Informatics Institute of Technology

Algorithms : Theory and Implementation

5SENG002C.2

Coursework-01

Name	: J.Thuwarakan
IIT ID	: 2019795
UoW ID	: W1790265

01 - Project overview

In here I crate a class named Flow_Network which is contained several method and several variables. This class give the output in a adjacency matrix to do that first this class have a two-dimensional array named **flowEdge** this array store the details of the edge and the capacity then once if we want we can print them. Then this class have a variable named **totalNodes** this variable contain the count of total nodes. Also we have some variables to save the start and end node of a flow network. Also create a two-dimensional array named **residualNetwork** for save the details of the residual values. This class contain some method also first this this class contain a **addEdge** method this method have a 3 parameters those are fromeNode, toNode, capacity this method help to add edges to a nodes. Also this code have a **removeEdge** method this method use to delete the edge then this code have a **printFlow** method this method print the network flow as a adjacency matrix to do that I used a nested for loop.

The main task of this course work is finding the maximum flow for a particular flow network for that we want to implement to algorithms one for calculate the max flow and one for finding the augmenting path. For finding augmenting path we learn two algorithms in here I use Breadth First Search. For do that I have a method named **searchNode.** This method search there are augmenting is here or not and return a Boolean value.

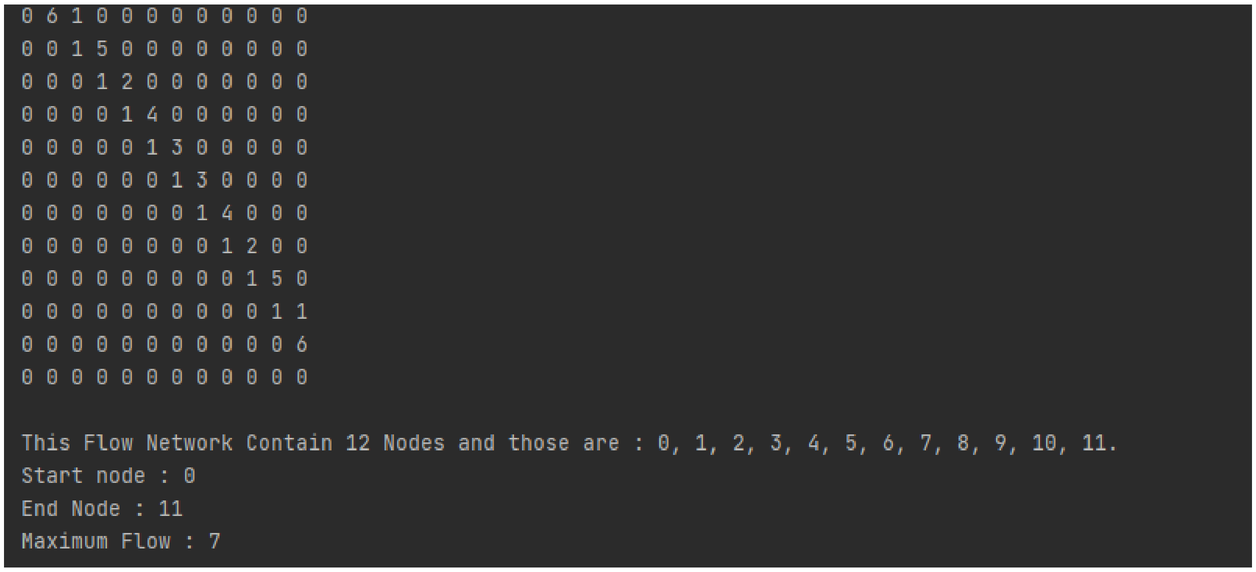
To find the max flow in here we use Ford-Fulkerson Algorithm this algorithm will help to overcome the minimum cut problem. So in here we use Edmonds-karps idea in ford Fulkerson. This idea is using the Breadth First search with ford-fulkerson to calculate the maximum flow. In here I create a method named getMaxFlow this method will return the maximum value of that flow network

02 – Run of an Example

In here I am going run the ladder_2.txt file and show the output this file contain 12 nodes full file input is shown blow :-

```
12

0 1 6

0 2 1

1 2 1

1 3 5

2 3 1

2 4 2

3 4 1

3 5 4

4 5 1

4 6 3

5 6 1

5 7 3

6 7 1

6 8 4

7 8 1

7 9 2

8 9 1

8 10 5

9 10 1

9 11 1

10 11 6
```

