EE 450
Spring 2013 Nazarian
Lab1: Packet Routing and Dijkstra's Algorithm

Assigned: Thursday 01/17/2013

Due: 01/27/2013, at 11:59pm (use digital Dropbox) Late submission is accepted for two days with a max penalty of 15% per day. For each day, submission between 12 am- 1am: 3%, 1am-2am: 7%, 2-3am: 12%. After 3am:15%.

The main purpose of this assignment is for you to practice C/C++ programming before working on the final project.

Notes:

This is an individual assignment and no collaborations are allowed. You may refer to the syllabus to review the academic honesty policies, including the penalties. Any questions or doubts about what is cheating and what is not, should be referred to the instructor or the TAs.

Any references used (pieces of code you find online, etc.) should be clearly cited in the readme.txt file that you will submit with your code.

We may pick some students in random to demonstrate their design and simulations.

Post your questions on Lab1 discussion forum. You are encouraged to participate in the discussions. It may be helpful to review all the questions posted by other students and the answers.

If you need to email your TAs, please follow the syllabus guidelines mentioned under the assignments.

1. Problem Definition

Routing

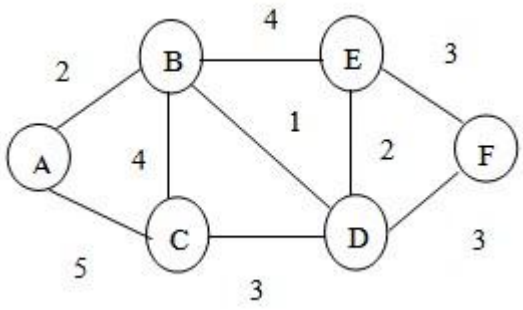
Every communication node (Router is our case) in the network stores information about other communication nodes. This information is used to decide how the packet should be forwarded throughout the network till it arrives to the destination. A routing table should be used by the routers to find the routes.

The routing information includes the following data: the Final Destination Node, the Next Node in the route, and also the path Cost to that next node.

A routing algorithm should be used to create the routing table. In this lab, our goal is to use Dijkstra's shortest path algorithm, which is a single-source shortest path algorithm. This means a router node should be chosen as the source (root in the graph model) and the cost of that source to the rest of the network should be calculated. This also means that Dijkstra's algorithm should be run n times for a network of n routers to find the shortest path from every node in that network to the rest of the nodes. More details are provided in the discussion class.

Building Routing Tables Using Dijkstra's Algorithm

The following shows an example of a network of routers and the application of Dijkstra's algorithm to create the routing tables.



The shortest distance from A to F is also the shortest distance from F to A

Figure 1: The graph model of a network of routers.

Referring to the network of routers illustrated in Figure 1, The Dijkstra's algorithm starts by initializing the cost of the nodes as follows:

Each node only knows its neighbors and the cost to its neighbors. For example, Node A's table like the Table 1.

Node A		
Destination	Line Cost	Next Node
B	2	-
C	5	-

Table 1: The Initialization Outcome

The algorithms continues by going through a loop that eventually produces a tables similar Table 2. In this tables, the cost of A to each node, denoted by C(node), and also its predecessor node also knows as the parent node, denoted by P(node), is recorded. For example, the shortest path from A to F is A->B->D->F and with the cost of D as C(D)=3 and its parent node as P(D)=B.

Step	N	C(B), P(B)	C(C), P(C)	C(D), P(D)	C(E), P(E)	C(F), P(F)
0	A	2, A	5, A	∞	∞	∞
1	AB		5, A	3, B	6, B	∞
2	ABD		5, A		5, D	6, D
3	ABCD				5, D	6, D
4	ABCDE					6, D
5	ABCDEF					

Table 2: The Loop Outcome

Given Table 2, the path from node A to every other node is known. To identify the next node in the path (which is necessary for routing tables) a backward traversal should be used from every node to the source. For example, to know to next node from A to F, a backward traversal would identify A->B->D->F as the shortest path, which means the next node is B. Dijkstra's algorithm should be executed for every node in the network as the source (e.g., 6 times for the network of Figure1) and backward traversal would reveal the next node. The following routing table can then be generated:

Node A		
Destination	Line Cost	Next Node
B	2	-
C	5	-
D	2	B
E	2	B
F	2	B

Table 3

2. Lab1 Tasks

a. In this lab, you will be given all the initialized routing tables for all nodes in text files.

b. You are responsible for implementing the Dijkstra's algorithm and completing the routing tables.

c. You should develop an interface for your code, so the user can enter the source node.

d. NOTE: The node names in this lab should be case-insensitive.

e. Your code should be able to print on the screen, the shortest path from the source to any other node in the following format:

The Shortest Path from A to x	Total Cost
A to B: AB	2
A to C: AC	5
A to D: ABD	3
A to E: ABDE	5
A to F: ABDF	6

f. Compile your code on nunki.usc.edu. You can use gcc or g++ compilers if you want.

g. Write a report that includes snapshots of your input and output screen for the provided test cases: from sources A, B, C, D, E, and F.

3. Submission Rules

a. You should submit a zip file that contains all your code files and report as well as a readme.txt file. The name of the zip file should follow this pattern: Firstname_Lastname.zip.

b. We will compile and run the code on nunki.usc.edu. So please make sure to test your code on nunki before submission.

c. Along with your code files, include a readme.txt with the following information.

i. Your full name and student ID

ii. Compilation steps to run your programs.

iii. The status of your code (compilation error, working, or not working)

iv. Additional info, if you think necessary.

d. You should use the DEN digital dropbox to submit your zip file. In the "File Information" part, use your file name as the "Name". Note: You should click on "Send File" with your zip file attached, otherwise it won't be submitted.