CS 594/690, Graph Algorithms, Applications and Implementations
Spring 2017, Homework 3

1. In a finite, simple graph, one can find the length of the shortest path between two vertices *u* and *v* by performing a BFS, starting at *u* and continuing until encountering *v*, keeping track of the depth at which each vertex is first encountered. Assume that indices are used to order branching. Answer the following. If a strategy does not work, provide a graph of order at most five as an example of where it fails.

a. Will the BFS strategy work to find the longest path between *u* and *v*?

b. Will the BFS strategy work to find the longest cycle between *u* and *v*?

c. Will a DFS strategy work to find the longest path between *u* and *v*?

d. Will a DFS strategy work to find the longest cycle between *u* and *v*?

2. Write a program that inputs a weighted (but undirected) graph and finds the shortest path between two vertices using Dijkstra's algorithm, using indices to order the branching. The program should take three command-line arguments: the name of the graph file, followed by a source vertex, followed by a destination vertex. It should output to standard output the vertices in a shortest path in order, including both the source and destination vertices, as well as the total weight of the path. You may assume that any input graphs will have no negative-weight edges. A user should see something very similar to the following when invoking your program.

```
>./dijkstra graph3.txt 2 3
2 5 6 1 0 3
3.51
>
```

Bring a hard copy of your answers to problem 1 to turn in at the beginning of class on Wednesday. You may draw any graphs by hand, but the rest of your answers must be written electronically and printed out.

The following apply as usual.

- You may choose any programming language you wish, as long as your program compiles and runs when invoked from the Linux command line on the EECS Linux machines, using only software currently installed. I will test your code on one of the Hydra Lab machines.
- Do not use any library routines specifically designed for graphs (e.g. Boost).
- Include an example how to compile and/or invoke your program in a README.txt file.

Submit your program by emailing all files necessary to compile and run your code to cphill25@utk.edu prior to the beginning of class next Wednesday, February 1. Include your

README.txt file. If you have any questions, please do not hesitate to email me or drop by during office hours.