28. Floyds Algorithm

#include <stdio.h>

#include <limits.h>

#define MAX\_N 100

void floydWarshall(int graph[][MAX\_N], int n) {

int dist[MAX\_N][MAX\_N], next[MAX\_N][MAX\_N];

// Initialize distance and next matrices

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

dist[i][j] = graph[i][j];

next[i][j] = (i == j) ? i : -1; // Initialize next to itself for diagonal elements

}

}

// Floyd-Warshall algorithm for all-pairs shortest paths

for (int k = 0; k < n; k++) {

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (dist[i][k] != INT\_MAX && dist[k][j] != INT\_MAX && dist[i][k] + dist[k][j] < dist[i][j]) {

dist[i][j] = dist[i][k] + dist[k][j];

next[i][j] = next[k][j]; // Update next node for shortest path

}

}

}

}

// Check for negative cycles

for (int i = 0; i < n; i++) {

if (dist[i][i] < 0) {

printf("Graph contains a negative cycle.\n");

return;

}

}

// Print the shortest distances

printf("Shortest distances between all pairs:\n");

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (dist[i][j] == INT\_MAX) {

printf("INF "); // Print "INF" for unreachable nodes

} else {

printf("%d ", dist[i][j]);

}

}

printf("\n");

}

// Optionally, print the next hop table for reconstructing paths (if needed)

printf("\nNext hop table:\n");

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

printf("%d ", next[i][j]);

}

printf("\n");

}

}

int main() {

int graph[MAX\_N][MAX\_N] = {

// Example adjacency matrix (modify as needed)

{0, 3, INF, 7},

{INF, 0, 4, 1},

{1, INF, 0, 5},

{2, INF, INF, 0}

};

int n = sizeof(graph[0]) / sizeof(graph[0][0]); // Number of vertices

floydWarshall(graph, n);

return 0;

}

