Metro Maze

PROJECT REPORT

Objective

PROJECT THE "METRO MAZE" PROJECT REPORT'S MAIN GOAL IS TO ASSESS, CREATE, AND PUT INTO PRACTICE AN INTELLIGENT SYSTEM THAT CAN IDENTIFY THE FASTEST AND MOST DIRECT ROUTE BETWEEN TWO METRO STATIONS. BY GIVING METRO PASSENGERS ACCESS TO REAL-TIME INFORMATION ON THE BEST ROUTE THAT MINIMIZES TRIP TIME AND DISTANCE, THIS TECHNOLOGY SEEKS TO IMPROVE THEIR COMMUTER EXPERIENCE.

PRINCIPAL GOALS:

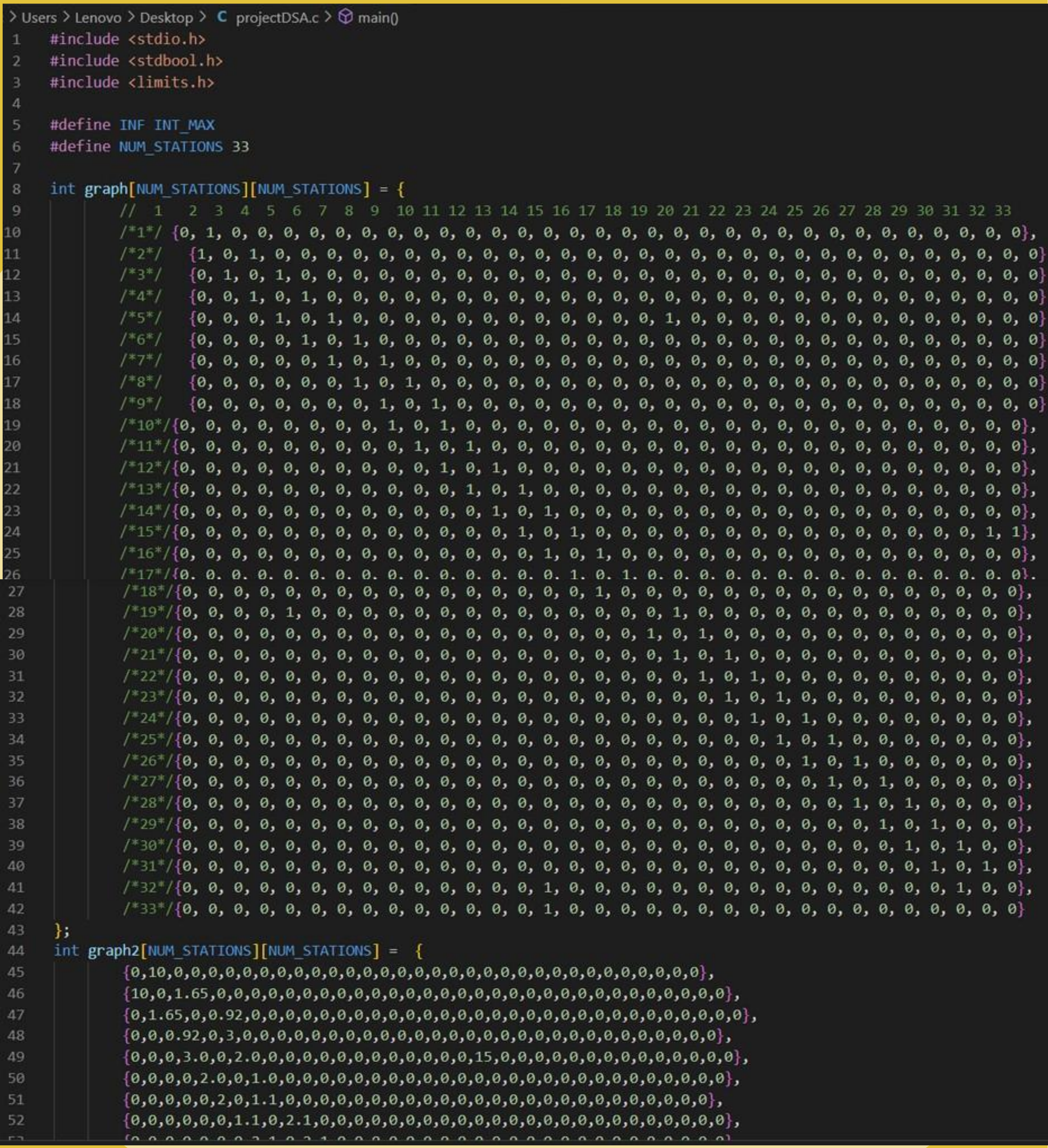
SHORTEST PATH IDENTIFICATION: WITHIN THE SPECIFIED METRO NETWORK, CREATE ALGORITHMS AND TECHNIQUES TO DETERMINE THE SHORTEST PATH BETWEEN TWO METRO STATIONS.

TIME-EFFICIENCY ANALYSIS: USE REAL-TIME DATA TO EXAMINE THE TRAFFIC SITUATION AT THE MOMENT, THE AMOUNT OF CONGESTION AT THE STATION, AND OTHER PERTINENT ELEMENTS AFFECTING JOURNEY TIME. GIVE USERS THE FASTEST ROUTE POSSIBLE DEPENDING ON CHANGING CIRCUMSTANCES.

