Programming Assignment – CMSC 481 – Fall 2014

Due Wednesday 11/26/2014 11:59pm

Objective: This assignment will give you a chance to gain your competency in Dijkstra algorithm and also understand how to use C programming language to program the link state protocol.

Note: This assignment will be coded in C language. Related topics are basic functions and file I/O operations. It may take a long time to complete. Please get started early.

Description: You are going to write a program in C language to demonstrate Dijkstra algorithm. You will be given an input file in which you need to interpret its content and keep those numbers in your favorite data structure. Your program will allow the user to find out the shortest path from any node to any node defining in the input file. The source and destination will be modified in order to test your program. Eventually, you will provide the output as text file mentioned in the given rules below.

Rules:

1. Maintain Dijkstra algorithm. The pseudo code for Dijkstra algorithm is given below:

$G = (\mathcal{N}, \mathcal{A})$. It requires that all arc weights are non-negative $\forall (i,j) \in \mathcal{A} : w_{ij} \geq 0$.

Dijkstra's Algorithm involves the labelling of a set of permanent nodes P and node distances D_j to each node $j \in \mathcal{N}$. Assume that we wish to find the shortest paths from node 1 to all nodes. Then we begin with: $P = \{1\}$ and $D_1 = 0$ and $D_j = w_1 j$ where $j \neq 1$. Dijkstra's algorithm then consists of following the procedure:

1. Find the next closest node. Find $i \notin P$ such that:

$$D_i = \min_{j \notin P} D_j$$

2. Update our set of permanently labelled nodes and our nodes distances:
   $P := P \cup \{i\}$.
3. If all nodes in $\mathcal{N}$ are also in P then we have finished so stop here.
4. Update the temporary (distance) labels for the new node i. For all $j \notin P$

$$D_j := \min[D_j, w_{ij} + D_i]$$

5. Go to the beginning of the algorithm.

2. You have to use file I/O operations in order to read your graph input and export your calculation from your program to file name "output.txt". This file output.txt must have the information as below:

a. The summary of traversal: From where to where? How many nodes traversed? Which nodes your program traverse? Total distance?

b. Routing table of each node: You must print out the routing table of each node. The idea of maintaining routing table as following links (next page):

http://cs.gettysburg.edu/~jfink/courses/cs322slides/3-19.pdf

http://www.eoinbailey.com/content/dijkstras-algorithm-illustrated-explanation

Please note that example 1 is clearly shown what you need to print to the terminal for your output. You need to add these columns (destination, next hop, and cost) in order to maintain link-state protocol (mandatory). Example 2 has "Distance to Node from Node 'a'" column for the distance from the source and the "visited" column is an option to keep track of your algorithm. For this example, you need to add the next hop column to be similar to example 1.

At the very end step of your calculation, each node will have all latest updated entries from your algorithm (in the above example link, it is only node 1 or node a). You need to print out all given nodes from the input file (interpreted by your file I/O operation e.g. 6, 7 or 8 nodes/tables)

c. The structure of your output file (output.txt) might be as below:

---------- (beginning of summary)

(Summary of your shortest path from selected source and destination)

Source ? → Destination ? (e.g. 1 → 5)

Traversal (e.g. 1 →2 → 4 →5)

Total distance: _____

---------- (end of summary and separation)

(Routing tables in order)

Routing table of node 1

(at least has 3 columns; To, Next hop, Distance, Visited (option) – no need to create the table border)

Routing table of node 2

…..

Routing table of node 3

…..

Table of last node

Structure of the input file

Your input file, graph.txt, has to be specified as: nodeName$connectingNode$edgeWeight

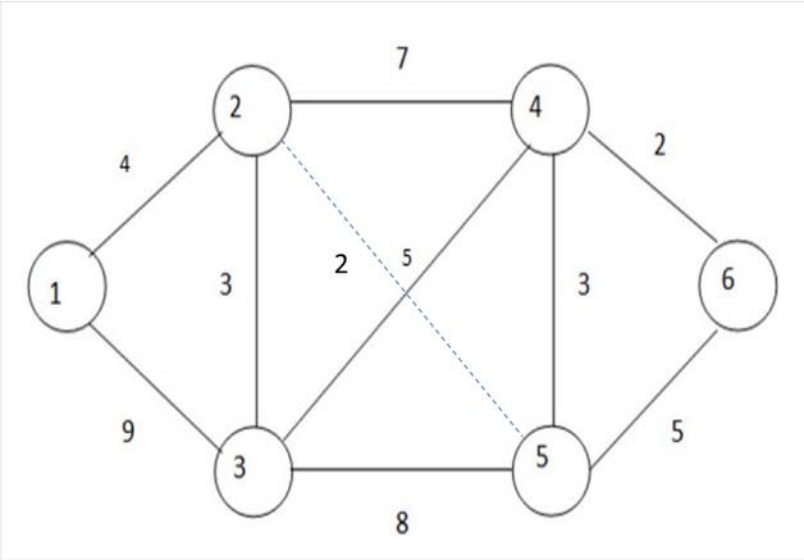
You can assume that the text file has one record per line; all the fields are presented (i.e. nodeName, connectingNode, etc.). They are separated by a $, and that all the fields can be converted to numbers.

Below is the file used for testing your code as basic test case. There will be no negative weight edge for this input. Please make sure your algorithm works efficiently because some lines may be added into the input file as an additional edge or node in order to test your program.

For example, line 1 has 1$2$4. That means node 1 connecting to node 2 with the weight of 4. Please note that there won't be 2$1$4 in the input file since it is the same as 1$2$4 but you might need to maintain routing table as required. You may represent the given information as a graph for your abstraction.

Example of the input file/graph (graph.txt)

```
1$2$4
1$3$9
2$4$7
2$3$3
2$5$2 (may be added)
3$4$5
3$5$8
4$5$3
4$6$2
5$6$5
Source$1
Destination$5
```



Submission

When you've finished your assignment, please submit your program on blackboard. You need to submit as a package (zipped/compressed). Your file must contain your .c file, readme.txt (including the status of your program), input.txt (your test case), output.txt (result from your test case)

Please be aware that you must submit the file before the deadline in order to access the assignment's link. You can upload file as you can. The latest upload will be graded.

Note: Please do not submit via email to both professor and TAs because it will not be graded!

Guideline

You can refresh Dijkstra Algorithm from the below link. Apply this explanation to your favorite data structure to keep visited nodes e.g. typedef. Pointers might be needed. The graph in this video is a directed graph but your input graph is an undirected graph.

https://www.youtube.com/watch?v=8Ls1RqHCOPw