COMP2102 Data Structures and Algorithms
Fall 2022, Project #3
Due Date: January 10 2023, 23:59

IŞIK UNIVERSITY
COMPUTER SCIENCE AND ENGINEERING

COMP2112 Game of Thrones Network

In this project we will model the connections between the characters in Game of Thrones TV series. The data are available here:

https://github.com/melaniewalsh/sample-social-network-datasets/blob/master/sample-datasets/game-of-thrones/got-edges.csv

Our weighted graph will represent this by edge-weights equal to the connection strength among two individual characters.

You can name your class as SeriesCharacter. The graph vertices are Series Characters. (Use a hashtable to map SeriesCharacters into graph vertices: converting strings to integers as vertex numbers).

More explanation on Hashing:

Implement a method or a piece of code using hashing techniques which stores some strings as names (e.g., "ece", "araz", "ali", "sara", …) in an array after generating a hashcode (index) for each name. For example, if hashcode ("sara") = 8, it should be stored in index 8. If index 8 is already full, collision management techniques should be used. The length of hash table (array) is up to you, but do not choose it too big. The initial list of names should be read from an input file that contains 353 lines. In this file, each line contains two names as vertices, and an integer value as edge weight between two vertices. Your program should also be able to search a given string in the array in constant time using hashing techniques (not linear search) and return the index of array containing this name, or -1 if the name does not exist in the array.

Note: We will not use a different input file to test your code but we will enter different input values as parameters of the methods that you are asked to implement (given below).

A) **ReadGraphFromFile():** loads graph data a from file into an adjacency matrix. Before constructing this matrix, a hash table should be also created.

B) **IsThereAPath(String name1, String name2):** Returns true if there is a path between vertex name1 and vertex name2 or false, otherwise.

C) **AllPathsShorterThanEqualTo (int pathLen, int VertexNo, string name1):** prints all paths starting from vertex name1, shorter than or equal to the given path length, with at least VertexNo vertices. No vertex can be repeated in the same path but different order of vertices is assumed as different paths.

D) **ShortestPathLengthFromTo(String name1, String name2):** returns the length of shortest path from vertex name1 to vertex name2. It prints "infinity" if there is no path.

E) **NoOfPathsFromTo(String name1, String name2):** returns the number of paths from name1 to name 2.

F) **BFSfromTo(String name1, String name2):** prints the sequence of vertices while starting a BFS from name1 until reaching name2.

G) **DFSfromTo(String name1, String name2):** prints the sequence of vertices while starting a DFS from name1 until reaching name2.

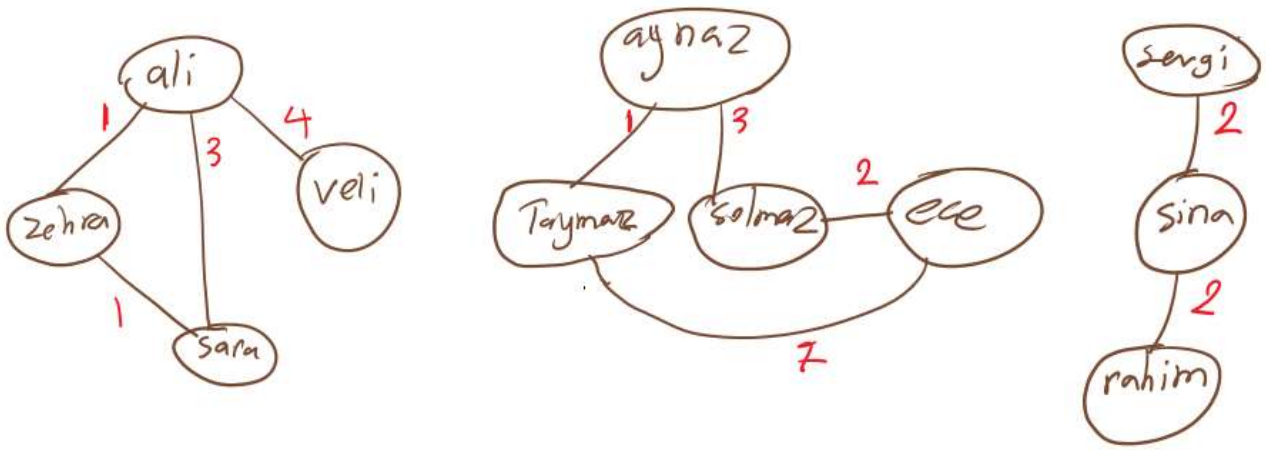
H) **NoOfVerticesInComponent(String name1):** prints the number of vertices that exist in the component that contains name1.

I) **Main method:** in this method, all methods mentioned above should be called one by one, and their input parameters should be received from the user interactively. For example, when calling AllPathsShorterThanEqualTo (int pathLen, int VertexNo, string name1), two integers and one string value should be received from the user in the command prompt.

NOTE 1: In all the methods mentioned above, no vertex can be repeated in the same path.

NOTE 2: In BFS, the edge order while visiting the children of a vertex, starts from edge with minimum weight and continues to the edge with maximum weight, for example, BFS starting from "ali" in the example given below is: ali, zehra, sara, veli

A toy Example:



Input file:

ali zehra 1

ali veli 4

zehra sara 1

sara ali 3

aynaz taymaz 1

aynaz solmaz 1

ece solmaz 2

ece taymaz 7

sevgi sina 2

sina Rahim 2

//---

SAMPLE RUN:

IsThereAPath(“aynaz”, “ece”): TRUE

IsThereAPath(“ali”, “sina”): FALSE

AllPathsShorterThanEqualTo(4, 3, “ali”):

ali, zehra, sara

ali, sara, zehra

ShortestPathLengthFromTo(“ali”, “sara”): 2

ShortestPathLengthFromTo(“ali”, “rahim”): infinity //as there is no path

NoOfPathsFromTo(“ece”, “taymaz”): 2

NoOfPathsFromTo(“rahim”, “sevgi”): 1

NoOfPathsFromTo(“rahim”, “ece”): 0

DFSfromTo(“ece”, “aynaz”): ece, solmaz, aynaz

DFSfromTo(“ece”, “ali”): No path

BFSfromTo(“ece”, “aynaz”): ece, solmaz, taymaz, aynaz

NoOfVerticesInComponent(“Taymz”): 4

//---

Good luck, Comp2112 team

Fall 2022, Isik University

You can use existing class implementations. Do not copy-paste internet code . NEVER!