DISTANCE OPTIMIZATION: USE ALGORITHMS TO DETERMINE THE FASTEST PHYSICAL PATH BETWEEN TWO METRO STATIONS, MAKING SURE THE ROUTE SELECTED MINIMIZES THE TOTAL JOURNEY DISTANCE WHILE SIMULTANEOUSLY SAVING TIME.

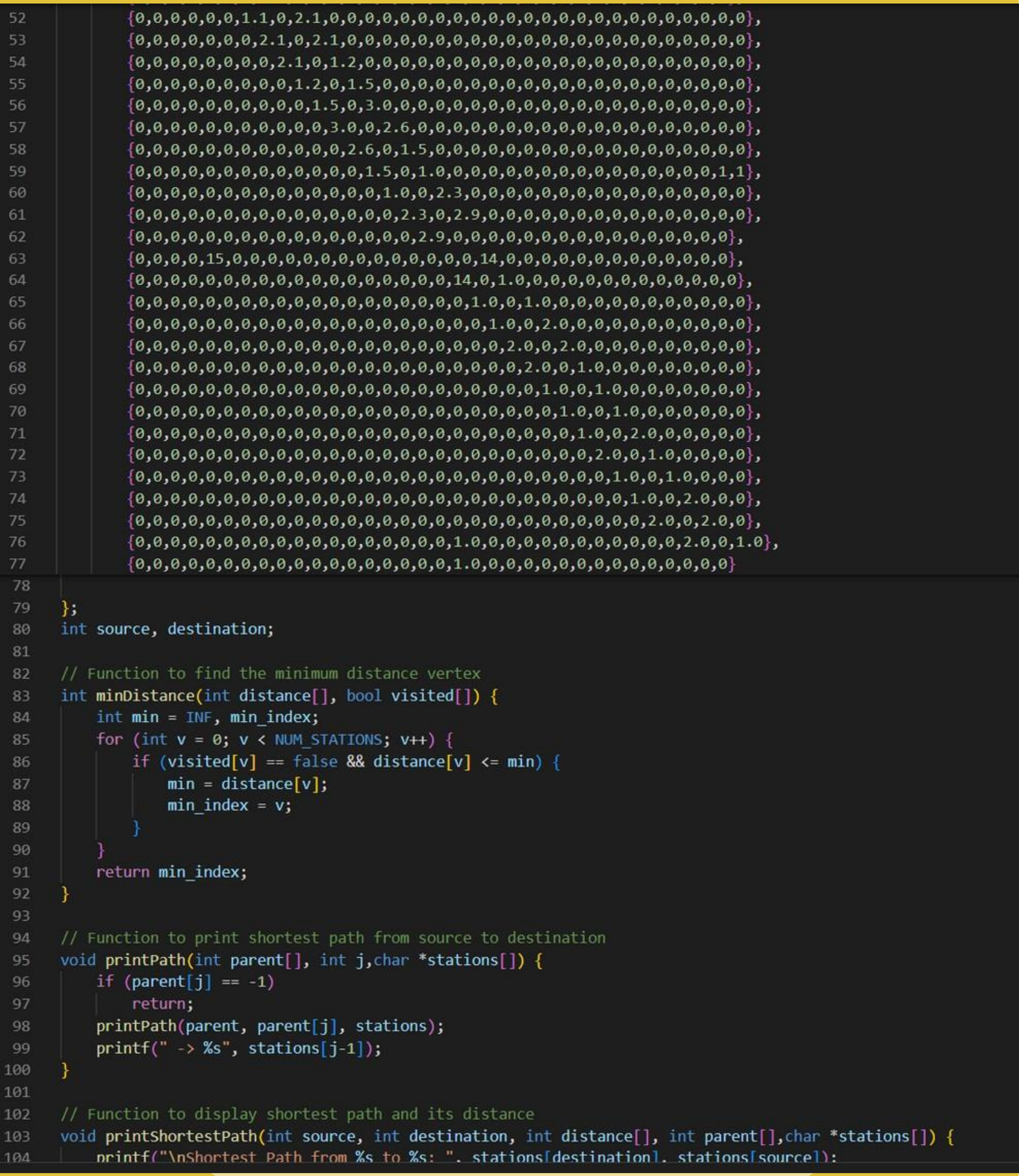
DATA VISUALIZATION: UTILIZE GRAPHICAL DISPLAYS, INCLUDING GRAPHS AND MAPS, TO SHOW THE SUGGESTED PATH, OTHER OPTIONS, AND RELATED TIME AND DISTANCE MEASUREMENTS. IMPROVE USER COMPREHENSION AND INTERACTION USING DATA VISUALIZATION.

