Computer Networks CSE 4344
Project 3
Routing using Dijkstra's Shortest Path Algorithm

Instructor: **Sajib Datta**
GTA (Section 002): Jees Augustine

Fall 2018

"A wise man can learn more from a foolish question than a fool can learn from a wise answer" ***Bruce Lee***

Objectives

- To understand the Dijkstras algorithm.
- Establish a routing table using Dijkstra's algorithm.
- Publish all routes using a command '`list_all_paths`'.
- Publish results to a particular destination using user command '`list_path_to`'.

Due Date

December 03, 2018 (Monday) 11:59 PM[1]

Submission Guidelines

- This is an individual project.
- Submit a single zipped file with the naming convention,

$$< Dijstra\_lname\_UTA\_id > .zip$$

- Your submission should have the following items to be considered for evaluation,

[1] All Submissions should be completed through BlackBoard

(a) You submit the source code.

(b) **Clear instructions** on how to execute your code on any given system.

(c) a *readme* file, which clearly mentions the underlying assumptions and system requirements.

- Make sure you write your **names** and your **UTA IDs** in the beginning of the source code that you are submitting.
- Make sure that submissions of the zipped file is through *UTA BlackBoard*[2].
- Late submission will be accepted only through email [3]. Address all emails to your Professor and add a copy to TA. I have simplified this for you here, just click me to do what all are meentioned above
- There will be a reduction of 10 points for the first day and 5 pointers each for each subsequent day.
- screen shots of execution of commands, clearly marking the path and nature of commands

Problem Statement

Use the Dijkstra's algorithm to find the shortest path between each pair of nodes in a network. Your graph is a `undirected graph` defined by a set of Nodes-$\mathcal{N}$, a set of Routers-$\mathcal{R}$, and set of Edges-$\mathcal{E}$ that connects $\mathcal{N}$ and $\mathcal{R}$. Mathematically your graph is defined as follows,

$$\mathcal{G} = ((\mathcal{N} \cup \mathcal{R}), \mathcal{E})$$

Program Input File Location

Your program will take three input files to complete this assignment `Nodes.txt, Routers.txt,` and `Edges.txt`. All your input files should be in a folder named '`data`' where your program is located. In other words your program should look for its input in a folder `data` in your current directory. All your input files `Nodes.txt, Routers.txt` and `Edges.txt` should be located within this folder. Your output, the `Routes.txt` file should be a folder called '`paths`' and it should also be also in a location of your program.

```
your program folder
└── dijkstra.(py/java/c/c++/*)
```

[2] Please strictly follow the naming convention of the zipped file

[3] Please mention the subject line as *"CSE 4344 - Project 3"*

```
├── readme.txt
├── data
│   ├── Nodes.txt
│   ├── Routers.txt
│   └── Edges.txt
└── paths
    └── Routes.txt
```

Programming Language

You might choose any programming language of your choice however, using Python will be great option and simplifies your code a lot. You might also try Java or C or C++. Any other language is also fine but you should be able to give me a working demonstration of your code immediately after your submission.

Usage of Available Packages

Please try to use your own code to simulate the Dijskstra's algorithm. You might use packages for parsing the input files but **should not be extended to finding the shortest paths**.

Program Inputs

Your program should work with any given inputs and not just for your inputs. We will try to validate your algorithm with our own input files, which will follow the suit below.

You should have a file `Nodes.txt` which lists all the nodes in the network. Nodes file contains all the nodes which can act as both source as well as destinations. So if you have 3 nodes $\mathcal{N} = \{0, 1, 2\}$ in the network you can have $3(\binom{3}{2})$ source-destination pairs namely $\mathcal{SD} = \{(0, 1), (0, 2),(1, 2)\}$[4]. You will have to find the paths between each of the elements in $\mathcal{SD}$, which is in-turn a source-destination pair. Each of the node name(preferably number starting from 1) is written down to a line in the file `Nodes.txt`. For example your `Nodes.txt` should contain entries like this.

1
2
.. .. ..
10
.. .. ..

[4]This number 3 is just an example, we should be able to give any number of nodes in the network

There should be a routing node file which contains all the routers name in the network namely `Routers.txt`. This file looks exactly like `Nodes.txt`, however, the entries will be the names of the routers and should look like this.

R0001
R0002
.. .. ..
R0010
.. .. ..

You should have an edge file, `Edges.txt` where in you define all the edges in your network and its weight separated by spaces. For example your `Edges.txt` should contain entries like this.

10 20 05
12 34 08
09 53 15
.. .. ..
.. .. ..

First line in the file is interpreted as *source 10* is connected to *destination 20* and weight of the edge is *5*.

Sample Inputs and Sample is provided along with the code

Program Outputs

Your output file should be a simple file, `Routes.txt`, which contains all the routes between each of the source-destination pairs that you have from the `Nodes.txt`. it should have the name of the source-destination pair followed by the actual path. The file will look like this.

01 02 01 R0001 R0002 R003 02
11 27 11 R0023 R0021 27
09 53 09 R0012 R0083 R0061 R0038 09
.. ..
.. ..

The first two entries in each line represents the source-destination pair and in above file the first like corresponds to a path between source 01 and destination 02and the path is {01 R0001 R0002 R003 02}. The traffic from 01 should first go to router 'R0001' and then to 'R0002' and then to 'R0003' and reaches its destination 02. The path is actually like this, 01 → R0001→ R0002 → R0003 → 02. This route is found out by your Dijkstra's algorithm over your network and by the edge weights.

Prohibited Entries - Node in the middle

You should ensure that all the traffic is going through routers and never through a source or destination. We will never have path with a node in the middle of a path. None of the paths will have any entries from the `Nodes.txt`. It will have entries only from the `Routers.txt` in the middle. For example in your you will never have any path in `Routes.txt` like this, {05 02 05 R0001 20 R003 02}. In this path, node 20 appears in the path between 05 and 02 which is actually conceptually wrong as routers are routing the traffic for you and not the nodes or end point users like anyone using the internet.

Prohibited Entries - Routers as Sources, or Destinations, or Source-Destination Paris

You should also ensure that you are not having entries for router to router in the file `Routers.txt`. You will never have a path recorded where your router appears either as source or as destination or both. Hence the following entries are not allowed.

- R0010 02 R0010 R0001 R0002 R003 02
- 01 R0123 01 R0001 R0002 R003 R0123
- R0010 R0010 R0010 R0001 R0002 R003 R0010

In the first line R0010 is the source which is not not allowed. In second line R0123 is the destination which is also not allowed. In third line both source, R0010 and destination R0123 as it Routers can never be sources or destinations.

Program Interface

Program once initiated through command line or visual interface should run the Dijkstra's algorithm and should create the file **Routes.txt**. In addition to generating this output your program should wait in the interface (command line or visual) to accept two commands. (1) command to list all paths, `list_all_paths` (2) command to list a path between a given source and destination, `list_path_to`. A user should be able to choose between these two and whould be able to get the results from your program.

Program Testing and Evaluation

We(Prof and TA) will provide you with a random set of input files, {`Nodes.txt, Routers.txt, Edges.txt`} and your answers should **exactly match with our answers**(order of paths in the file does not matter, however the paths should exactly match with ours). **These input files will not be provided apriori**. We will feed this input to your program while evaluating.

Grading Rubric

Will be filled out later but sooner.

Wish You All The Best