**INFORMATICS INSTITUTE OF TECHNOLOGY**

**IN COLLABORATION WITH**

**UNIVERSITY OF WESTMINSTER (UOW)**

**BSc Computer Science**

**5SENG002C.2 – Algorithms: Theory Design and Implementation**

Coursework

**Module Leader: Sudharshan Welihinda**

**UOW ID:** w12345

**Student ID:** 12345

**Student Name**: Sabiq Sabry

**The choice of Algorithm & Data Structure**

For the estimation of the max flow network the Ford-Fulkerson algorithm was used. Breadth First Quest (BFS) has been used to figure out if a source to sink path occurs. BFS was selected because BFS often takes the trail with the least number of edges. Hence, the time complexity of the worse instance can also be minimized.

In this case, the greedy algorithm was adopted. For the queue generated in the Breadth First Search (BFS) process, a LinkedList (queue) was used. The first part of the queue was returned from the queue and removed by BFS poll() method from the LinkedList

**Explaining the Algorithm on the smallest benchmark**

- My code contains 3 classes which are Main, BFS and Ford Fulkerson class.

- The Main Class contains all the necessary code from reading the data from the text file, creating graph/ adjacent matrix and displaying the maximum flow through the graph.

- I have made use of the Adjacent Matrix inorder to create the graph matrix representation. Since this is a 2D graph matrix the index values shows the edge value and the value on the matrix will show you the capacity between the edge in those indexes.

- The user is also given the right to modify the graph by adding extra edges and also allowed to remove edges from the graph as well. By doing so the maximum flow will also change accordingly.

- Once all the data is read from the text file and a graph is created, its then sent to the ford Fulkerson algorithm class in order to perform the max flow finding.

**-** Its very important to note that the residual graph will be an exact copy of the original graph at the starting stage.

**-** The traversing algorithm BFS will be used to find if any augmenting path is present from the source to sink.

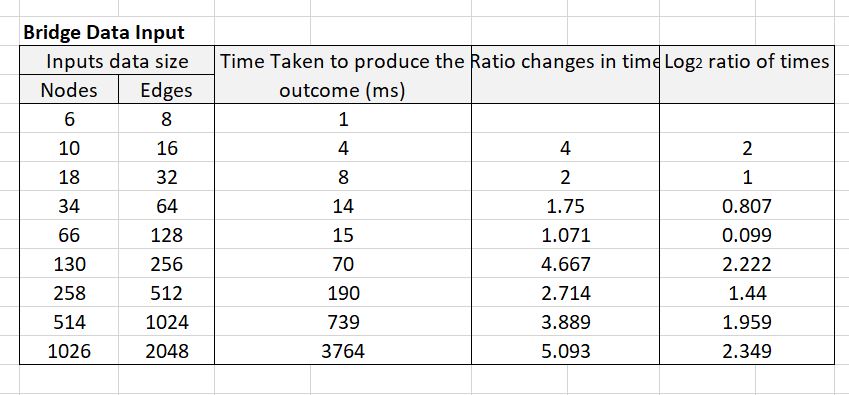
- The parent array is used to store all the path found during the BFS procedure.

**-** We should set the max flow to 0 at the beginning because currently there is no flow found

**-** Once the BFS finds the path then the bottleneck value or capacity for that augmenting path is found and then the residual graph is updated accordingly as well as the max flow also gets updated by adding this bottleneck capacity to it.

**-** Once everything goes on well then the maximum flow for the graph is displayed to the user.

**Performance Analysis of Algorithm.**



Using the Doubling Hypothesis, we can clearly see that the log ratio value rounds down to a value of 2.

Since the value of the log ratio found is equal to 2, we can say that the Big O notation for my algorithm is O(n2).

I have used for loop within a for loop (nested ones) which also confirms that the time complexity for this algorithm has to be O(n2).

Big O notation = O(n2)