COMP 251 Assignment 3

Prepared by Prof. Michael Langer

Posted: Wed. Feb. 26, 2014
Due: Sunday, March 16, 2014 at 23:59.

General Instructions

- The T.A.s handing this assignment are Bentley Oakes and Syed Ahmad. Their office hours will be posted on the public web page. Office hours will be in Trottier 3110.

- The other instructions are the same as in Assignments 1 and 2.

Introduction

Flow networks are a very powerful and surprisingly general tool for solving optimization problems in computer science. You will learn much more about network flows in COMP 360 but there you will only cover theoretical aspects.

In this assignment, you will have the opportunity to get some experience with a simple flow network data structure and the Ford-Fulkerson algorithm for computing maximum flow. This algorithm finds augmenting paths by traversing the residual graph. Here you will use breadth first search. It can be shown (and may see this in COMP 360) that breadth first search automatically leads to very fast performance for finding the maximum flow in a graph. This is known as the Edmunds-Karp algorithm.

I am providing you with a partial implementation of a **FlowNetwork** class which is designed to be consistent with the algorithms and data structures presented in class. I have also included a slightly more elaborate **Graph** class than the one from Assignment 2, which now includes a breadth first search method.

Notes:

- I have set the edge weights to be of type double, even though the algorithm presented in class assumed integer weights. The reason I have done so is that I would like you to explore what happens when you run the code with non-integer weights. For the assignment grading, the graph will be tested on integer values only (that is, integers of type double), as in the example graph that is provided for you.

- The algorithm presented in class implicitly assumed that if there is an edge from vertex u to v in the graph G, then there is no edge from v to u. This was the case for the examples given as well. If we had allowed an edge from u to v and another edge from v to u, i.e. a cycle of length 2, then we might end up with both a forward and backward edge from u to v in the residual graph (and another forwards and backwards edge from v to u). Having two edges from u to v would make it a “multigraph” and we have not discussed how to represent such data structures. To avoid this situation, you may just assume that there are no cycles of length 2 in the input graph.

Your task

Implement the following methods:

- **findAugmentingPath()** - 30 points
- **computeBottleneck()** - 30 points
- **augment()** - 40 points

The assignment is out of 100 points..

What to Submit

- A single file called **FlowNetwork.java** with your name and student ID at the top. Please read the General Instructions carefully to avoid problems with your submission.

Have fun, get started early, and good luck!