**IT300 – Design and Analysis of Algorithms**

Lab Assignment – 4

**Name:** Niraj Nandish

**Roll No:** 191IT234

* Program 1 – Currency Conversion Problem
  + Code

#include <iomanip>

#include <iostream>

#include <limits.h>

#include <map>

#include <math.h>

#include <stdio.h>

#include <vector>

using namespace std;

map<double, *string*> ma;

struct *Edge*

{

double source, dest, act\_cost, weight;

};

void printPath(vector<double> const &*parent*, double *vertex*, double *d*)

{

if (*vertex* < 0)

{

return;

}

printPath(*parent*, *parent*[*vertex*], *d*);

if (*vertex* != *d*)

{

cout << ma[*vertex*] << " -> ";

}

else

{

cout << ma[*vertex*];

}

}

void bellmanFord(vector<*Edge*> const &*edges*, double *source*, double *destination*, double *N*)

{

double u, v, w, k = *N*;

vector<double> distance(*N*, INT\_MAX);

vector<double> parent(*N*, -1);

distance[*source*] = 0;

while (--k)

{

for (*Edge* edge : *edges*)

{

u = edge.source;

v = edge.dest;

w = edge.weight;

if (distance[u] != INT\_MAX && distance[u] + w < distance[v])

{

distance[v] = distance[u] + w;

parent[v] = u;

}

}

}

for (*Edge* edge : *edges*)

{

u = edge.source;

v = edge.dest;

w = edge.weight;

if (distance[u] != INT\_MAX && distance[u] + w < distance[v])

{

cout << "Negative-weight cycle is found!!";

return;

}

}

cout << "\n1 " << ma[*source*] << " = " << pow(10, -1 \* distance[*destination*]) << " " << ma[*destination*] << "\nIts path of conversion is ";

printPath(parent, *destination*, *destination*);

cout << endl;

}

int main()

{

double N = 10;

vector<*Edge*> edges;

ma[0] = "Gold";

ma[1] = "US Dollar";

ma[2] = "Swiss Franc";

ma[3] = "Japanese Yen";

ma[4] = "Euro";

ma[5] = "UK Pound";

ma[6] = "Canadian Dollar";

ma[7] = "Mexican Peso";

ma[8] = "New Zealand Dollar";

ma[9] = "Pakistani Rupee";

edges.push\_back({0, 1, 327.25, (double)(-1 \* log10(327.25))});

edges.push\_back({0, 2, 455.2, (double)(-1 \* log10(455.2))});

edges.push\_back({0, 5, 208.1, (double)(-1 \* log10(208.1))});

edges.push\_back({2, 1, 0.7182, (double)(-1 \* log10(0.7182))});

edges.push\_back({2, 4, 0.6677, (double)(-1 \* log10(0.6677))});

edges.push\_back({3, 1, 0.008309, (double)(-1 \* log10(0.008309))});

edges.push\_back({4, 1, 1.0752, (double)(-1 \* log10(1.0752))});

edges.push\_back({4, 3, 129.52, (double)(-1 \* log10(129.52))});

edges.push\_back({5, 2, 2.1904, (double)(-1 \* log10(2.1904))});

edges.push\_back({1, 7, 20.58, (double)(-1 \* log10(20.58))});

edges.push\_back({1, 8, 1.45, (double)(-1 \* log10(1.45))});

edges.push\_back({7, 8, 0.07, (double)(-1 \* log10(0.07))});

edges.push\_back({4, 6, 1.47, (double)(-1 \* log10(1.47))});

edges.push\_back({6, 8, 1.14, (double)(-1 \* log10(1.14))});

edges.push\_back({6, 9, 134.10, (double)(-1 \* log10(134.10))});

double source, destination;

cout << "Currencies available are: \n1.Gold\n2.US Dollar\n3.Swiss Franc\n4.Japanese Yen\n5.Euro\n6.UK Pound\n7.Canadian Dollar\n8.Mexican Peso\n9.New Zealand Dollar\n10.Pakistani Rupee";

cout << "\n\nEnter input currency option number: ";

cin >> source;

cout << "Enter output currency option number: ";

cin >> destination;

bellmanFord(edges, source - 1, destination - 1, N);

return 0;

}

* + Graphical user interface, text

    Description automatically generatedScreenshot
* Program 2 – IP Routing Problem
  + Code

#include <iostream>

#include <limits.h>

#include <stdio.h>

using namespace std;

int V, no\_E, startNode;

// V: Number of vertices from user

// no\_E: Number of edges from user

// startNode: Start Dijkstra at inputted Vertex

void dijkstra(int *adj\_mat*[][50], int *start*, int *dest*)

{

int visited[V], path[V], distance[V];

for (int i = 0; i < V; i++)

{

visited[i] = 0;

distance[i] = INT\_MAX;

}

distance[*start*] = 0;

for (int count = 0; count < V; count++)

{

int min = INT\_MAX;

int pos = 0;

for (int i = 0; i < V; i++)

{

if (visited[i] == 0)

{

if (distance[i] < min)

{

min = distance[i];

pos = i;

}

}

}

visited[pos] = 1;

for (int j = 0; j < V; j++)

{

if (*adj\_mat*[pos][j] && visited[j] == 0 && (distance[pos] + *adj\_mat*[pos][j]) < distance[j])

{

distance[j] = distance[pos] + *adj\_mat*[pos][j];

path[j] = pos;

}

}

}

cout << "\nShortest cost from routers " << *start* << " to " << *dest* << ": " << distance[*dest*] << endl;

int tempArr[V];

int x = 0, t = *dest*;

tempArr[0] = *dest*;

x++;

do

{

tempArr[x] = path[t];

t = path[t];

x++;

} while (t != *start*);

cout << "Shortest path from routers " << *start* << " to " << *dest* << ": ";

cout << *start*;

for (int z = x - 2; z >= 0; z--)

{

cout << " -> ";

cout << tempArr[z];

}

cout << endl;

}

void dijkstraToAll(int *adj\_mat*[][50], int *start*)

{

cout << "\n----------------------------------------------\n";

for (int i = 0; i < V; i++)

{

if (i != *start*)

{

dijkstra(*adj\_mat*, *start*, i);

cout << endl;

}

}

}

int main()

{

int v1, v2, weight;

int i;

cout << "Enter the number of routers(max 50): ";

cin >> V;

cout << "Enter the number of connections: ";

cin >> no\_E;

if (V > 50 || V <= 0 || no\_E <= 0)

{

cout << "\nNo routers present or no connections present.\n";

return 0;

}

if (V == 1)

{

cout << "\nOnly one router present in network.\n";

return 0;

}

int adj\_matrix[50][50];

for (i = 0; i < V; i++)

{

for (int j = 0; j < V; j++)

{

adj\_matrix[i][j] = 0;

}

}

cout << "\nEnter router number between 0 and " << V - 1;

for (i = 1; i <= no\_E; i++)

{

cout << "\nConnection " << i << ":\n";

cin >> v1 >> v2;

cout << "Enter cost: ";

cin >> weight;

adj\_matrix[v1][v2] = weight;

adj\_matrix[v2][v1] = weight;

}

do

{

cout << "\nEnter source router number: ";

cin >> startNode;

} while (startNode < 0 || startNode >= V);

dijkstraToAll(adj\_matrix, startNode);

return 0;

}

* + Text

    Description automatically generatedText

    Description automatically generated with medium confidenceScreenshots