COMS2015 : Analysis of Algorithms

Lab 3

1 Adjacency Lists

Today's lab is focused on familiarising yourself with a particular graph representation known as an Adjacency List. As shown in the book and discussed in class, a graph is a structure made up of vertices and edges. In this exercise we will directly represent the vertices as instances of a Vertex class and we will represent the edges as a list of other Vertex objects that each Vertex is connected to by an edge.

An Adjacency List can be constructed simply in Java, using a Vertex class as shown in the lectures, combined with Java's built in LinkedList and ArrayList classes.

1.1 Setup

1. In Eclipse, create a project for this lab
2. This Project will have a Vertex class and a Graph class

1.2 Vertex

This class is going to be used to represent an individual node in our graph

1. Create a new class in your project, called Vertex
2. This class must have the following attributes:
   - vertexNumber, which is an integer
   - colour, which is an integer
   - adjacencies, which is a LinkedList containing Vertex objects. These are the Vertex objects that are connected to the current Vertex by an edge.
3. It must also have a constructor that takes in an integer in order to set the vertexNumber property.
4. Note that the edge here is unweighted and undirected. This means that an edge between vertex 2 and vertex 5 is equally an edge between vertex 5 and vertex 2.
5. In addition to the constructor, the Vertex class must have the following methods:
   - addAdjacency - takes in a Vertex object and adds it to the adjacencies LinkedList
   - isAdjacent - takes in a Vertex object and returns true if the given Vertex is adjacent to the current Vertex object, and false otherwise
   - getDegree - returns the degree of the current Vertex

1.3 Graph

This class is going to be used to represent a collection of vertices.

1. Create a new class in your project, called Graph
2. This class must have the following attributes
   - vertices, which is an ArrayList containing Vertex objects
3. The Graph class must have the following methods:
   - addVertex - this method has no parameters. When called, it creates a new Vertex and adds it to the vertices ArrayList. Note that in order to create a new Vertex, it needs to pass a vertex number to the constructor of the Vertex object. Note that the size of the ArrayList is the number of vertices that have already been added, so when you want to add a new one, you can set its vertex number to be the size of the ArrayList.
   - getVertex - takes in a vertex number and returns the Vertex object in that position in the vertices ArrayList
   - addEdge - this method takes in two vertex numbers and adds them to each others' adjacency lists so that the vertices are adjacent
4. Since we do NOT want to make all the methods static, we use the Graph class's constructor instead of the main method. We then call the constructor from the main method. Since the constructor is not static, this allows us to keep all the methods from having to be static. An example of this is shown below.

```
public class SomeClass{
        public SomeClass(){
             //Code goes here instead of in the main method
        }

        public static void main(String args[]){
             new SomeClass();
        }
}
```

5. In the constructor, we must create five vertices, and the following four edges - {0,1},{0,2},{0,3},{2,4}.
6. For testing, the program must then print out the degree of each vertex

1.4 Input

Now, modify your program so that it can read the graph from standard input instead of manually creating the vertices and edges. The first line of the file will contain the number of vertices. The lines that follow contain the edges that must be represented. An example input is shown below:

```
11
0,1
0,2
```

```
0,3
2,4
3,5
3,6
3,7
7,8
7,9
9,10
-1
```

The program must output the vertex number and degree of each vertex as shown below:

```
0:3
1:1
2:2
3:4
4:1
5:1
6:1
7:3
8:1
9:2
10:1
```

Importantly, you have no way of knowing how many lines there will be in the input just from the number of vertices. In this case, we know there are 11 vertices, but we have no idea how many edges there are in the graph. The -1 at the end of the input is our indicator that there is no more input and therefore no more edges.

Note that this lab will be used as the basis for next week's lab, so be sure to get it right. Ask your tutors and lecturer for assistance if necessary.

1.5 Program

In order for the marking system to mark your program, you must have a class called Program. In this case, all the Program class will do is create an instance of the Graph class. Your main method will look as follows:

```
public static void main(String args[]){
   new Graph();
}
```