Report for project 2

Xunjie Liu 1613204

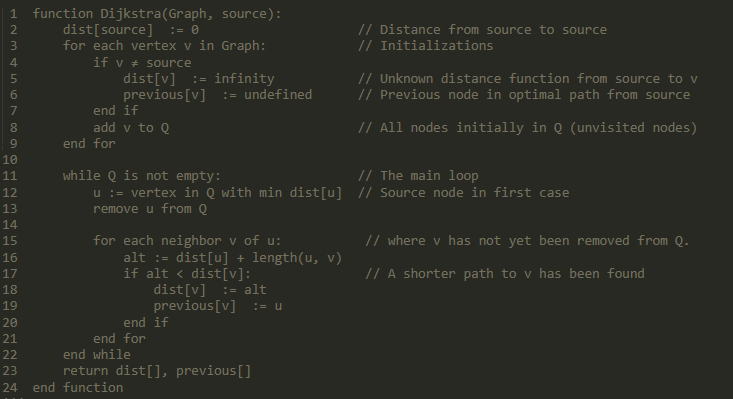
For question 1:

‘get\_graph(file)’ function will read txt file to return a dictionary describing this graph. The format of the dictionary is showed below: 

{Node\_name: [[neighbor\_name1, cost], [neighbor\_name2, cost]] and so on.

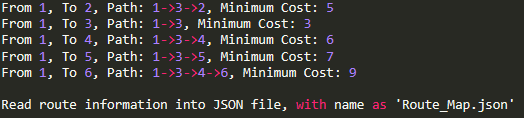
For question 2:

‘dijkstra(graph, source='1')’ function will read graph generated from question 1, and the source is default to ‘1’, and it mainly follow the psudocode showed below:



In order to realize priority queue, I used heapq module in python. This function will return two dictionaries: distance and previous.

For ‘distance’, its keys are destination nodes and values are corresponding minimum cost for the source. For ‘previous’, its keys are destination nodes and values are previous nodes in the shortest path from the source. I generated shortest paths from source to destination nodes using ‘distance’ and ‘previous’, then write them into Route\_Map.json file. In JSON file, it is a dictionary whose format is “route\_map[node] = [path, cost]”

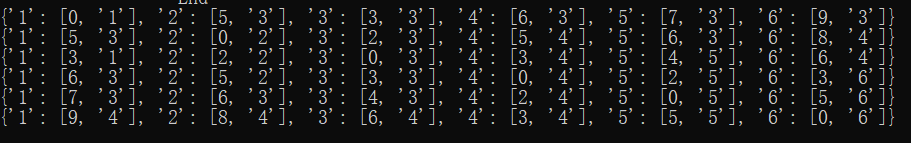


For question 3:

Protocol for question 3:

1. Port number of each process (router) is equal to 12000 + node\_name.
2. Routers/Processes should only actively start a TCP connection to Routers/Processes with larger number node name.
3. Once a Router/Process receive a TCP connection, save data into public container and send its own routing table.
4. For each process, the input parameter is a Router object which has the same attributes and relax (update the routing table) as the real router, and this process will generate two threads, one for listening, another for actively start TCP connections.
5. Divide neighbours of a Router into two arrays: connect\_to and connect\_in. Connect\_to stores the neighbours which has larger number node names. Connect\_in stores the neighbours with smaller number node names.
6. For Listener thread in each process, once it confirms that every neighbour in Connect\_in has actively sent TCP connection to it, this thread will end.
7. For Messager thread who actively starts TCP connections, after it sends TCP connections to every neighbour in Connect\_to, this thread will end.
8. For Messager, if it cannot connect to the neighbours, sleep for 2 seconds and retry.
9. Every time Listener or Messager thread receives a routing table from neighbours, immediately do the update operation in Router.
10. In order to initialize a Router object, write a blank routing table into a JSON file with a file name as its node name.

The final result in JSON file should be similar to this:



Which means: Destination: [cost, next\_node]

To run this python file, all necessary codes have been integrated into main\_1() and main\_2().