School of Computing
National University of Singapore
CS2010: Data Structures and Algorithms 2
Semester 2, AY 2015/16

Assignment 3 – **Undirected Graphs**
Assigned – March 2, 2016
Due – March 18th (Friday by 11.59pm), 2016 (in-lab assessment March 22, 2016)

1) Objective

- Using undirected graphs data structures
- Applying graph algorithms
- Even more fun with Java

2) Preparation

Download A3.zip. The zip contains the following: java source code (`A3Main.java`), data file `countries_borders.dat`. A set of example outputs will also be shared.

3) The input

In this assignment you given a list of countries and its neighbors. The data file is in the following format:

```
Country Name >> Neighbor Country Name#1[: XX km]; Neighbor Country Name#2[: XX
km]; ..
```

Here `[]` means it is optional. In most cases, the country has a border, but there are a few cases that do not. Note that every line ends with a semi-colon.

Example:

```
--
Adélie Land (France) >> Australian Antarctic Territory (Australia);
American Samoa (United States) >> ;
Ecuador >> Colombia: 590 km; Peru: 1,420 km;
--
```

In the first example, `Adélie Land (France)` has only one (1) neighbor, which is `Australian Antarctic Territory (Australia)`. However, no border length in given this case. For such examples, you can assume that the border length is 0, however, do not forget to add the neighboring country to the graph.

In the second example, `American Samoa (United States)` has no neighbors, however, this country should still be added to the graph. The examples of `Adélie Land` and `American Samoa` are territories of other countries (in this case France and US respectively), however in this assignment we will treat them as a separate countries.

The third example `Ecuador >> Colombia: 590 km; Peru: 1,420 km;` Ecuador, has two (2) neighbors. The border between Ecuador and Colombia is 590km, the border between Ecuador and Peru is 1420 km.

Your assignment will be to write (in Java) a menu-driven console interface that allows the user to list the countries and check if paths between two countries exist (and if so, what is the shortest path in terms of # of countries visited [note that border length does not matter in the shortest path computation]). Additional functionality can also be provided. See details in the following.

Example of menu:

```
================ Assignment 3 ==================
1. List all countries and bordering countries
2. Find shortest path
3. List all countries with X neighbors
4. Block a country
5. Find countries with borders larger than X km
6. Show all connected components (largest to smallest)
7. Exit
>
```

3) Assignment requirements

To get an "average" grade, the minimum you are required to do is to accomplish task 3.1. To get an "above average" grade, you need to implement additional tasks as listed.

3.1 – [Minimum Require for the Assignment (65/100 ~ B grade)]

a) List all countries by integer ID [0 ~ N-1] and its neighboring countries.

b) Find shortest path between two countries in terms of number of countries visited (you do not need to factor in the countries border in the shortest path computation). If a path does not exist, output that there is no path.

Additional tasks beyond the minimum requirement

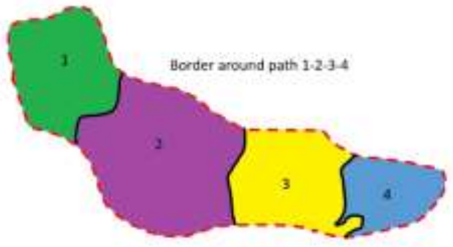
3.2 [List all countries with N neighbors] (+5)
List all countries with N neighbors, where the user provides the size of N.

3.3 [Block a country] (+10)
The user can give the vertex ID of a country to "block". This means that paths can use the blocked country. If you implement this, modify your shortest path output to first list all block countries, then provide the shortest path (see example). Note that once a country is blocked, it can't be "unblocked".

3.4 [Find countries with borders larger than X] (+5)
Find all countries with borders larger than X (where the user inputs X). Print the country and its neighbors.

3.5 [Show all connected components] (+10)
Show all the connected components of the country graph (assign a component ID and list all the countries for that component). Sort the components from largest to smallest.

3.6 [Show border around a path] (+5)
Modify the output of your shortest path operation to also output the border around the shortest-path. See example on left.



4) Required Data Structures and Notes

This assignment clearly requires a graph data structure. You are allowed to use Java's native Graph functionality, but it might be best to use the Graph code from our lecture notes (posted on the CS2010 schedule page). You are allowed to modify the Graph code in *any way* you like to achieve the assignment's requirements. This includes modifying the Graph class itself, as well as the Graph processing code such as BFS and DFS.

We do ask that you follow the menu-structure in the `A3Main.java` file.

5) Submission Instruction

What to submit?

Your source code (*.java files). There is no need to resubmit the country_border.dat file.

Submit the following:

1. Submit your code on IVLE before 11.59pm Feb … (late assignments will be deducted -15 pnts per day).
2. Please put your files in a folder and zip the folder. Use the following convention to name your zipped folder: MatriculationNumber_yourName_Assignment3. For example, if your matriculation number is A1234567B, and your name is Chow Yuen Fatt, for this assignment, your file name should be A1234567B_ChowYuenFatt_Assignment3.zip. As with assignment 2, please name this correctly, points will be deducted for incorrect submission names. We use a script to extract your code for the in-class assessment, in correct file name doesn't work with our script.
3. It is up to you to test to make sure that your Java code in the zipped file is complete. It is recommended that you unzip your Java code in a different location and test it to make sure it runs.

Assessment (Mar 21, 2016 – In Lab)

- You will be assessed individually in the lab session (you must attend your assigned lab!)
- You'll have approximately 4-5 minute Q&A session with Dr. Brown or one of the TAs (Hakki, Sixing or Abdel).
- We will run the code directly from your submitted IVLE submission, you won't be able to bring an updated version to the lab.
- We will run your code and then ask you anything about your code we want. Be prepared to answer questions. Please comment your code if you think you will forget by the time we do the assessment.