**CSC6010 Programming Assignment #4 Fall’17**

**Dijkstra’s Algorithm Implementation**

**You will implement an algorithm developed by Edsger Dijkstra (1930-2002) to solve the network shortest path problem in this lab.**

**In addition to the algorithm, there are several other things required to make a program that solves the shortest path problem.**

**First, it needs an interface that a user can interact with the program so that the program is "userfriendly."**

**Second, you need to create a "digital" network in the computer memory that the program can access. In this lab, you will first create the GUI of the program. Then, you will read a text file that defines the network into the computer memory and finally implement the Dijkstra algorithm. You have three lab sessions to complete this lab.**

**The learning goals for you are to learn how to read from/write to text files and to use List and HashTable to implement the Dijkstra algorithm.**

**The Dijkstra algorithm, developed a half century ago, finds the shortest path between a vertex and every other vertex on a network graph by constructing and searching a shortest path tree. The algorithm is widely used in modern routing applications that you encounter in everyday life (e.g., on Google map).**

**The algorithm outlined on the wikipedia website has the following steps. Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.**

**Let the node at which we are starting be called the initial node.**

**Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.**

**1. Assign to every node a distance value: set it to zero for our initial node and to infinity for all other nodes.**

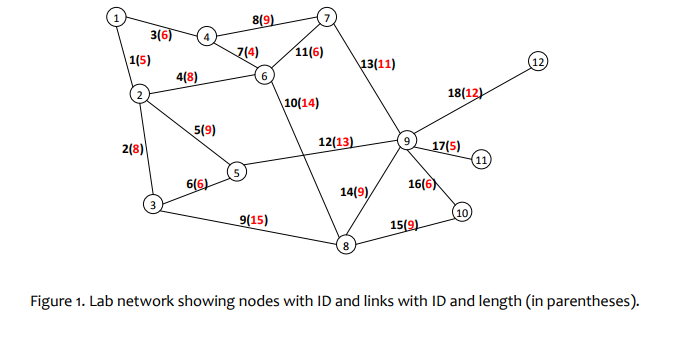
**2. Mark all nodes as unvisited. Set initial node as current.**

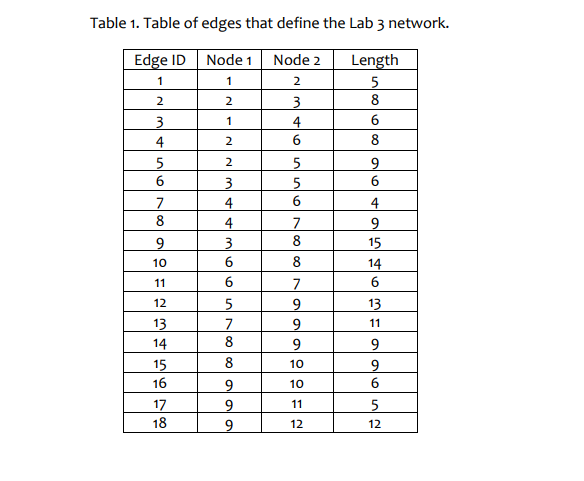
**3. For current node, consider all its unvisited neighbors and calculate their tentative distance (from the initial node). For example, if current node (A) has distance of 6, and an edge connecting it with another node (B) is 2, the distance to B through A will be 6+2=8. If this distance is less than the previously recorded distance (infinity in the beginning, zero for the initial node), overwrite the distance.**

**4. When we are done considering all neighbors of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.**

**5. If all nodes have been visited, finish. Otherwise, set the unvisited node with the smallest distance (from the initial node, considering all nodes in graph) as the next "current node" and continue from step 3.**

**Figure 1 and Table 1 show the network on which you will find the shortest paths.**





**PART 2: FILE I/O**

1. **First you need to create a text file that defines the network depicted in Figure 1.**
2. **We will use a simple edge definition format for this lab. Each edge on the network is recorded as having an ID, Node1, Node2, and Length (Table 1).**
3. **Two-way traffic is allowed on all edges. Use an ASCII text editor (e.g., Notepad.exe) to create the file and save it with a .txt extension name. Please refer to Tuesday's lecture slides on how to use OpenFileDialog and SaveFileDialog controls to read from/write to a text file.**