Source Code



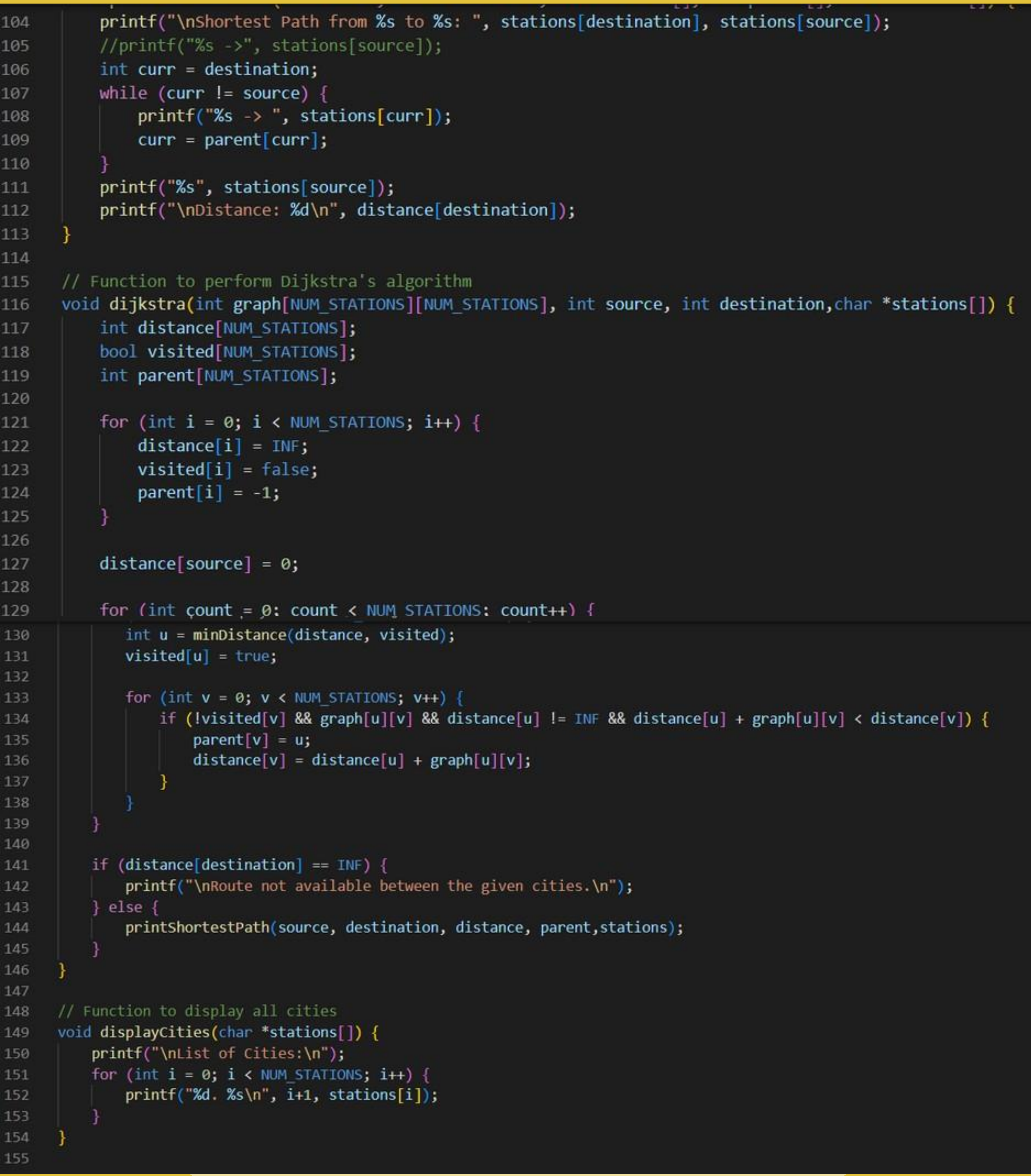
```c
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>

#define INF INT_MAX
#define NUM_STATIONS 33

int graph[NUM_STATIONS][NUM_STATIONS] = {
        //  1   2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
        /*1*/ {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*2*/   {1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
        /*3*/   {0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
        /*4*/   {0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
        /*5*/   {0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
        /*6*/   {0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
        /*7*/   {0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
        /*8*/   {0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
        /*9*/   {0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
        /*10*/{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*11*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*12*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*13*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*14*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*15*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1},
        /*16*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*17*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*18*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*19*/{0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*20*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*21*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*22*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*23*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        /*24*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
        /*25*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0},
        /*26*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0},
        /*27*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0},
        /*28*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0},
        /*29*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0},
        /*30*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0},
        /*31*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0},
        /*32*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0},
        /*33*/{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
};
int graph2[NUM_STATIONS][NUM_STATIONS] =  {
        {0,10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {10,0,1.65,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,1.65,0,0.92,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0.92,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,3.0,0,2.0,0,0,0,0,0,0,0,0,0,0,0,0,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,2.0,0,1.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,2,0,1.1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,1.1,0,2.1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,2.1,0,2.1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
```

