EE 441 HOMEWORK #3

Gamers Database

Due: December 30, 2016, 23:59
For questions: ccakmak@metu.edu.tr
eylen@metu.edu.tr

Introduction

So far you have completed the Gamer Database and helped the gamers play competitive multiplayer games by implementing matchmaking algorithms. This time, you will implement gamers network to fetch hidden data, which gives information on the relations between players.

Hidden data refers to data you extract from a given obvious data. Think about a player for instance, who beats even the strongest opponent, but has a low score since he/she didn't play as much games as other players did. With the help of some simple functions and a gamers network, this behavior can be detected and used for a better, fairer and more competitive game setup.

Gamers Network

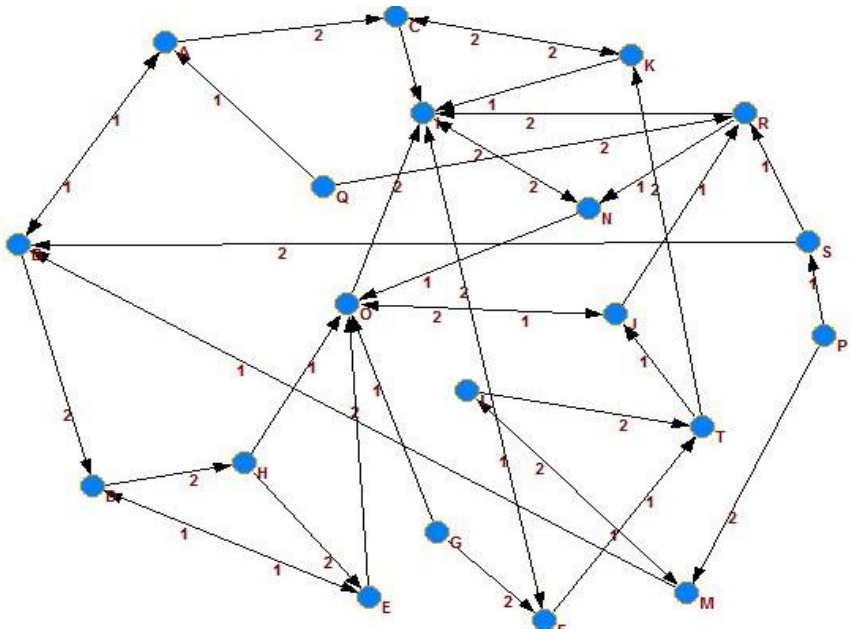
Your task is to implement algorithms to analyze the performance of the players of a 1v1 multiplayer game using individual match results only. The players in the database play games against each other and the result of each 1v1 match (e.g. Player_A beats Player_B) is saved in a log file. Using this data, i.e. individual match results, we will rank the gamers in the database.

For this purpose, you will first create a User class. Each User object will have one-directional weighted links to other User objects who he/she has beaten, the weight indicating how many times A beats B. This can be implemented by creating two arrays, one containing the pointers and the other containing the corresponding weights. Assume that the maximum size of the array is 5, meaning that a single player can beat up to 5 different players.

The information about match results will be retrieved from a log file. The created network should look like the one presented below. An arrow on the network from node *i* to node *j* means that gamer *i* has beaten gamer *j* at least once, and the weight on the arrow shows how many times this has happened.

There are 20 gamers (From A to T) in the network below. In your program, you can assume this is the maximum possible number of gamers.

REMARK: Please note that there are also two-headed arrows in the network. For simplicity, you can treat a bi-directional arrow as two separate one-directional arrows. On bi-directional arrows the weight closer to the node is the weight of the outgoing arrow from that node.



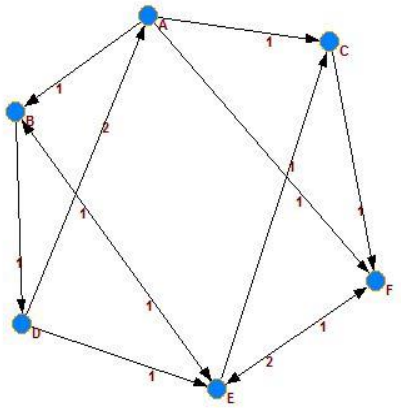
For instance, user P (on the right) in the network above will have the following arrays:

Link_array = [&M, &S, NULL, NULL, NULL]

Weight_array = [2,1,0,0,0]

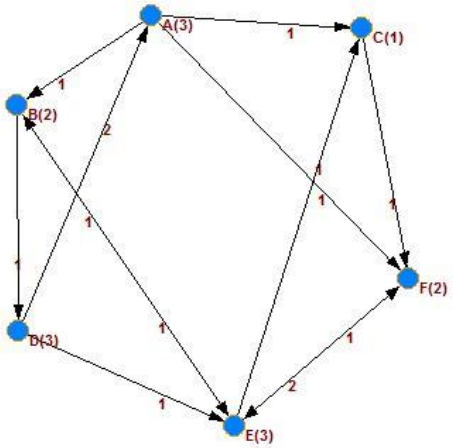
You will implement three functions using this network:

1. User object will keep an attribute called *Basic_Score* which is the sum of the weights of the outgoing links. Basic score gives information on how many games a User has won. Here is an illustration of how the score is calculated:



   Player D has a link to A with a value of 2 and another link to E with a value of 1, so *Basic_Score* of D is **2+1 = 3**.

