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";
```

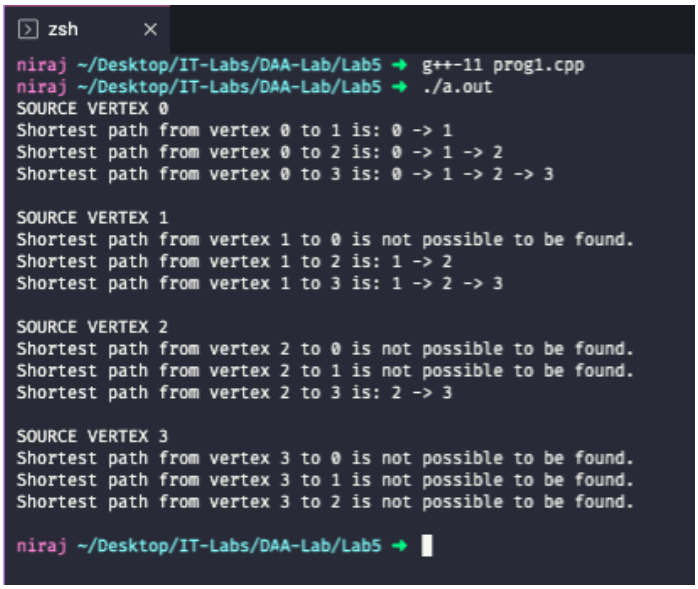
```
  ma[9] = "Pakistani Rupee";

  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;
}
```

- Screenshot



- 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;
}
```

- Screenshots

```
zsh
niraj ~/Desktop/IT-Labs/DAA-Lab/Lab4 ➜ g++-11 prog2.cpp
niraj ~/Desktop/IT-Labs/DAA-Lab/Lab4 ➜ ./a.out
Enter the number of routers(max 50): 9
Enter the number of connections: 14

Enter router number between 0 and 8
Connection 1:
0
1
Enter cost: 4

Connection 2:
1
2
Enter cost: 8

Connection 3:
2
3
Enter cost: 7

Connection 4:
3
4
Enter cost: 9

Connection 5:
4
5
Enter cost: 10

Connection 6:
5
6
Enter cost: 2

Connection 7:
6
7
Enter cost: 1

Connection 8:
7
0
Enter cost: 8

Connection 9:
1
7
Enter cost: 11

Connection 10:
7
8
Enter cost: 7

Connection 11:
2
8
Enter cost: 2

Connection 12:
8
6
Enter cost: 6

Connection 13:
2
5
Enter cost: 4

Connection 14:
3
5
Enter cost: 14
```

```
Enter source router number: 0

-----------------------------------------

Shortest cost from routers 0 to 1: 4
Shortest path from routers 0 to 1: 0 -> 1


Shortest cost from routers 0 to 2: 12
Shortest path from routers 0 to 2: 0 -> 1 -> 2


Shortest cost from routers 0 to 3: 19
Shortest path from routers 0 to 3: 0 -> 1 -> 2 -> 3


Shortest cost from routers 0 to 4: 21
Shortest path from routers 0 to 4: 0 -> 7 -> 6 -> 5 -> 4


Shortest cost from routers 0 to 5: 11
Shortest path from routers 0 to 5: 0 -> 7 -> 6 -> 5


Shortest cost from routers 0 to 6: 9
Shortest path from routers 0 to 6: 0 -> 7 -> 6


Shortest cost from routers 0 to 7: 8
Shortest path from routers 0 to 7: 0 -> 7


Shortest cost from routers 0 to 8: 14
Shortest path from routers 0 to 8: 0 -> 1 -> 2 -> 8
```