Source Code



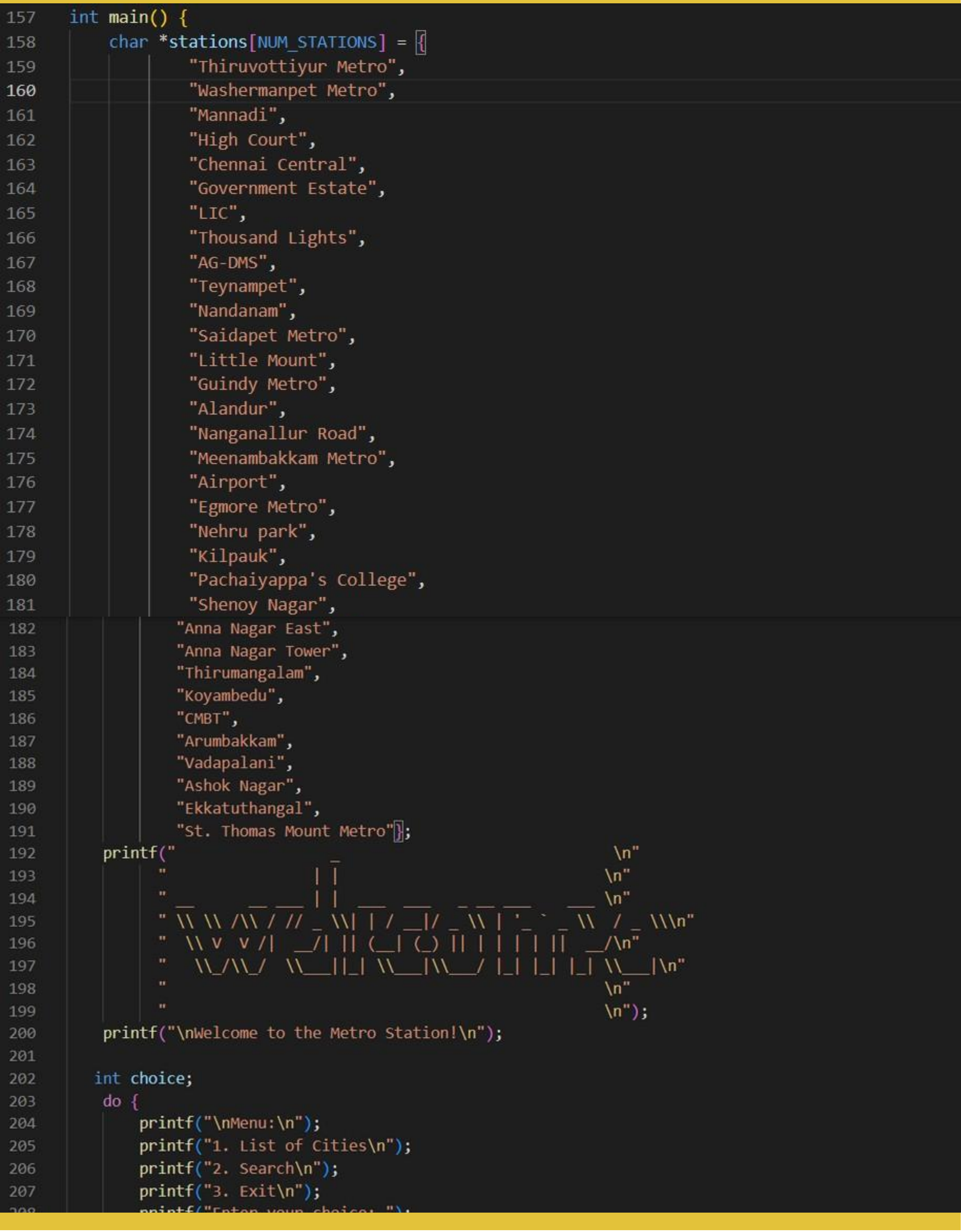
```
        {0,0,0,0,0,0,1.1,0,2.1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,2.1,0,2.1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,2.1,0,1.2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,1.2,0,1.5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,1.5,0,3.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,3.0,0,2.6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,2.6,0,1.5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,1.5,0,1.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.0,0,2.3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2.3,0,2.9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2.9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0,14,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,14,0,1.0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.0,0,1.0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.0,0,2.0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2.0,0,2.0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2.0,0,1.0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.0,0,1.0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.0,0,1.0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.0,0,2.0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2.0,0,1.0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.0,0,1.0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.0,0,2.0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2.0,0,2.0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.0,0,0,0,0,0,0,0,0,0,0,0,2.0,0,1.0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}

};
int source, destination;

// Function to find the minimum distance vertex
int minDistance(int distance[], bool visited[]) {
    int min = INF, min_index;
    for (int v = 0; v < NUM_STATIONS; v++) {
        if (visited[v] == false && distance[v] <= min) {
            min = distance[v];
            min_index = v;
        }
    }
    return min_index;
}

// Function to print shortest path from source to destination
void printPath(int parent[], int j,char *stations[]) {
    if (parent[j] == -1)
        return;
    printPath(parent, parent[j], stations);
    printf(" -> %s", stations[j-1]);
}

// Function to display shortest path and its distance
void printShortestPath(int source, int destination, int distance[], int parent[],char *stations[]) {
    printf("\nShortest Path from %s to %s: ", stations[destination], stations[source]);
```

