# SE2205: Algorithms and Data Structures for Object-Oriented Design Lab Assignment 3

**Assigned: Nov 11, 2019; Due: Dec 2, 2019 @ 10:00 p.m.**

## Note

If you are working in a group of two, then indicate the associated student IDs and numbers in the Graph.java file as a comment in the header and in the text box on the submission page for Assignment3 as well. Both individuals should send identical codes.

## INTRODUCTION

A **graph** is a data structure for storing connected data like a network of people on a social media platform. A graph consists of vertices and edges. A **vertex** represents the entity (for example, people) and an **edge** represents the relationship between entities (for example, a person's friendships). A graph can be represented in different forms like **adjacency matrix** and **adjacency list**. In this assignment we have focused on Adjacency List as our graph representation. Figure 1 shows how is a friendship on the online portal and its Adjacency List as an example:

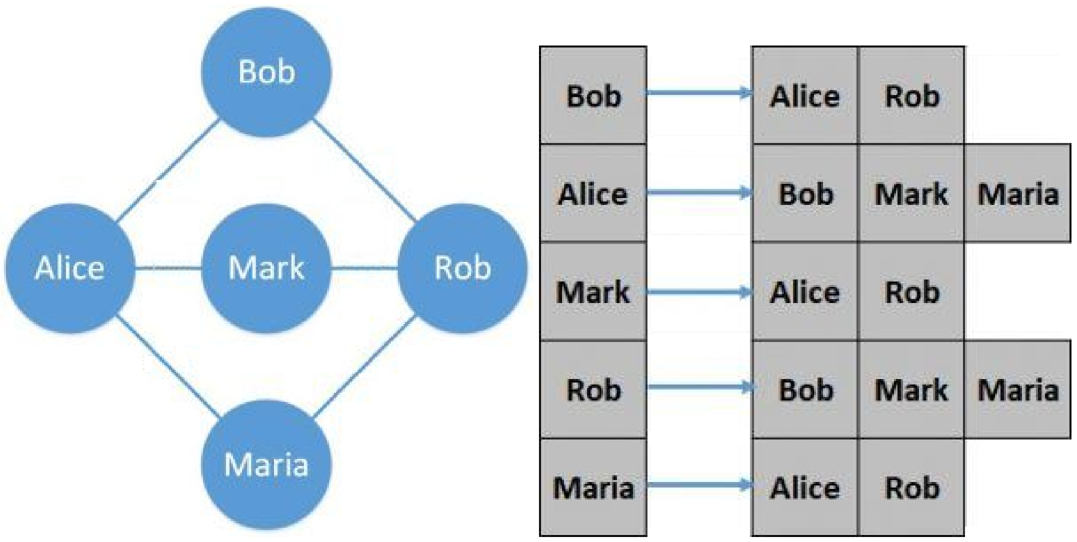

**Figure 1.** An example image of a graph with its Adjacency List.

## YOUR TASK FOR THIS LAB ASSIGNMENT

In this lab assignment, you will implement following algorithms for an undirected graph based on what you have learned in class:

**Breadth-First Search (BFS)** algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layerwise thus exploring the neighbour nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbour nodes.

**Depth-First Search (DFS)** algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

**PrintEdges** method that prints the vertices and the edges between them.

## MATERIALS PROVIDED

You will download content in the folder **SE2205B-LabAssignment3.zip** which contains a skeleton of function implementations and declarations. Your task is to expand **BFS, DFS, and PrintEdges** functions in **Graph.java** file.

- In order to implement the Graph we have used **HashMap** that is a Map based collection class that is used for storing Key & Value pairs. The class `Graph` contains a private variable named `adjacencyMap` which is our data structure for storing the graph representation. It can be seen that the type of this variable is `HashMap⟨Node, LinkedList ⟨Node⟩⟩` which means the `Node` is assigned to a `LinkedList` in which we store the list of connected nodes to the Node. In other words, the `adjacencyMap` represents the edges in which the `Node` is source and `LinkedList` is destination nodes.
  The `adjacencyMap.get(Node)` method returns the list of connected nodes to the `Node` and by using `adjacencyMap.put(Node,LinkedList)` method, you are able to define a edge that shows the connection between the `Node` and `LinkedList` which contains the destination nodes.

- In order to implement DFS algorithm, firstly you should get all neighbors of the input node and go through the all neighbors of it and check whether they have been visited or not using `visit(),unvisit(),isVisited()` methods in the `Node` class and mark them as visited if you visited them and continue until there is no more unvisited nodes.

- To implement BFS traversal algorithm, you should simulate a queue using `LinkedList<Node>` and enqueue your input node as initial point. Every time you dequeue an item from the queue and mark it as visited and enqueue the connected nodes and continue it untill there is no unvisited nodes.

- To implement `printEdges`, you should go through a loop over the `adjacencyMap` and print the nodes and adjacent vertices which are connected by edges.

**Test.java** evokes all the functions you will have implemented and is similar to the file that will be used to test your implementations.Use Test.java file to test all your implementations. **Note** that we will **NOT** use function calls with the same parameters for grading your lab assignment. Do **NOT** change the name of these files or functions.

## BONUS

There is 20% bonous for this assignment if you have plan to do the assignment using JavaFX. For this purpose, you can download **SE2205B-LabAssignment3-javafx.zip** from OWL. You have been provided with a JavaFX project which generates a GUI similar to the figure 2 below:

First you need to import project into Eclipse, NetBeans, or other IDE which you prefer to work with and implement the following items.After running project you will see a main form in which there are three main buttons on the right side.

