|  |  |
| --- | --- |
| **Floyd-Warshall in C++** | |
| #include <bits/stdc++.h>  using namespace std;  class Solution {  public:      void shortest\_distance(vector<vector<int>>&matrix) {          int n = matrix.size();          for (int i = 0; i < n; i++) {              for (int j = 0; j < n; j++) {                  if (matrix[i][j] == -1) {                      matrix[i][j] = 1e9;                  }                  if (i == j) matrix[i][j] = 0;              }          }          for (int k = 0; k < n; k++) {              for (int i = 0; i < n; i++) {                  for (int j = 0; j < n; j++) {                      matrix[i][j] = min(matrix[i][j],                                         matrix[i][k] + matrix[k][j]);                  }              }          }            for (int i = 0; i < n; i++) {              for (int j = 0; j < n; j++) {                  if (matrix[i][j] == 1e9) {                      matrix[i][j] = -1;                  }              }          }      }  };  int main() {      int V = 4;      vector<vector<int>> matrix(V, vector<int>(V, -1));      matrix[0][1] = 2;      matrix[1][0] = 1;      matrix[1][2] = 3;      matrix[3][0] = 3;      matrix[3][1] = 5;      matrix[3][2] = 4;      Solution obj;      obj.shortest\_distance(matrix);      for (auto row : matrix) {          for (auto cell : row) {              cout << cell << " ";          }          cout << endl;      }      return 0;  } | **Dry Run:**  **Input Matrix:**  The input adjacency matrix is:  matrix = [  [0, 2, -1, -1],  [1, 0, 3, -1],  [-1, -1, 0, -1],  [3, 5, 4, 0]  ]  **Step 1: Initialize the matrix**  Replace -1 with 1e9 and set matrix[i][i] = 0 for all i:  matrix = [  [0, 2, 1e9, 1e9],  [1, 0, 3, 1e9],  [1e9, 1e9, 0, 1e9],  [3, 5, 4, 0]  ]  **Step 2: Floyd-Warshall Algorithm**  Iterate over each intermediate node k and update the matrix.   * For k = 0 (Intermediate node 0):   + Check each pair (i, j) and update the matrix.   + No changes to the matrix as no shorter paths through node 0 are found. * For k = 1 (Intermediate node 1):   + For each pair (i, j):     - Update matrix[0][2] to matrix[0][1] + matrix[1][2] = 2 + 3 = 5.     - Update matrix[2][3] to matrix[2][1] + matrix[1][3] = 1e9 + 1e9 = 1e9 (no update). * For k = 2 (Intermediate node 2):   + For each pair (i, j):     - No changes as there are no shorter paths through node 2. * For k = 3 (Intermediate node 3):   + For each pair (i, j):     - Update matrix[2][1] to matrix[2][3] + matrix[3][1] = 1e9 + 5 = 1e9 (no update).     - Update matrix[3][1] to matrix[3][3] + matrix[3][1] = 0 + 5 = 5 (no update).   **Step 3: Final Matrix:**  After the Floyd-Warshall algorithm finishes, the matrix is:  matrix = [  [0, 2, 5, 8],  [1, 0, 3, 6],  [6, 8, 0, 9],  [3, 5, 4, 0]  ]  **Step 4: Convert 1e9 back to -1:**  If matrix[i][j] == 1e9, set matrix[i][j] = -1.  Final output matrix:  matrix = [  [0, 2, 5, 8],  [1, 0, 3, 6],  [6, 8, 0, 9],  [3, 5, 4, 0]  ]  **Output:**  Copy code  0 2 5 8  1 0 3 6  6 8 0 9  3 5 4 0 |
| **Output:-**  **0 2 5 -1**  **1 0 3 -1**  **-1 -1 0 -1**  **3 5 4 0** | |