Source Code



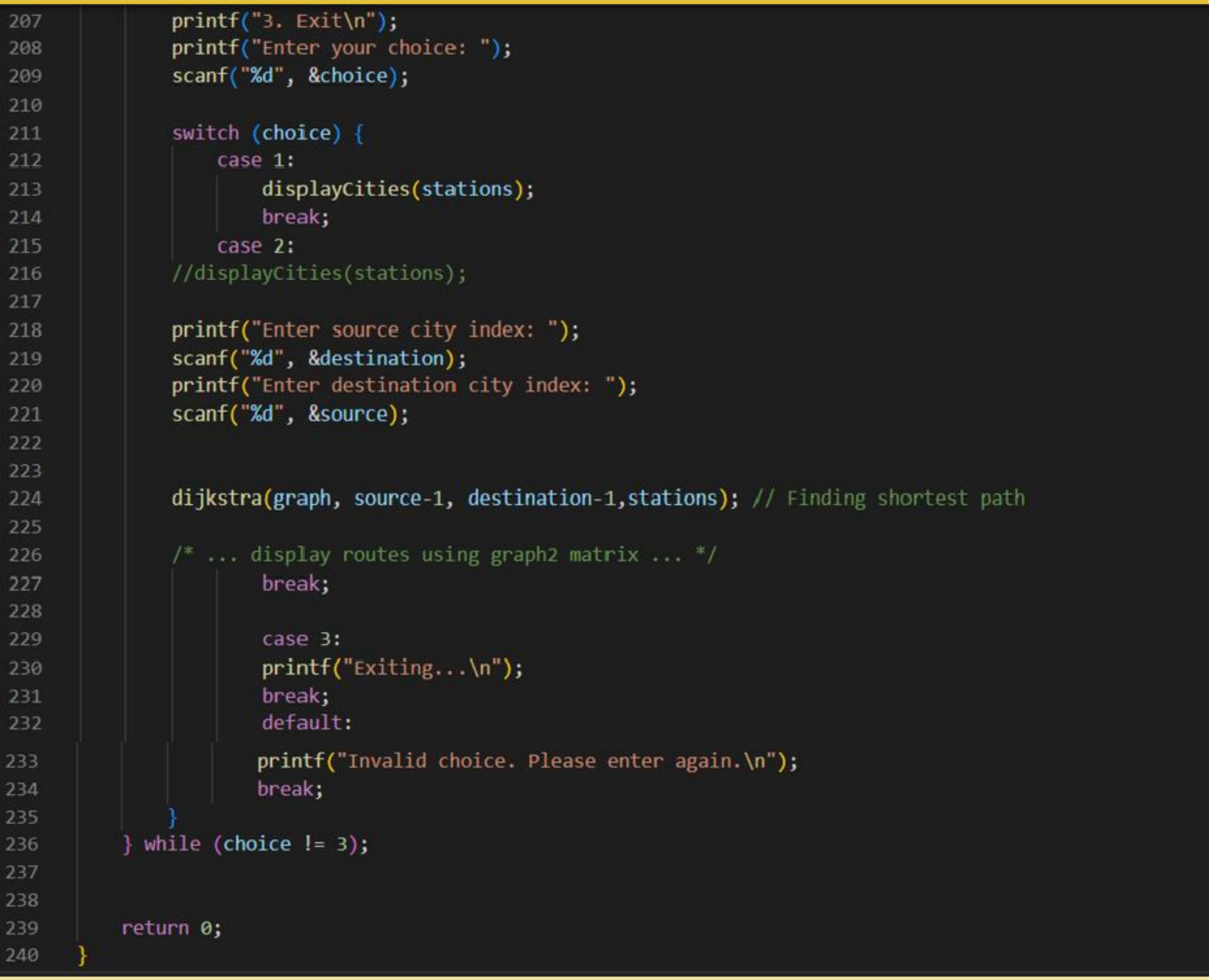
```c
    printf("\nShortest Path from %s to %s: ", stations[destination], stations[source]);
    //printf("%s ->", stations[source]);
    int curr = destination;
    while (curr != source) {
        printf("%s -> ", stations[curr]);
        curr = parent[curr];
    }
    printf("%s", stations[source]);
    printf("\nDistance: %d\n", distance[destination]);
}

// Function to perform Dijkstra's algorithm
void dijkstra(int graph[NUM_STATIONS][NUM_STATIONS], int source, int destination,char *stations[]) {
    int distance[NUM_STATIONS];
    bool visited[NUM_STATIONS];
    int parent[NUM_STATIONS];

    for (int i = 0; i < NUM_STATIONS; i++) {
        distance[i] = INF;
        visited[i] = false;
        parent[i] = -1;
    }

    distance[source] = 0;

    for (int count = 0; count < NUM_STATIONS; count++) {
        int u = minDistance(distance, visited);
        visited[u] = true;

        for (int v = 0; v < NUM_STATIONS; v++) {
            if (!visited[v] && graph[u][v] && distance[u] != INF && distance[u] + graph[u][v] < distance[v]) {
                parent[v] = u;
                distance[v] = distance[u] + graph[u][v];
            }
        }
    }

    if (distance[destination] == INF) {
        printf("\nRoute not available between the given cities.\n");
    } else {
        printShortestPath(source, destination, distance, parent,stations);
    }
}

// Function to display all cities
void displayCities(char *stations[]) {
    printf("\nList of Cities:\n");
    for (int i = 0; i < NUM_STATIONS; i++) {
        printf("%d. %s\n", i+1, stations[i]);
    }
}
```

