|  |  |
| --- | --- |
| **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;  } | **Input Grid:**  [8, 2, 1, 6]  [6, 5, 5, 2]  [2, 1, 0, 3]  [7, 2, 2, 4]  We’re filling the dp[i][j] table from **bottom-right to top-left**.  **✅ DP Formula Recap:**  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]);  **🧮 DP Table (Filled from bottom-right):**  Let's build dp[i][j] step by step:  **Starting from dp[3][3] = grid[3][3] = 4**  Then filling right-to-left and bottom-to-top:   | **i\j** | **0** | **1** | **2** | **3** | | --- | --- | --- | --- | --- | | 0 | *?* | *?* | *?* | *?* | | 1 | *?* | *?* | *?* | *?* | | 2 | *?* | *?* | *?* | *?* | | 3 | 15 | 8 | 6 | 4 |   Now build upward:  **Row 2:**   * dp[2][3] = grid[2][3] + dp[3][3] = 3 + 4 = 7 * dp[2][2] = 0 + min(7, 6) = 6 * dp[2][1] = 1 + min(6, 8) = 7 * dp[2][0] = 2 + min(7, 15) = 9   **Row 1:**   * dp[1][3] = 2 + 7 = 9 * dp[1][2] = 5 + min(9, 6) = 11 * dp[1][1] = 5 + min(11, 7) = 12 * dp[1][0] = 6 + min(12, 9) = 15   **Row 0:**   * dp[0][3] = 6 + 9 = 15 * dp[0][2] = 1 + min(15, 11) = 12 * dp[0][1] = 2 + min(12, 12) = 14 * dp[0][0] = 8 + min(14, 15) = 22   **✅ Final DP Table:**   | **i\j** | **0** | **1** | **2** | **3** | | --- | --- | --- | --- | --- | | 0 | 22 | 14 | 12 | 15 | | 1 | 15 | 12 | 11 | 9 | | 2 | 9 | 7 | 6 | 7 | | 3 | 15 | 8 | 6 | 4 |   **🧾 Output:**  22 |
| Output: 22 | |