CS 202 Fundamental Structures of Computer Science II

Assignment 5

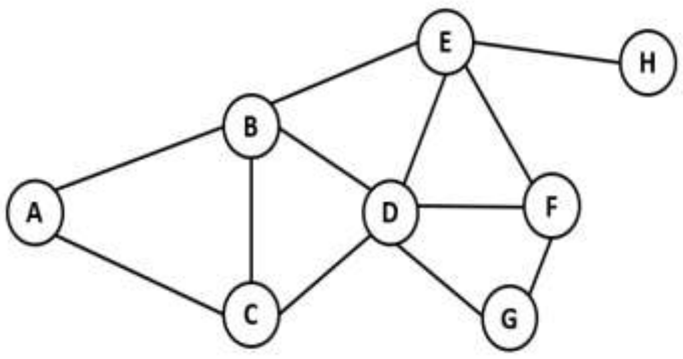
Date Assigned: December 11, 2015
Due Date: December 25, 2015 - 23:55 (sharp)

Question-1 (20 points)

For the graph given below, give the sequence of vertices when they are traversed starting from *vertex A* using

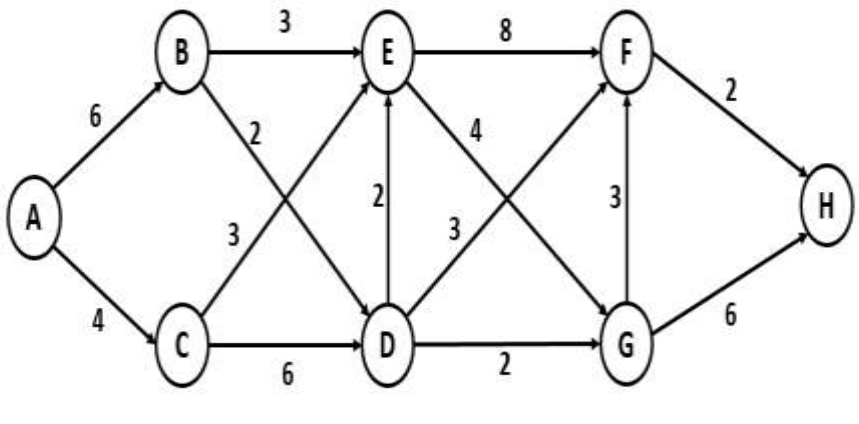
- The depth first traversal algorithm
- The breadth first traversal algorithm

In your solution, for a vertex, use the lexicographical order to visit its neighbors.



Question-2 (10 points)

For the following weighted directed graph, find the shortest paths from *vertex A* to all other vertices using Dijkstra's shortest path algorithm. Show all steps of Dijkstra's algorithm.



Question-3 (70 points)

Programming Assignment

In this question, you are asked to implement a simple flight route network which represents airport flight relationships. In this implementation, you will represent this airport network by a graph and answer the queries, each of which corresponds to calling a member function, on this graph. Draft of your public class **AirportNet** is given below and name of the class must be **AirportNet.** This class must include public member functions to conduct required tasks. These public member functions will be used to test your implementation.

```
#include <string>
using namespace std;
class AirportNet{
public:
      AirportNet(const string aname);
      AirportNet(const AirportNet& aNet); // copy contructor
      ~AirportNet(); // destructor
      void listDirectFlights(const string aname);
      void listRoutesOnHubAirport(const string aname);
      void findConnectedComponents();
      void displayMostCentralAirport();

      // ...
      // define other public member functions here,
      // if you have any
private:
      // ...
      // define your data members here
      // define private member functions here, if you have any

};
```

The interface for the class must be written in a file called **AirportNet.h** and its implementation must be written in a file called **AirportNet.cpp**. You can define additional public and private member functions and private data members in this class. You can also define additional classes in your solution. The details of the member functions are as follows:

- **AirportNet(const string anetname);**

  The default constructor loads an airport network from an input file called **anetname**. The first row of this file indicates the number of connections in the network. Subsequent lines of file include information of routes. Connections between airports are bidirectional, that is, if there is a flight from O to D, then there also exists a flight from D to O. Information in the lines contains *<origin> <destination> <distance>* tokens separated by white spaces. For a particular route R

  - **<origin>** is the starting airport of the route

- **<destination>** is the terminal airport of the route
- **<distance>** is the distance between origin and destination airports

In this assignment, you may assume that the contents of the input file are always valid. You may also assume that names of the airports are unique. Note that, if the input file **anetname** does not exist, then the default constructor creates an empty airport network.