2. User object will keep another attribute called *Weighted_Score.* This is obtained from a one-step iteration over *Basic_Score.* Weighted_score takes into account not only the number of wins but also the opponents in these games. The *Weighted_Score* of a player *I* is the weighted sum of *Basic_Score*s of all the players *i* has beaten. Below is an illustration of how the score is calculated:



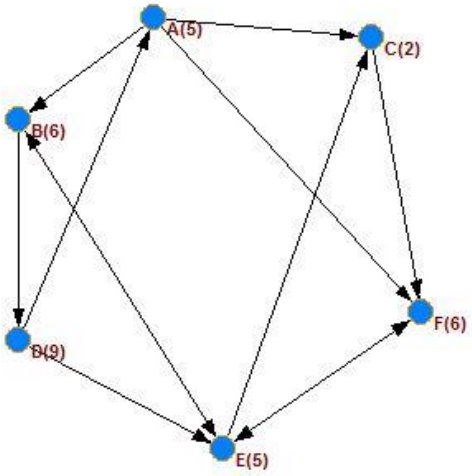
The *Basic_Score* of all players are written next to their names in parentheses. D has beaten A by 2 and E by 1. Thus, the *Weighted_Score* of D is the weighted sum of *Basic_Scores* of A and E, which is calculated as: **2****Basic_Score***(A) + 1****Basic_Score***(E) = 2*3 + 1*3 = 9**

3. User object will keep a third score, which is *Network_Score*. For this score, you will first write a method, which traverses the graph and saves a tuple for each node (hence 2 arrays) (*minDistance, Weighted_Score*) for every reachable player in the network. For example, in User object A, there will be 5 tuples since all other players B, C, D, E and F are reachable from A. Please keep in mind that there may be cases where some players are not reachable from a specific user.

   minDistance specifies the number of edges in the shortest path from the user to another user, e.g. for User A, *minDistance* to E is 2 (A->F->E). Once the minimum distance for every reachable player and their corresponding *Weighted_Score*s are saved, then the *Network_Score* for each User is calculated as follows:

$$\sum_{n} \frac{Weighted\_Score(n)}{minDistance(n)}$$

   where n ϵ All Reachable Players.



For the example case above, minimum distances of other players to User B are as follows

	A(5)	B(6)	C(2)	D(9)	E(5)	F(6)
B	2	-	2	1	1	2

Thus, the *Network_Score* for B is calculated as:

$$\frac{5}{2}+\frac{2}{2}+\frac{9}{1}+\frac{5}{1}+\frac{6}{2}=\ 20.5$$

4. In main, write a function to sort the players based on their scores (either *Basic*, *Weighted, Network*). You have to implement the fastest sorting algorithm.

REMARK: All the links mentioned in this homework are outgoing links. For simplicity, incoming links are not taken into account.

Input / Output

Use the terminal as the user interface of your program. However, match results are to be read from a text file. Your program will first read this file, construct the network described and display an output for the selected function through the terminal. Assume that the input text file looks like this:

```
A,B,C,D,E,F
A B
A C
A F
B D
B E
C F
D A
D A
D E
E B
E C
E F
F E
F E
```

First line shows the players in the network. The following lines show the result of each match. They can be read as: "A won against B", "A won against C" etc. Please note that E and F has won against each other, which is also a valid case.

An example terminal output and function selection menu for the given input text file can be:

```
Welcome to Gamers Database,

Choose an option to perform;
1- Calculate Basic Scores,
2- Calculate Weighted Scores,
3- Calculate Network Scores.
```

```
1
A(3),
D(3),
E(3),
B(2),
F(2),
C(1).

Choose an option to perform;
1- Calculate Basic Scores,
2- Calculate Weighted Scores,
3- Calculate Network Scores.

2
D(9),
B(6),
F(6),
A(5),
E(5),
C(2).

Choose an option to perform;
1- Calculate Basic Scores,
2- Calculate Weighted Scores,
3- Calculate Network Scores.

3
A(21),
B(20.5),
E(20.16),
D(17),
C(13.75),
F(13.25).
```

Regulations:

1. You should insert comments to your source code at appropriate places without including any unnecessary detail. Comments will be graded. You have to write to-the-point comments in your code, otherwise it would be very difficult to understand. If your output is wrong, the only way we can grade your homework is through your comments.

2. Use **Code::Blocks IDE** and choose GNU GCC Compiler while creating your project. Name your project as "e<student_ID>_HW3". Send the whole project folder compressed in a rar or zip file. You will not get full credit if you fail to submit your project folder as required.

3. Your C++ program should follow object oriented principles, including proper class and method usage and should be correctly structured including private and public components. Your work will be graded on its correctness, efficiency and clarity as a whole.

4. Late submissions are welcome, but penalized according to the following policy:

    - 1 day late submission: HW will be evaluated out of 70.
    - 2 days late submission: HW will be evaluated out of 50.
    - 3 days late submission: HW will be evaluated out of 30.
    - 4 or more days late submission: HW will not be evaluated.

Good Luck!