COSC 201 – Assignment #3
Fall 2016

Objective: In my opinion, Graphs and Graph Algorithms are essential to any Algorithms and Data Structures course. In this project, you will explore how to represent a graph as a data structure, then apply two pathfinding/graph search algorithms:

Dijkstra's algorithm
Floyd-Warshall algorithm

Both algorithms can be considered for a positively-weighted, directed graph (though in both cases, they can be applied to undirected graphs as well), and both algorithms are actively used in industry, specifically in videogames. In addition, Floyd-Warshall is a more general algorithm, and can be used for graphs regardless of whether the weight is positive or negative and actually computes all-pairs shortest path. Dijkstra's algorithm can be found in section 14.3, and the Floyd-Warshall algorithm can be found here:

http://math.mit.edu/~rothvoss/18.304.1PM/Presentations/1-Chandler-18.304lecture1.pdf
http://www.cs.umd.edu/~meesh/351/mount/lectures/lect24-floyd-warshall.pdf
https://www.cs.usfca.edu/~galles/visualization/Floyd.html

(as well as other places on the web). Note, that these algorithms have been chosen because you can find a ton of resources for these algorithms. As a reminder: **you must cite your sources** if you use a source other than one of the ones provided in this document. **Failure to do so will result in a 0 for this project.**

Representing our graph could be done either as an adjacency matrix or adjacency list (see section 14.1). It is my opinion that an adjacency matrix is the easier of the two to understand and implement. As such, I will be using an adjacency matrix to represent our graph (see below). Again, you can find a ton of resources for these via simple Google search.

This project is designed to test your ability in reading algorithms and translating them to Java. This is something that is commonly asked of students and practitioners, and as such, is an essential skill. To help promote this, the TA and I will answer any specific question in regards to understanding the algorithms (no "Could you walk through and explain the entire algorithm for me?"), but will not discuss how to translate the algorithms to code. Nor will we dedicate a lecture in class to the algorithms (we will discuss, briefly, the possible representations of a graph).

Code Requirements: As with the last project, you will utilize a driver that I provide to test your code. Here are the details:

Class name: GraphAlgorithms
Class constructor signature: public GraphAlgorithms()
Primary method signature: public int processGraph(int[][] graph, int algorithm, int s, int d);

The parameters for the method signature include a two-dimensional integer array representing our graph, an int (algorithm) representing which algorithm I would like run on the graph (1 for Dijkstra, 2 for Floyd-Warshall), an integer representing the source node, and finally an integer representing the destination node. Your code should then compute the shortest weighted path from the source node to the destination node using the algorithm indicated, print the path, and return the cost.

As before, you will be able to test your code with my driver before needing to submit your code. If your code does not work with my driver, check your signatures with the above requirements first.

Testing: You should write your code with the method signatures noted above. I will be using my own driver class to run your projects and you should test your code using said driver class. The driver

(Proj3Driver.java) will be posted to the course website by 11/30 and will include appropriate and informative output. The driver should not be altered by you in any way and will include the following errors that you will need handle by printing out an informative message and continuing (do not let the program crash on an invalid input).

Errors

No path present from s to d.
Source vertex does not exist.
Destination vertex does not exist.

Expectations: Your code will need to be neat, concise, well documented and above all, correct (see Testing). All classes should have headers and each method should have comments describing the method's function including pre and post conditions (see http://www.cs.colorado.edu/~main/supplements/chapt01.ppt for a good introduction to pre and post conditions). Any novel or possibly confusing code should be explained, as I do get confused and distracted easily.

Grading rubric will be given out at least a week ahead of the due date. You may work in teams of up to three for this project. We recommend working in teams on this project. If you are choosing to work in a team, a single member of the team must email me by November 30th at 5pm. No exceptions.

Learning Targets: graph data structure, graph algorithms, two-dimensional arrays, translation of pseudocode, dynamic programming

DUE: All source code due December 7, 11:59 pm Eastern via Blackboard.