Source Code



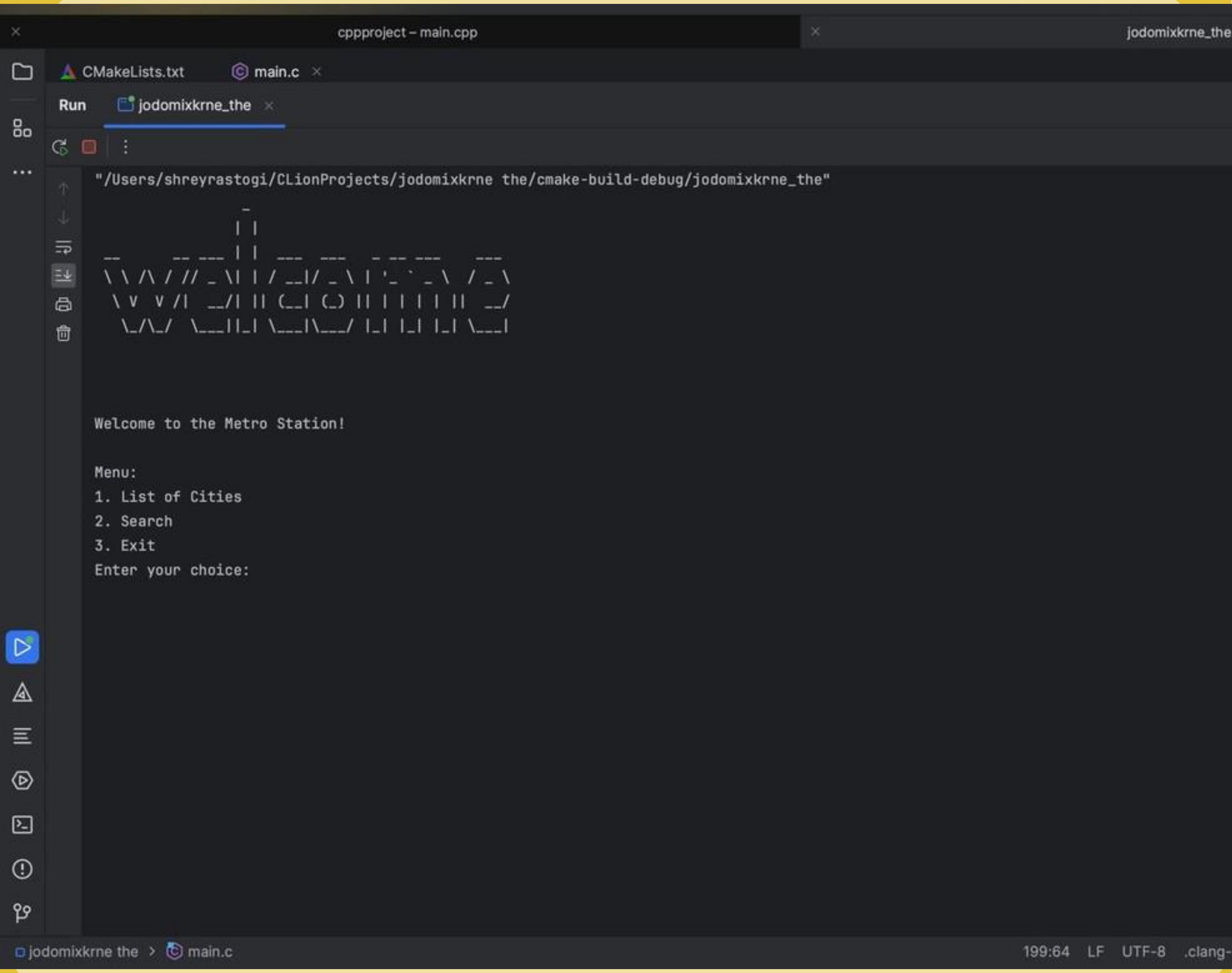
```c
int main() {
    char *stations[NUM_STATIONS] = {
            "Thiruvottiyur Metro",
            "Washermanpet Metro",
            "Mannadi",
            "High Court",
            "Chennai Central",
            "Government Estate",
            "LIC",
            "Thousand Lights",
            "AG-DMS",
            "Teynampet",
            "Nandanam",
            "Saidapet Metro",
            "Little Mount",
            "Guindy Metro",
            "Alandur",
            "Nanganallur Road",
            "Meenambakkam Metro",
            "Airport",
            "Egmore Metro",
            "Nehru park",
            "Kilpauk",
            "Pachaiyappa's College",
            "Shenoy Nagar",
            "Anna Nagar East",
            "Anna Nagar Tower",
            "Thirumangalam",
            "Koyambedu",
            "CMBT",
            "Arumbakkam",
            "Vadapalani",
            "Ashok Nagar",
            "Ekkatuthangal",
            "St. Thomas Mount Metro"};
    printf("                _                            \n"
        "               | |                           \n"
        " __      _____| | ___ ___  _ __ ___   ___  \n"
        " \\ \\ /\\ / // _ \\| | / __|/ _ \\ | '_ ` _ \\  / _ \\\n"
        "  \\ V  V /|  __/| || (__| (_) || | | | | ||  __/\n"
        "   \\_/\\_/  \\___||_| \\___|\\___/ |_| |_| |_| \\___|\n"
        "                                             \n"
        "                                             \n");
    printf("\nWelcome to the Metro Station!\n");

    int choice;
    do {
        printf("\nMenu:\n");
        printf("1. List of Cities\n");
        printf("2. Search\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
```

