

Program 5

Due: April 24, 2009 (by midnight).

Graphs can be used to represent several different kinds of networks (e.g., computer networks, social networks, etc.). One interesting property of a node in a network (vertex in a graph) is how well connected it is to the other nodes in the network. Well-connected nodes have greater influence over the spread of information through the network. For example, marketers can target such nodes to quickly spread news of their product through the network. Those wanting to disrupt a network can attempt to eliminate such nodes.

As a measure of how well-connected a vertex is in a graph, we will use *closeness*. The closeness of a vertex v in a graph G is defined as the average *cost* of the shortest paths from v to every other vertex in G reachable from v . The goal of this programming assignment is to implement a program that can identify the vertices in a graph with minimal closeness (most well connected). The details follow.

1. Implement a **Graph** class that uses an adjacency list representation to store a directed, weighted graph. Undirected graphs can be represented by adding two directed edges between vertices connected by an undirected edge.
2. Implement the **dijkstra** shortest path algorithm as a method within your **Graph** class.
3. Implement a method within your **Graph** class to compute the **closeness** of a vertex.
4. Implement a **readGraph** method within your **Graph** class that can input a graph from a file in GML format. See <http://www.infosun.fim.uni-passau.de/Graphlet/GML/> for a description of the GML specification. We will be using a subset of GML; namely, graphs only have the “directed” property, nodes have an “id” and an optional “label”, and edges have a “source” id, a “target” id, and an optional “value” (the edge weight). If an edge’s value is absent, its weight is assumed to be 1. *Any other properties can be ignored.*
5. There are several interesting networks in GML format at Mark Newman’s website <http://www-personal.umich.edu/~mejn/netdata/>. Your program should work on any of these datasets. We will use some of these to test your program.
6. Your program should accept the GML file name as a command-line argument. Your main procedure should read the graph from the given file, compute the closeness of each vertex in the graph, and then output the id, label (if available), and closeness value for every vertex in the graph in increasing order by their closeness value.
7. Create a **readme.txt** file that describes exactly how to compile and execute your program and on what platform.

8. Collect your source code, readme file and any other files needed to compile and execute your program into one ZIP file called `<Your_last_name>-pgm5.zip` and include it as an attachment to an email to me (holder@wsu.edu) by the deadline. Grading will be based not only on correctness, but also on programming style and documentation.

Special notes:

- This programming assignment is more difficult than the previous programming assignments and will be worth more points. Be sure to reserve extra time to complete this assignment.
- There are a lot of information and source codes on the web related to this problem. Still, your solution must be your own. Using code from others, either in whole or in part, will be considered plagiarism, and thus cheating.