The following table shows an example input file for the defined airport network. This file contains 43 connections, as indicated in its first line. For example, the fourth line of this file indicates that the route from *Istanbul* to *Trabzon* has *9* unit distance. This line also informs that there is route from *Trabzon* to *Istanbul* with *9* unit distance. The input file of the network illustrated below:

```
43
Istanbul Edremit 4
Istanbul Samsun 7
Istanbul Trabzon 9
Istanbul Erzincan 10
Istanbul Erzurum 12
Istanbul Van 14
Istanbul Mus 13
Istanbul Sirnak 16
Istanbul Batman 13
Istanbul Elazig 12
Istanbul Sivas 8
Istanbul Mardin 14
Istanbul Malatya 10
Istanbul Diyarbakir 12
Istanbul Kayseri 7
Istanbul Ankara 6
Istanbul Konya 6
Istanbul Antalya 8
Istanbul Izmir 6
Istanbul Denizli 7
Istanbul Nevsehir 8
Ankara Trabzon 8
Ankara Diyarbakir 8
Ankara Bodrum 7
Ankara Erzurum 9
Ankara Antalya 5
Ankara Adana 5
Izmir Samsun 10
Izmir Mardin 12
Izmir Diyarbakir 10
Izmir Elazig 9
Izmir Malatya 8
Izmir Hatay 7
Izmir Adana 6
Izmir Antalya 4
Adana Van 8
Adana Trabzon 10
Adana Bodrum 7
Adana Erzurum 9
Adana Antalya 4
Antalya Trabzon 11
Antalya Malatya 8
Antalya Diyarbakir 10
```

- **void listDirectFlights (const string aname);**

  It outputs the airports which have direct flight from airport “aname”. If this given airport does not take place in the airport network, give a warning message. See the output example below for the format. You may assume that the names are unique within the airport network.

- **void listRoutesOnHubAirport(const string aname);**

  It outputs the routes (<o, d> tuples) for which there exist no direct connection between <o> and <d>, but <d> can be reached from <o> using airport “aname” as a hub airport. If this given airport does not take place in the airport network, give a warning message. Similarly, you may assume that the names are unique within the airport network.

- **void findConnectedComponents();**

  This member function determines whether given graph has more than one connected components or not. If graph includes only one connected component than, you should give this as an information message, otherwise you should display number of nodes of connected components.

- **void displayMostCentralAirport();**

  It should calculate betweenness centrality score of each vertex in graph and it should output the most central airport which has the highest betweenness centrality score.

  “**Betweenness centrality** is an indicator of a node's centrality in a network. It is equal to the number of shortest paths from all vertices to all others that pass through that node” (from wikipedia).

  The betweenness centrality of vertex *v* can be calculated as follows.

$$bc(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

  where σ_{st} is the total number of shortest paths from vertex s to vertex t and $\sigma_{st}(v)$ is the number of those paths that pass through vertex v.

  Below is an example test program that uses this class and the corresponding output. This test program uses the airport network illustrated above. Assume that the name of the input file is “anetname”. We will use a similar program to test your solution so make sure that the name of the class is **AirportNet**, its interface is in the file called **AirportNet.h**, and the required member functions are defined as shown above.

```
#include "AirportNet.h"
#include <iostream>
using namespace std;

int main(){
      AirportNet AN("anetname");
      AN.listDirectFlights("Antalya");
      AN.listDirectFlights("Hakkari");
      cout << endl;

      AN.listRoutesOnHubAirport("Ankara");
      AN.listRoutesOnHubAirport("Balikesir");
      cout << endl;

      AN.findConnectedComponents();
      cout << endl;

      AN.displayMostCentralAirport();
      cout << endl;
      return 0;
}
```

The output of this program will be as follows. Following output is not exact output for the input file. It is for sample output format and may include incomplete results or wrong results considering the input file.

```
From  Antalya  3  airports  are  directly  reachable:  Trabzon,  Malatya,
Diyarbakır.
Hakkari does not exist in the airport network.

If Ankara is considered as hub these routes are possible:
<Antalya, Erzurum>
<Antalya, Bodrum>
<Adana, Diyarbakır>

There are two connected components:
For component 1: 12 nodes
For component 2: 7 nodes

Most central airport in the network is: Istanbul
```

Code Format and Notifications

You have to follow the following instructions about the format, programming style and general layout of your program.

- You can use the codes provided in your textbook or the lecture slides. However, you cannot use any external graph implementation such as STL's in your code.
- Don't forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of every file that you are submitting.
- Don't forget to write comments at important parts of your code.
- You are free to write your programs in any environment (you may use either Linux or Windows). On the other hand, we will test your programs in a Linux environment and we will expect your programs to compile and run on Linux. If we cannot get your program work properly on Linux, you will lose a considerable amount of points. Therefore, we recommend you to make sure that your program compiles and properly works on Linux before submitting your assignment.
- You should upload your solutions using the online submission form existing in the Moodle system, attaching one zipped file that contains
  - ✓ ***Section_ID_SurnameName.pdf,*** the file containing the answers to Questions 1 and 2 (do NOT send photos of your solution, type it if it is possible, scan if it is not), and the sample output of the program in Question 3 for the provided main function,
  - ✓ AirportNet.cpp and AirportNet.h, the files containing the C++ codes, and the files for your additional classes, if you implemented any, and
  - ✓ readme.txt, the file containing anything important on the compilation and execution of your program in Question 3.
  - ✓ Do not submit any code containing the main function, for Question 3. We will write our own main function to test your code.
  - ✓ Do not change the prototypes of the member functions given above, for Question 3. We will call these functions, as they are, to test your code.
  - ✓ In the implementation of Question 3, you should not have any memory leaks.
  - ✓ Do not forget to put your name, student id, and section number, in all of these files. Well comment your implementation.
- Late submissions will not be graded.
- This homework will be graded by your TA Mehmet Güvercin (mehmet dot guvercin at bilkent edu tr). You may contact him for further questions.

DO THE HOMEWORK YOURSELF. PLAGIARISM AND CHEATING ARE HEAVILY PUNISHED!!!