Source Code



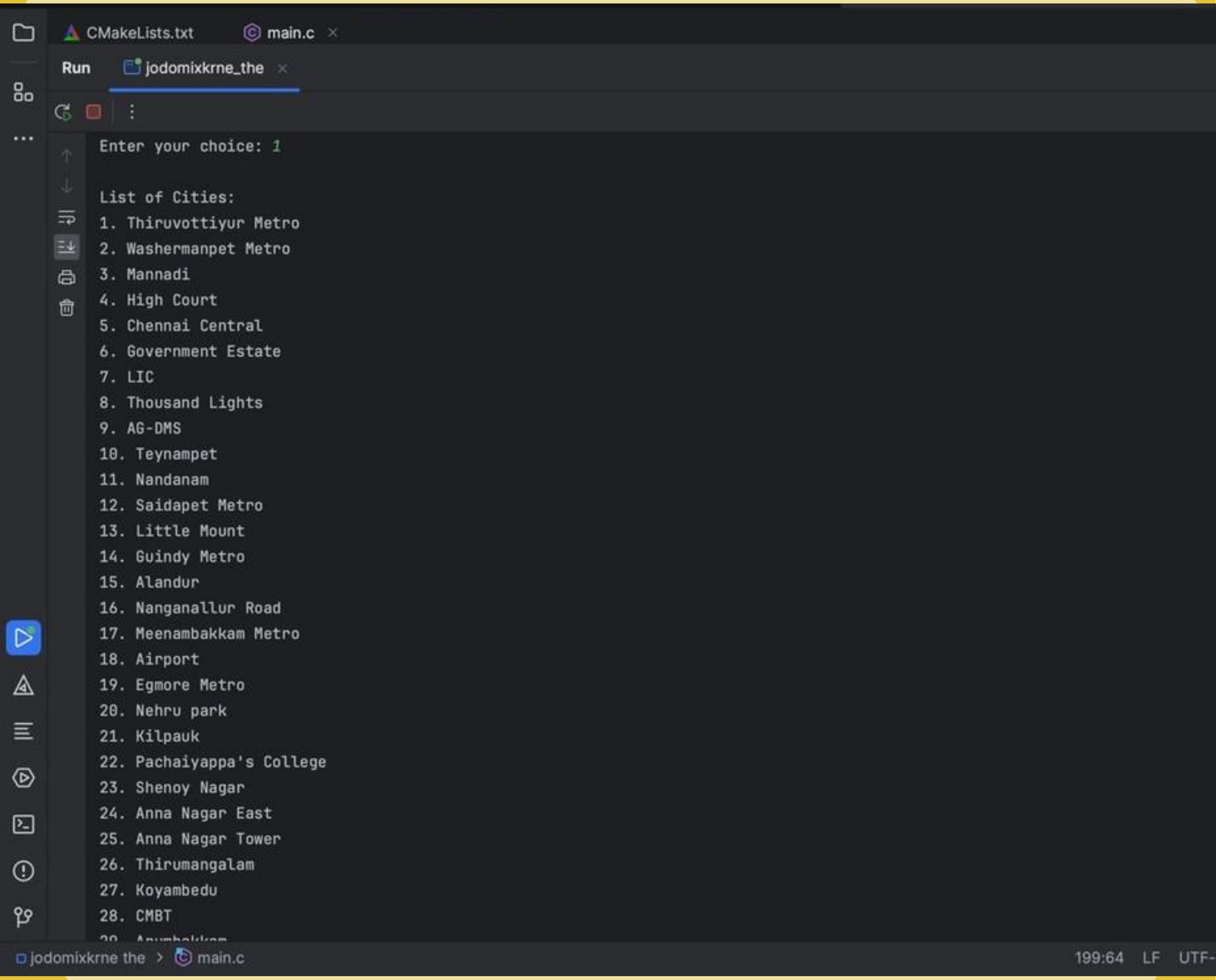
```c
            printf("3. Exit\n");
            printf("Enter your choice: ");
            scanf("%d", &choice);

            switch (choice) {
                case 1:
                    displayCities(stations);
                    break;
                case 2:
            //displayCities(stations);

            printf("Enter source city index: ");
            scanf("%d", &destination);
            printf("Enter destination city index: ");
            scanf("%d", &source);


            dijkstra(graph, source-1, destination-1,stations); // Finding shortest path

            /* ... display routes using graph2 matrix ... */
                    break;

                    case 3:
                    printf("Exiting...\n");
                    break;
                    default:
                    printf("Invalid choice. Please enter again.\n");
                    break;
            }
        } while (choice != 3);


        return 0;
    }
```

