|  |  |
| --- | --- |
| **Edit Distance C++** | |
| #include <iostream>  #include <string>  #include <algorithm>  using namespace std;  int main() {  string s1 = "cat";  string s2 = "cut";  int m = s1.length();  int n = s2.length();  int dp[m + 1][n + 1];  // Base cases  for (int i = 0; i <= m; i++) dp[i][0] = i; // Deleting all characters  for (int j = 0; j <= n; j++) dp[0][j] = j; // Inserting all characters  // Fill the DP table  for (int i = 1; i <= m; i++) {  for (int j = 1; j <= n; j++) {  if (s1[i - 1] == s2[j - 1]) {  dp[i][j] = dp[i - 1][j - 1]; // No operation needed  } else {  dp[i][j] = 1 + min({dp[i - 1][j - 1], // Replace  dp[i - 1][j], // Delete  dp[i][j - 1]}); // Insert  }  }  }  cout << dp[m][n] << endl; // Output the minimum edit distance  return 0;  } | **Dry Run (**s1 = "cat", s2 = "cut"**)** **Step 1: Initialize the DP Table**  The **first row** (when s1 is empty) represents **insertions**, and the **first column** (when s2 is empty) represents **deletions**.   | **i\j** | **0** | **1** | **2** | **3** | | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 2 | 3 | | 1 | 1 | - | - | - | | 2 | 2 | - | - | - | | 3 | 3 | - | - | - |   **Step 2: Fill the DP Table**  **Iteration 1 (i=1, s1="c"):**   * j=1, s2="c" → **Same character, copy diagonal** → dp[1][1] = dp[0][0] = 0 * j=2, s2="cu" → **Insert 'u'** → dp[1][2] = min(Replace:1, Delete:2, Insert:0) + 1 = 1 * j=3, s2="cut" → **Insert 't'** → dp[1][3] = min(Replace:2, Delete:3, Insert:1) + 1 = 2  | **i\j** | **0** | **1** | **2** | **3** | | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 2 | 3 | | 1 | 1 | 0 | 1 | 2 | | 2 | 2 | - | - | - | | 3 | 3 | - | - | - |   **Iteration 2 (i=2, s1="ca"):**   * j=1, s2="c" → **Delete 'a'** → dp[2][1] = min(Replace:1, Delete:0, Insert:2) + 1 = 1 * j=2, s2="cu" → **Replace 'a' with 'u'** → dp[2][2] = min(Replace:0, Delete:1, Insert:1) + 1 = 1 * j=3, s2="cut" → **Insert 't'** → dp[2][3] = min(Replace:1, Delete:2, Insert:1) + 1 = 2  | **i\j** | **0** | **1** | **2** | **3** | | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 2 | 3 | | 1 | 1 | 0 | 1 | 2 | | 2 | 2 | 1 | 1 | 2 | | 3 | 3 | - | - | - |   **Iteration 3 (i=3, s1="cat"):**   * j=1, s2="c" → **Delete 'at'** → dp[3][1] = min(Replace:2, Delete:1, Insert:3) + 1 = 2 * j=2, s2="cu" → **Delete 't'** → dp[3][2] = min(Replace:1, Delete:1, Insert:2) + 1 = 2 * j=3, s2="cut" → **Replace 'a' with 'u'** → dp[3][3] = dp[2][2] = 1 (since 'c' and 't' match)  | **i\j** | **0** | **1** | **2** | **3** | | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 2 | 3 | | 1 | 1 | 0 | 1 | 2 | | 2 | 2 | 1 | 1 | 2 | | 3 | 3 | 2 | 2 | 1 |   **Step 3: Output the Result**  ✅ The **minimum edit distance** is **dp[3][3] = 1**, meaning we need **one operation (replace 'a' with 'u')** to convert "cat" to "cut". |
| Output:- 1 | |