MAT 258 - CODING ASSIGNMENT #3

due Monday, December 4, 2017 at 11:50 PM.

OBJECTIVE: Students will implement algorithms from graph theory.

GRADING: The assignment is worth 5% of your course grade.

INSTRUCTIONS:

- Students may work individually or in pairs. Each team must submit their own code, but they may ask questions and clarification from classmates and the instructor.
- Students must submit their projects on Moodle.

SUBMIT THE FOLLOWING:

- An executable. This should be able to run on a clean machine, please compile it accordingly.
- A copy of your code.
- A read-me file explaining how to run your code.
- Answer Sheet with answers to the specific problems.

PROJECT:

I. Code the following problems:

1. Connectivity
   - User will input n, the number of vertices in the graph G.
   - User will input the (weighted) adjacency matrix for the graph.
   - Print the (weighted) adjacency matrix.
   - Check if G is connected: if connected, print "G is connected"; if not connected, print the connected components (as disjoint sets of vertices).
2. Dijkstra's Algorithm
   - User will input n, the number of vertices in the graph G.
   - User will input the (weighted) adjacency matrix for the graph.
   - Print the (weighted) adjacency matrix.
   - User will chose two vertices a and z, with $1 \leq a, z \leq n$.
   - Use Dijkstra's Algorithm to find the **shortest path** from vertex a to vertex z.
   - Print the length of the shortest path
   - Print the shortest path, with vertices in the order traveled.

3. Prim's Algorithm
   - User will input n, the number of vertices in the graph G.
   - User will input the (weighted) adjacency matrix for the graph.
   - Print the (weighted) adjacency matrix.
   - Use Prim's Algorithm to find a **minimum spanning tree** for G.
   - Print the (weighted) adjacency matrix for the resulting tree.
   - Print the total weight of the tree.

4. Kruskal's Algorithm
   - User will input n, the number of vertices in the graph G.
   - User will input the (weighted) adjacency matrix for the graph.
   - Print the (weighted) adjacency matrix.
   - Use Kruskal's Algorithm to find a **minimum spanning tree** for G.
   - Print the (weighted) adjacency matrix for the resulting tree.
   - Print the total weight of the tree.

II. Test your program on the following graphs and include the output in the **Answer Sheet.**

1. Connectivity

(a) $\begin{bmatrix} 0 & 3 & 1 & 0 \\ 3 & 0 & 7 & 1 \\ 1 & 7 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

(b) $\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

(c) $\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$

(d) $\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$

2. Dijkstra's Algorithm: Use $a = 1$, and $z = 4$ for each example.

(a) $\begin{bmatrix} 0 & 3 & 1 & 0 \\ 3 & 0 & 7 & 1 \\ 1 & 7 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

(b) $\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

(c) $\begin{bmatrix} 0 & 4 & 0 & 0 & 2 & 0 \\ 4 & 0 & 3 & 0 & 0 & 3 \\ 0 & 3 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 3 \\ 0 & 3 & 0 & 1 & 3 & 0 \end{bmatrix}$

(d) $\begin{bmatrix} 0 & 4 & 0 & 0 & 4 & 0 \\ 4 & 0 & 8 & 0 & 1 & 0 \\ 0 & 8 & 0 & 6 & 1 & 2 \\ 0 & 0 & 6 & 0 & 0 & 2 \\ 4 & 1 & 1 & 0 & 0 & 5 \\ 0 & 0 & 2 & 2 & 5 & 0 \end{bmatrix}$

3. Prim's Algorithm

(a) $\begin{bmatrix} 0 & 3 & 1 & 0 \\ 3 & 0 & 7 & 1 \\ 1 & 7 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

(b) $\begin{bmatrix} 0 & 4 & 0 & 0 & 2 & 0 \\ 4 & 0 & 3 & 0 & 0 & 3 \\ 0 & 3 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 3 \\ 0 & 3 & 0 & 1 & 3 & 0 \end{bmatrix}$

(c) $\begin{bmatrix} 0 & 4 & 0 & 0 & 4 & 0 \\ 4 & 0 & 8 & 0 & 1 & 0 \\ 0 & 8 & 0 & 6 & 1 & 2 \\ 0 & 0 & 6 & 0 & 0 & 2 \\ 4 & 1 & 1 & 0 & 0 & 5 \\ 0 & 0 & 2 & 2 & 5 & 0 \end{bmatrix}$

(d) $\begin{bmatrix} 0 & 2 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 4 & 0 & 3 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 3 & 0 & 3 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 3 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 1 & 0 \end{bmatrix}$

4. Kruskal's Algorithm

(a) $\begin{bmatrix} 0 & 3 & 1 & 0 \\ 3 & 0 & 7 & 1 \\ 1 & 7 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

(b) $\begin{bmatrix} 0 & 4 & 0 & 0 & 2 & 0 \\ 4 & 0 & 3 & 0 & 0 & 3 \\ 0 & 3 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 3 \\ 0 & 3 & 0 & 1 & 3 & 0 \end{bmatrix}$

(c) $\begin{bmatrix} 0 & 4 & 0 & 0 & 4 & 0 \\ 4 & 0 & 8 & 0 & 1 & 0 \\ 0 & 8 & 0 & 6 & 1 & 2 \\ 0 & 0 & 6 & 0 & 0 & 2 \\ 4 & 1 & 1 & 0 & 0 & 5 \\ 0 & 0 & 2 & 2 & 5 & 0 \end{bmatrix}$

(d) $\begin{bmatrix} 0 & 2 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 4 & 0 & 3 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 3 & 0 & 3 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 3 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 1 & 0 \end{bmatrix}$