CS 202 Fundamental Structures of Computer Science II
Assignment 5 – Graphs

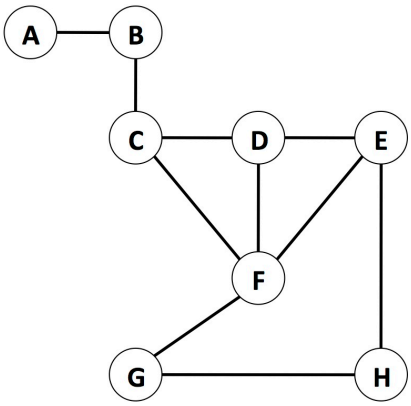
Assigned on: 6 May 2015 (Wednesday)
Due Date: 18 May 2015 (Monday)

Question-1 (20 points)

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

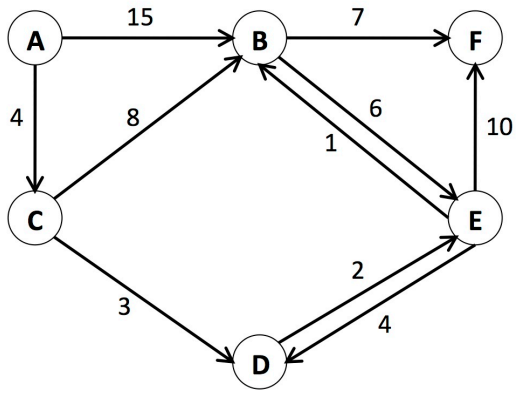
- The depth first traversal algorithm
- The breadth first traversal algorithm

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 Suppose that a postal service company wants to use a system that organizes the delivery of its trucks. In this programming assignment, you will implement a part of this system. In your implementation, you are supposed to compute the minimum number of cities that a truck should visit to deliver a packet from one city to another.

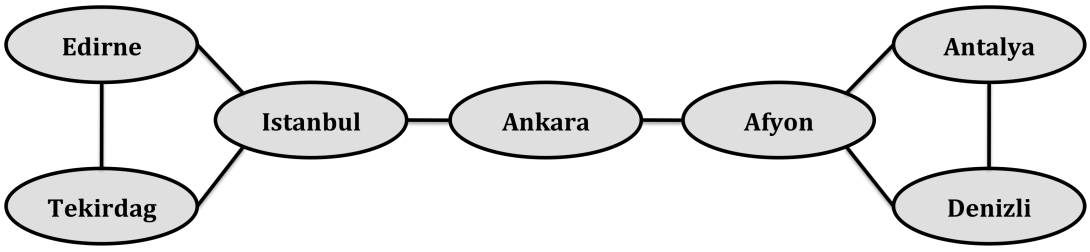
In this implementation, you will be given a map file, which stores the information of the cities and their connectivity (roads) from each other. The first row of the input file indicates the number of the cities and each subsequent row includes information of a particular city. This information contains <id> <name> <degree> <neighbor id>* tokens where there might be many neighbor ids. For a particular city *C*,

- **Id** and **name** are the id and name of city *C*, respectively.
- **Degree** is the number of the neighbors of city *C* (i.e., it is the number of cities that have a direct road from/to city *C*)
- **Neighbor id** is the id of the neighbor city and there will be exactly *degree* neighbor ids after the degree token

For instance, the following table gives an example input file. This file contains 7 cities, as indicated in its first line. After this first line, there exists information for each city. For example, the fifth line of this file indicates that the city with an id of 3 has the name of Ankara and has a direct road to two other cities (i.e., it has two neighbors). The ids of these neighbor cities are 2 and 4. The graph given in this example input file is illustrated in the figure given below.

Input file:

```
7
0     Tekirdag    2     2     1
1     Edirne      2     0     2
2     Istanbul    3     0     3     1
3     Ankara      2     2     4
4     Afyon       3     3     5     6
5     Antalya     2     4     6
6     Denizli     2     5     4
```



In this assignment, your program will take three command line arguments. The first one is the name of the input file and the other two are the names of a source city and a destination city. Your program will output the minimum number of cities to be visited to deliver a packet from the source city to the destination city. It will also output the names of the cities in the visiting order.

```
username@dijkstra:~>./pathFinder  <inputFile>  <sourceCity>  <destinationCity>
```

Suppose that you have an executable called **`pathFinder`**. This command calls the executable with three command line arguments that are the names of the input file, the source city, and the destination city. This executable should output the minimum number of cities. Moreover, it should also list the names of the cities that are to be visited.

For the example input file given above (assume that its name is "input.txt"), the sample outputs are given below:

```
username@dijkstra:~>./pathFinder   input.txt   Ankara   Denizli
3 cities to be visited:
Ankara
Afyon
Denizli

username@dijkstra:~>./pathFinder   input.txt   Tekirdag   Denizli
5 cities to be visited:
Tekirdag
Istanbul
Ankara
Afyon
Denizli
```

In your implementation, you should make the following assumptions:

- The roads between cities are bidirectional. That is, if there is a road from A to B, then there also exists a road B to A.
- In this problem, since we focus on finding the minimum number of cities (but not finding the minimum distance), the input graph does not include the exact distance from one city to another. Thus, you should consider only the available information, which is whether or not there exists a road between two cities. In other words, this problem can be solved using an unweighted graph and there is no need to consider weighted graphs.
- There may exist different solutions but the minimum number of cities to be visited should be always the same. These solutions may contain different lists of the visited cities. It is sufficient to output one of such solutions.
- You may assume that the input file is always valid. The ids of the cities are in between 0 and N-1, where N is the number of the cities. The data in the input file are separated with a white space character.
- You may assume that all cities are connected.

Hint

You may use the breadth first traversal to find the cities to be visited. Note that if all distances (edge weights) are equal, the breadth first traversal can be used to solve the shortest path problem (Remember that in an unweighted graph, one may consider that every edge has a unit weight).

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.
- This assignment is due by 23:59 (sharp) on Monday, May 18, 2015. You should upload your solutions using the online submission form, http://www.cs.bilkent.edu.tr/~saksoy/courses/cs202-Spring2015/upload.html. You should upload a **single zip** file that contains your solution to the first part as **pdf** (do NOT send photos of your solution, type it if it is possible, scan if it is not), code files (only the "**.cpp**" and "**.h**" files that you write for the homework) of the second part and a "**Makefile**" for the compilation of your code that produces the executable named "**pathFinder**". In the end, your zip file should contain ONLY **code files**, "**Makefile**" and **pdf** of the first part; any violation of these causes a significant loss from your grade. Name of the pdf should be "***Section_ID_SurnameName.pdf***".
- If you do not know how to write a Makefile, you can find lots of tutorials on the Internet. Basically they are files that contain a list of rules for building an executable that is used by "make" utility of Unix/Linux environments. "make" command should build an executable called "pathFinder", so write your Makefile accordingly (i.e. at the end, when you type "make" in terminal on your working directory, it should produce "pathFinder" executable)
- Late submissions will not be graded.
- This homework will be graded by your TA Gündüz Vehbi Demirci (gunduz dot demirci at bilkent edu tr). You may contact him for further questions.

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