**"Add Node" button** by clicking on this button, a new windows will be opened in which you are able to add a new vertex to your graph.

**"Add Edge" button** will open a new window in which you define an edge between two vertices.

**"Print Edges" button** prints the vertices and the connected edges.

On the left side,

**"RadioButtons"** for selecting DFS or BFS algorithms have been placed.

**"Run" button** runs a algorithm based on what user has selected and show the results in the result section.

**Hint:** First you need to implement the BFS,DFS, and printEdges methods in Graph.java file. Then you include **Graph.java** and **Node.java** files into the **Main.java** file of GUI project. After that, for each GUI components in the forms such as button, radiobutton, etc you should set appropriate BFS, DFS, addEdge, printEdges, etc methods from Graph.java file.

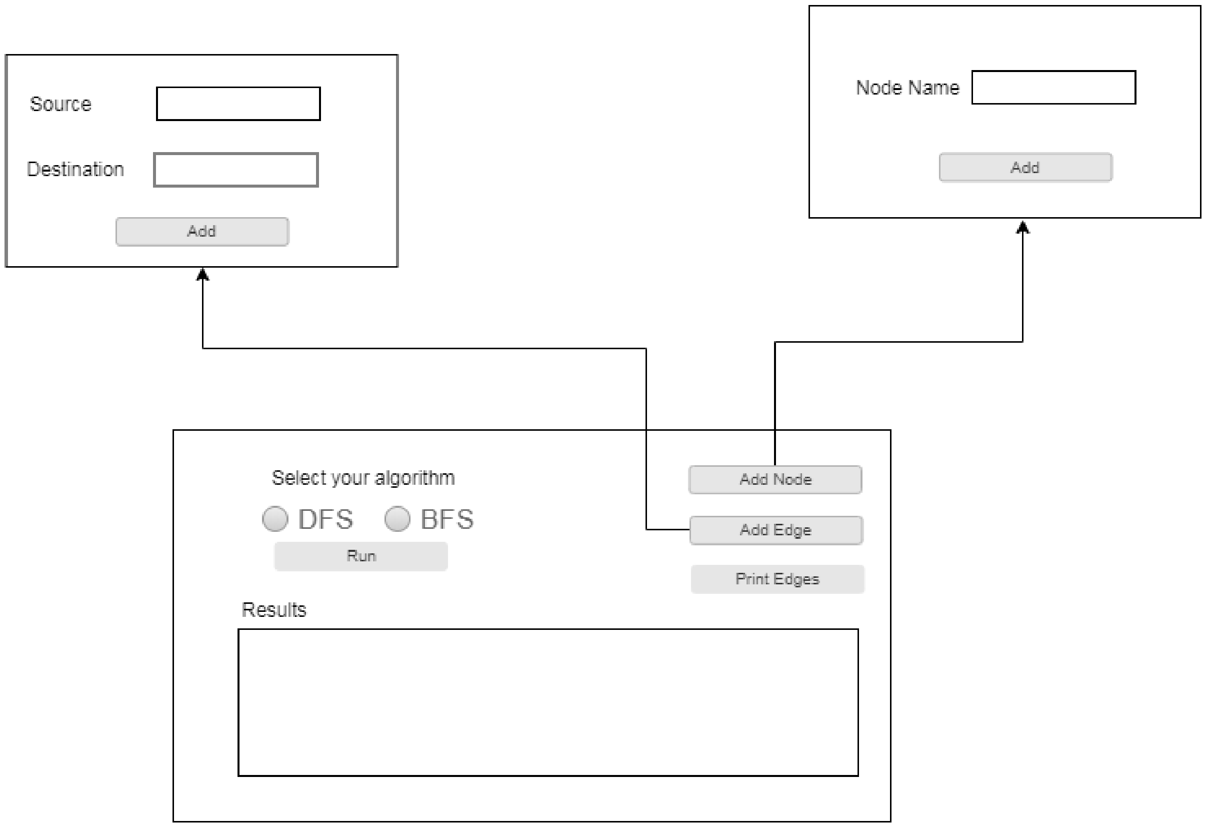

**Figure 2.** JavaFX form for the assignment

## GRADING: FINAL MARK COMPOSITION

It is **IMPORTANT** that you follow all instructions provided in this assignment very closely. Otherwise, you will lose a significant amount of marks as this assignment is auto-marked and relies heavily on you accurately following the provided instructions. Following is the mark composition for this assignment (total of 40 points):

1. Successful compilation of all program files i.e. the following command results in no errors [3 points]:
   javac Node.java Graph.java Test.java
2. Successful execution of the following commands: [2 points]
   java Test
3. Output for BFS matches expected output (the expected output has been commented in the Test.java file)[10 points]
4. Output for DFS matches expected output (the expected output has been commented in the Test.java file)[10 points]
5. Output for PrintEdges matches expected output (the expected output has been commented in the Test.java file) [5 points]
6. Output Code content [10 points]

** For GUI, the final mark composition is similar.

## CODE SUBMISSION

1. Only upload Graph.java file to OWL. If you have worked on GUI, please zip the whole project and upload it to OWL.
2. If you are working in a group of two, then indicate the associated student IDs and numbers in the Graph.java file as a comment in the header and in the text box on the submission page for Assignment3 as well. Both individuals should send identical codes.
3. You submit before the deadline.
4. Your code compiles without error (if it does not compile then your maximum grade will be 3/40).