2 + 7 = 9
  dp[0][3] = grid[0][3] + dp[1][3] = 6 + 9 = 15

- **Remaining cells**:
    - **Row 2**:

      dp[2][2] = grid[2][2] + min(dp[2][3], dp[3][2]) = 0 + min(7, 6) = 6
      dp[2][1] = grid[2][1] + min(dp[2][2], dp[3][1]) = 1 + min(6, 8) = 7
      dp[2][0] = grid[2][0] + min(dp[2][1], dp[3][0]) = 2 + min(7, 15) = 9

    - **Row 1**:

      dp[1][2] = grid[1][2] + min(dp[1][3], dp[2][2]) = 5 + min(9, 6) = 11
      dp[1][1] = grid[1][1] + min(dp[1][2], dp[2][1]) = 5 + min(11, 7) = 12
      dp[1][0] = grid[1][0] + min(dp[1][1], dp[2][0]) = 6 + min(12, 9) = 15

<table>
<tr>
<td></td>
<td>

Row 0:

dp[0][2] = grid[0][2] + min(dp[0][3], dp[1][2]) = 1 + min(15, 11) = 12
dp[0][1] = grid[0][1] + min(dp[0][2], dp[1][1]) = 2 + min(12, 12) = 14
dp[0][0] = grid[0][0] + min(dp[0][1], dp[1][0]) = 8 + min(14, 15) = 22

Final DP Table:

dp = {
{22, 14, 12, 15},
{15, 12, 11, 9},
{9, 7, 6, 7},
{15, 8, 6, 4}
}
</td>
</tr>
<tr>
<td colspan="2">Output:
22</td>
</tr>
</table>