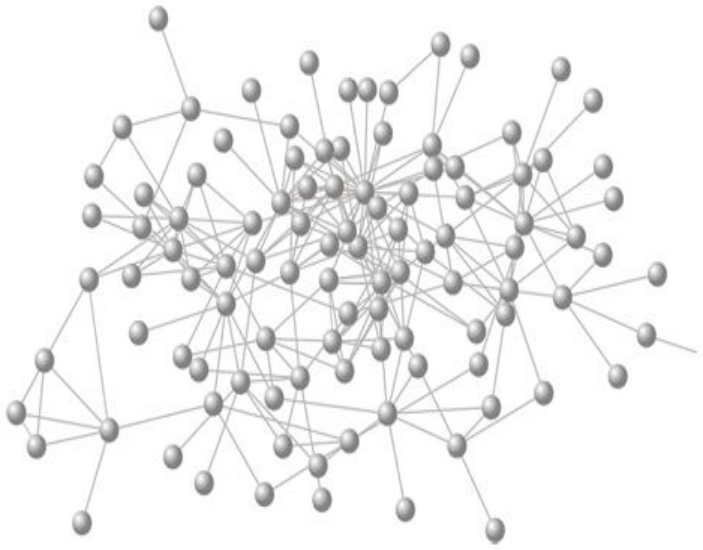
CSCE 210 Fall 2015
Instructor: Dr. Amr Goneid
Assignment # 6 (Group Assignment)
Date : Mon Dec 7 Due: Thu Dec 17, 2015

Implement a program to solve the following Graph problems:

Problem (1): Finding all shortest paths in a weighted graph

A network of highways connects N cities. Such network can be represented by a graph with N nodes and the weights on the edges represent the distances between pairs of cities. A text file "**CitiesG.txt**" gives the adjacency matrix representation for such weighted graph for N = 14 cities. An Excel sheet representation is also given in **"CitiesG.xlsx"**. The graph is connected and the weights are all positive integers in the range 20 – 100. Zero weights represent the absence of a highway, or the distance between a city and itself. The cities are simply named (A, B, C,….). The first number in the text file is the number of cities (14). The following 14 numbers represent the first row in the adjacency matrix, followed by 14 numbers representing the second row, and so on (see assignment 1).



Required Implementation:

Develop a program to:

1. Traverse the given graph using the ***Breadth-First Search (BFS)*** algorithm.
2. Determine the shortest paths from a given city (e.g., A) to all the other cities in the given graph using **Dijkstra's** algorithm.

Problem (2) : Reachability Matrix for a Graph

Consider the graph of Problem (1) that represents a network of highways connecting N cities. Assume that the weight W_{ij} of an edge between city (i) and city (j) represents the RISK of using the highway (i,j). We consider city (j) to be unreachable DIRECTLY from

city (i) if $W_{ij} = 0$ (there is no direct highway), or if the risk is too high (W_{ij} > say 50). Of course, city (j) might be reachable from city (i) INDIRECTLY via other cities or might be totally unreachable.
Under these conditions, our problem is to find a matrix $\mathbf{A}^*$ such that $\mathbf{A}^*(i,j) = 1$ iff there is a path from city (i) to city (j) and $\mathbf{A}^*(i,j) = 0$ otherwise.

Required Implementation:

- Implement an algorithm to generate the reachability matrix $\mathbf{A}^*$ for the graph of Problem (1) taking the weights to be too risky if $W_{ij} > T$, where T is an upper limit of the risk.
- Show results for $\mathbf{A}^*$ for different values of T to illustrate the validity of the algorithm used.

Notes on the design and implementation

- We use the adjacency matrix representation of the graph, i.e., a 2-D array of size V_{max} by V_{max} where V_{max} is the maximum number of vertices (e.g $V_{max} = 50$).
- The vertices are given numbers (0,1,….V-1) where V is the actual number of vertices in the graph as given in the file. These numbers can be mapped to names (e.g. A , B , C…).
- An edge (u,v,w) has one vertex u, a second vertex v and a weight w (a positive integer). It is represented as an object of a class "Edge". This class will also contain the definition of weight type (integer). A 1-D array of size E_{max} is used to store the non-zero edges in the graph, where E_{max} is the maximum number of possible edges = V_{max} *(V_{max} -1)/2. The actual number of such edges in the graph will be E so that the edges will be stored in locations (0 .. E-1).
- A class "***Graphs***" represents the above ADT. The "skeletal" class header and implementation files are given in files **"Graphs.h"** and **"Graphs.cpp"** in the zip file **"210as6F15.zip"**. Also included is the **"Edge.h"** file representing the "Edge" class. You should complete the implementation of the **"Graphs"** class.
- The zip file **"210as6F15.zip"** contains a test graph file **"TestG.txt"** representing a graph of 7 vertices, as well as the corresponding Excel file. You can test your implementation using this file. The sample output for that file is also given in file **"Sample.txt"** in the zip file.
- The zip file also contains the graph file **"CitiesG.txt"** that will be used in the assignment.