Aidan Duffy
Boston University
METCS 526

Module 6 Homework (Problems 1 through 4)

Problem 1 (10 points)
For this graph the DFS, starting at G, is as follows:

G→E→B→A→C→F→D
This is because we choose E over F. From there, since B comes before D and F in the alphabet, that is the node we go to. From B, we go to A, which is the lowest depth. After A, we go back to B and visit C. From C, since E has already been visited, we visit F. Finally, we go to C, then B, then E, then finally we visit D. Thus, we complete the DFS.

Edge Classifications:
Tree Edges: G→E, E→B, B→A, B→C, C→F, E→D
Forward Edges: G→F, E→F
Back Edges: D→G, C→E
Cross Edges: D→A

Problem 2 (10 points).

BFS starting at node I:

I→F→H→E→G→B→C→D→A

Here, we start at I, then the next depth is F, then H. Then, the next depth from F is E and from H is E and G. Since we already visited E, we just go right to G. From E, we visit the next depth in alphabetical order, which is B, then C, then D. The next depth from G is C, but that has already been visited. From here, we go to node B and its next depth, which is just node A. Thus, we finish our BFS.

Problems 3 (10 points each).

3.1: D-values at every iteration

Iteration 0 (S, a, b, c, d, e): [0, infinity, infinity, infinity, infinity, infinity]
Iteration 1 at S: [0, 16, 5, 12, infinity]
Iteration 2 at b: [0, 8, 5, 10, 9, infinity]
Iteration 3 at a: [0, 8, 5, 10, 9, infinity]
Iteration 4 at d: [0, 8, 5, 10, 9, infinity]
Iteration 5 at c: [0, 8, 5, 10, 9, 12]
Iteration 6 at e: [0, 8, 5, 10, 9, 12]
Now that we have visited all the nodes, we have finished.

3.2: Shortest Paths

From S to a: S→b→a
From S to b: S→b
From S to c: S→b→c
From S to d: S→b→d
From S to e: S→b→c→e

Problem 4 (10 points).

4.1: Sequence of Nodes

A→C→B→D→G→F→E

4.2: MST T as a Set of Edges

T = {A→C, C→B, B→D, D→G, D→F, F→E}

Problem 5 (60 points).

Discussions/Observations: I thought that the best approach for this was to modularize the steps. First I designed a method that read in from the given input file of friend pairings which generated a hash map whose key-value pairs are Strings. The keys are the individuals names, and the values are that person's friends, separated by commas. From here, I developed a method that would take that HashMap, restructure it as a TreeMap so all the individual's names are in alphabetical order for printing, and populated the adjacency matrix. Finally, I developed a method which printed out the adjacency matrix and all of the names in a readable way. After all this, the main method would create the main menu and continue to run it in a while that won't exit until the user inputs 3. Ultimately, I actually found the printing to be the most difficult task. I found the map data structure to be by far the most useful as pairing a person with all of their friends in a way that is quick to access and update allowed for the program to run quickly.