Output



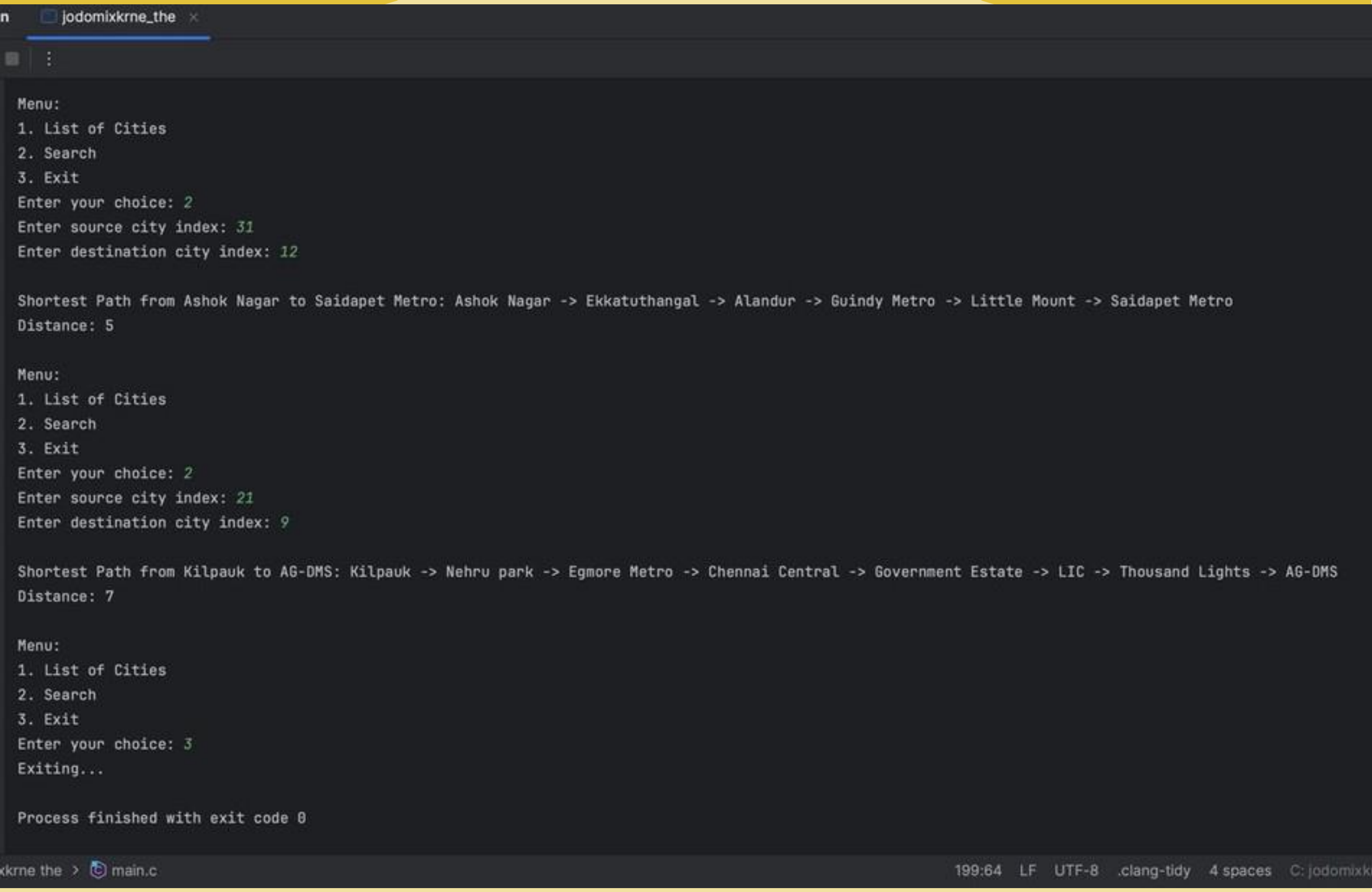
```
"/Users/shreyrastogi/CLionProjects/jodomixkrne the/cmake-build-debug/jodomixkrne_the"
                 _
                | |
 __      __ ___ | |  ___  ___   _ __ ___    ___
 \ \ /\ / // _ \| | / __|/ _ \ | '_ ` _ \  / _ \
  \ V  V /|  __/| || (__| (_) || | | | | ||  __/
   \_/\_/  \___||_| \___|\___/ |_| |_| |_| \___|



Welcome to the Metro Station!

Menu:
1. List of Cities
2. Search
3. Exit
Enter your choice:
```

Output



```
Enter your choice: 1

List of Cities:
1. Thiruvottiyur Metro
2. Washermanpet Metro
3. Mannadi
4. High Court
5. Chennai Central
6. Government Estate
7. LIC
8. Thousand Lights
9. AG-DMS
10. Teynampet
11. Nandanam
12. Saidapet Metro
13. Little Mount
14. Guindy Metro
15. Alandur
16. Nanganallur Road
17. Meenambakkam Metro
18. Airport
19. Egmore Metro
20. Nehru park
21. Kilpauk
22. Pachaiyappa's College
23. Shenoy Nagar
24. Anna Nagar East
25. Anna Nagar Tower
26. Thirumangalam
27. Koyambedu
28. CMBT
29. Anumbakkam
```

Output



```
Menu:
1. List of Cities
2. Search
3. Exit
Enter your choice: 2
Enter source city index: 31
Enter destination city index: 12

Shortest Path from Ashok Nagar to Saidapet Metro: Ashok Nagar -> Ekkatuthangal -> Alandur -> Guindy Metro -> Little Mount -> Saidapet Metro
Distance: 5

Menu:
1. List of Cities
2. Search
3. Exit
Enter your choice: 2
Enter source city index: 21
Enter destination city index: 9

Shortest Path from Kilpauk to AG-DMS: Kilpauk -> Nehru park -> Egmore Metro -> Chennai Central -> Government Estate -> LIC -> Thousand Lights -> AG-DMS
Distance: 7

Menu:
1. List of Cities
2. Search
3. Exit
Enter your choice: 3
Exiting...

Process finished with exit code 0
```

Conclusion

By determining the quickest route between two stations and reducing trip time, the Metro Maze report provides insightful information on maximizing transit.

The need for metro network optimization grows as cities expand and put more strain on their public transportation systems. The knowledge gathered from this paper highlights how further progress in infrastructure development, route planning, and technology integration may improve metro systems all around the world.

As a result of the report's findings, the current metro maze is made more efficient and opens the door for future transportation advancements and breakthroughs.

To sum up, the Metro Maze report is an invaluable tool for urban planners, transportation authorities, and commuters alike. It provides a road map for maximizing movement inside metro networks and promoting a more efficient, convenient, and pleasurable passenger experience.

Reference

1) \Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269–271.

2) Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

3) GeeksforGeeks. (n.d.). Dijkstra's Shortest Path Algorithm. Retrieved from https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

4) Programiz-https://www.programiz.com/