|  |  |
| --- | --- |
| **Paint Houses in C++** | |
| #include <iostream>  #include <vector>  #include <algorithm>  using namespace std;  int main() {      // Input array representing costs to paint each house with three colors      vector<vector<int>> arr = {{1, 5, 7}, {5, 8, 4}, {3, 2, 9}, {1, 2, 4}};      int n = arr.size(); // Number of houses      // Initialize dp array      vector<vector<long long>> dp(n, vector<long long>(3, 0));      // Base case: First row initialization      dp[0][0] = arr[0][0];      dp[0][1] = arr[0][1];      dp[0][2] = arr[0][2];      // Fill dp array from second row onwards      for (int i = 1; i < n; i++) {          dp[i][0] = arr[i][0] + min(dp[i - 1][1], dp[i - 1][2]);          dp[i][1] = arr[i][1] + min(dp[i - 1][0], dp[i - 1][2]);          dp[i][2] = arr[i][2] + min(dp[i - 1][0], dp[i - 1][1]);      }      // Find the minimum cost to paint all houses      long long ans = min(dp[n - 1][0], min(dp[n - 1][1], dp[n - 1][2]));      // Output the minimum cost      cout << ans << endl;      return 0;  } | **Input Matrix (Cost of painting houses):**  House 0: [1, 5, 7]  House 1: [5, 8, 4]  House 2: [3, 2, 9]  House 3: [1, 2, 4]  We denote the colors as:   * 0 → Red * 1 → Blue * 2 → Green   **🧮 DP Table Filling Explanation:**   | **House** | **dp[i][0] (Red)** | **dp[i][1] (Blue)** | **dp[i][2] (Green)** | | --- | --- | --- | --- | | 0 | 1 | 5 | 7 | | 1 | 5 + min(5,7) = 10 | 8 + min(1,7) = 9 | 4 + min(1,5) = 5 | | 2 | 3 + min(9,5) = 8 | 2 + min(10,5) = 7 | 9 + min(10,9) = 18 | | 3 | 1 + min(7,18) = 8 | 2 + min(8,18) = 10 | 4 + min(8,7) = 11 |   **✅ Final DP Table:**   | **House** | **Red** | **Blue** | **Green** | | --- | --- | --- | --- | | 0 | 1 | 5 | 7 | | 1 | 10 | 9 | 5 | | 2 | 8 | 7 | 18 | | 3 | 8 | 10 | 11 |   **🧾 Output:**  The minimum total cost is:  min(8, 10, 11) = 8 |
| Output:- 8 | |