|  |  |  |
| --- | --- | --- |
|  | Faculty of Computing, Engineering and Science |  |

**Assessment Cover Sheet and Feedback Form** 2019-20

|  |  |  |  |
| --- | --- | --- | --- |
| Module Code:  CS3S661 | Module Title:  Computer Networks | | Module Team:  Dr Emlyn Everitt |
| Assessment Title and Tasks:  **“Finding the shortest available paths and calculating Max flow of a Network”** | | | Assessment No.  1 |
| Date Set:  **23-Sep-19** | | Submission Date:  **10-Jan-20** | Return Date:  **07-Feb-20** |

**IT IS YOUR RESPONSIBILITY TO KEEP RECORDS OF ALL WORK SUBMITTED**

|  |
| --- |
| **Marking and Assessment** |
| This assignment will be marked out of 100%  This assignment contributes to 50% of the total module marks. |
| **Learning Outcomes to be assessed** (as specified in the validated module descriptor [https://icis.southwales.ac.uk/](https://icis.southwales.ac.uk/studentmodules/14926/studentmodulespecifications) ):  1) Compare and analyse the theoretical and practical issues of network infrastructure. |
| *Provisional mark only: subject to change and / or confirmation by the Assessment Board* |

# Assessment – Detailed Requirements

***“Finding the shortest path and calculating Max flow of a Network”***

Optimising networks for maximum data throughput is crucial for the delivery of high quality network services; and Ford-Fulkerson algorithm will find the optimal max flow configuration for any given network based on available network capacity.

**Your task is to implement a solution based on the pathing algorithm developed in tutorials. No other solution will be accepted.**

**First Deliverable: Dijkstra [70%]**

Base your solution on the following code/ data structures:

infinity = 1000000

invalid\_node = -1

class Node:

previous = invalid\_node

distfromsource = infinity

visited = False

class Dijkstra:

def \_\_init\_\_(self):

'''initialise class'''

self.startnode = 0

self.endnode = 0

self.network = []

self.network\_populated = False

self.nodetable = []

self.nodetable\_populated = False

self.route = []

self.route\_populated = False

self.currentnode = 0

def populate\_network(self, filename):

'''populate network data structure'''

def populate\_node\_table(self):

'''populate node table'''

def parse\_route(self, filename):

'''load in route file'''

def return\_near\_neighbour(self):

'''determine nearest neighbours of current node'''

def calculate\_tentative(self):

'''calculate tentative distances of nearest neighbours'''

def determine\_next\_node(self):

'''determine next node to examine'''

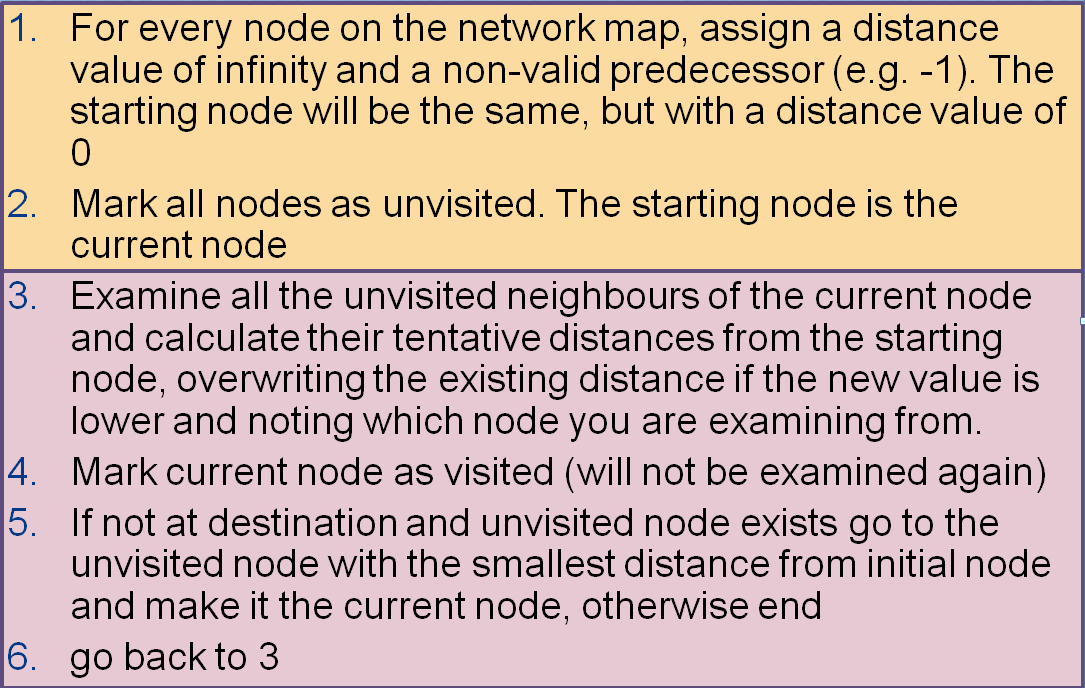
def calculate\_shortest\_path(self):

'''calculate shortest path across network'''

def return\_shortest\_path(self):

'''return shortest path as list (start->end), and total distance'''

and the following pseudo code:



Code your own version of Dijkstra’s algorithm using the above that is capable of discovering the shortest path across a network (graph).

Towards this end, your algorithm should be able to:

* Read in a file called “network.txt” – a multi-line file with node, nearest neighbour and distance metric information
* Read in a file called “route.txt” – a single line file with start and destination information, where the line is of the form “B>F” (any start / end combination is possible)
* Print out to the screen the shortest path and total distance
* During this process, both ‘route.txt’ and ‘network.txt’ should remain unchanged.

Your code should be constructed using best programming practice, including the appropriate use of functions and data structures. Your code should also be suitably commented to ensure readability and to demonstrate your understanding of how it works.

**Second Deliverable: Max Flow [30%]**

Using your Dijkstra’s algorithm you created for part 1, create a solution for Max Flow. This should be done in a fashion similar to the one demonstrated in lectures.

Towards this end, your algorithm should be able to:

* Read in a file called “network.txt” – a multi-line file with node, nearest neighbour and distance metric information
* Read in a file called “route.txt” – a single line file with start and destination information, where the line is of the form “B>F” (any start / end combination is possible)
* Print out to the screen every path discovered during the calculation, along with its associated bottleneck flow value
* Then print out the total max flow for the network
* During this process, both ‘route.txt’ and ‘network.txt’ should remain unchanged.
* A complete solution should first instantiate the Dijkstra Object and use that to display its results for a given route and network, and then instantiate the MaxFlow object to display its results for the same route/ network; all of which should be done in the main function of the program.

Base your Max Flow solution on the following code/ data structures:

class MaxFlow(Dijkstra):

'''inherits from Dijkstra class.

Expose and override Dijkstra methods and add new ones where required, but must use original Dijkstra’s algorithm as part of the calculation'''

## Submission

You need to submit of your python file using the following naming convention: *<insert student number>*.py to Blackboard. **Do not upload your network.txt or route.txt files.**

## Demo

All submissions will be marked via demo at the earliest opportunity after submission and during the scheduled tutorial slots. Any unjustified delays in demoing will result in lost marks.