Network Architect

It is a common problem today to wire offices with Ethernet hubs such that all employees and community devices (printers, servers, shared terminals, etc.) have easy physical access to the network. Networking an office can be very expensive if planned poorly.

For this lab, you will be tasked with developing a tool that can take in data describing Ethernet ports throughout an office and the distance between each. Your application will then produce two sets of output: the minimum amount of cable needed to serial connect the ports and the suggested location of the network hub to achieve the best normalized latency to all connected devices.

**Requirements**

Your application is expected to have the following features:

* Upon startup, your application must prompt the user for the path to a text file containing the network data.
* The data contained in the file will be entered in the following format:
* The very first line will list out every socket ID in the office to be wired. This is done in a comma-separated list.
* IDs can be a letter, number, or alphanumeric string.
* Each subsequent line consists of a comma-separated list of inputs
* The first input of each line is the ID of a wall socket.
* Each input after the ID represents the IDs of other wall sockets and the distances (in inches) between them and the current socket. These inputs are in the format of ID:dist.
* Example: A7,B1:156,C1:72,C2:36,A19:312  
  The above line is the connections from socket A7 to sockets B1, C1, C2, and A19. The numbers after the colon represent the number of inches in cable needed to reach that socket. So, from A7 to B1 requires 13 feet (156 inches) of cable, from A7 to C1 requires 6ft (72 in), and so on.
* It is a valid scenario that not all sockets will be able to directly connect to every other socket. Using the previous example, if there is a socket E3, but A7 has no input for E3, it means that cable cannot be physically laid between A7 and E3 directly (although a serial connection path may still exist).
* Once the data is entered into the app, you will use either Prim’s or Kruskal’s algorithm to produce the minimum spanning tree (MST) of the network. In the case of an office with disconnected sets of nodes, you will create the MST for each set. These MSTs compose the network map for the office.
* In networking, it is often important to manage latency between devices and the central network hub. Placing the hub on the far end of the network map would make it so that sockets nearest the hub location have very low latency, but those on the opposite end of the map would have much higher latency. Latency is a direct result of cable length; the longer the cable distance from the hub to the socket, the greater the latency. You are to find the socket that gives the best normalized latency, meaning that the latency measurements of all sockets are as close to equal as possible. Make sure to indicate the optimal hub socket for each tree in the network map.
* For example, if placing the hub at one socket ends up giving certain sockets a latency of 1, but other sockets a latency of 100, it is likely not the optimal location for the hub. However, if placing the hub on the same network at a different socket gives certain sockets a latency of 42 and others a latency of 47, you’ve definitely found a more optimal location. Remember, we want the latency to all sockets to be as close to equal as possible. It is important to note that each MST within the network map will require its own network hub.
* After your algorithms have finished, you will output the number of MSTs required, the socket set per tree, the optimal hub socket per tree, how much cable each tree requires (in feet and inches, like 55ft 9in for example) , and the total amount of cable needed for all trees in the network (also in feet and inches).

You may do this as a console or GUI application. While you do not need to “draw/print” a visual representation to the UI, it might help to prove your algorithms are correct.

**Rubric**

**Automatic Zero:** Your application has no dynamic way to take in network data.

(10 points) Correctly reads in the graph data (including socket IDs and cable lengths)

(30 points) Correctly uses Prim’s or Kruskal’s algorithm to produce all necessary MSTs and their respective derived data

(10 points) Correctly identifies the optimal hub position to normalize socket latency per MST