Output :-



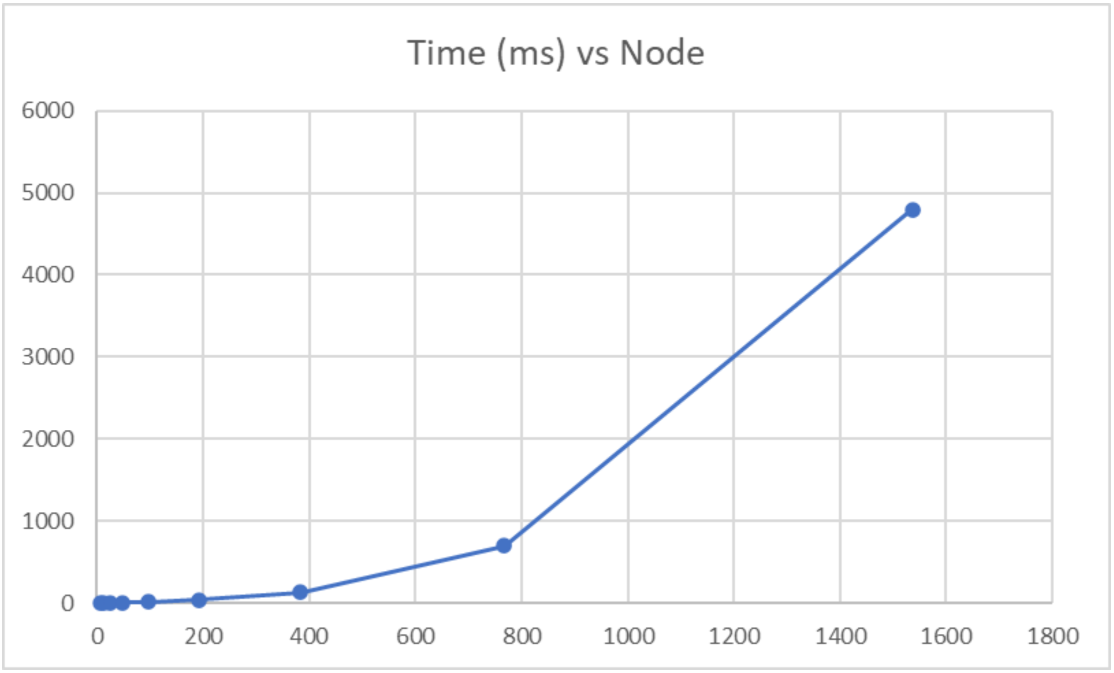
```
0 6 1 0 0 0 0 0 0 0 0 0
0 0 1 5 0 0 0 0 0 0 0 0
0 0 0 1 2 0 0 0 0 0 0 0
0 0 0 0 1 4 0 0 0 0 0 0
0 0 0 0 0 1 3 0 0 0 0 0
0 0 0 0 0 0 1 3 0 0 0 0
0 0 0 0 0 0 0 1 4 0 0 0
0 0 0 0 0 0 0 0 1 2 0 0
0 0 0 0 0 0 0 0 0 1 5 0
0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 6
0 0 0 0 0 0 0 0 0 0 0 0

This Flow Network Contain 12 Nodes and those are : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11.
Start node : 0
End Node : 11
Maximum Flow : 7
```

First we are printing the adjacency matrix then print the details of start node, end node and maximum flow

03 – performance analysis

The following diagram represents the Empirical study on the Algorithm. The time increases with the increase of vertex. Because of the differences in capacity and edges connectivity there were different outcomes of time for the same vertex count. There for More than 5 times a vertex was tested and the average is represented in the below figure.

Node	**Time (ms)**					
	Test 01	**Test 02**	**Test 03**	**Test 04**	**Test 05**	**Average**
6	8.8	7.9	7.6	8.6	7.1	8.0
12	7.1	7.6	9.2	9.8	8.2	8.4
24	8.9	10.1	7.5	8.9	10.2	9.2
48	11.3	10.4	11.7	10.0	9.8	10.7
96	10.9	16.8	15.2	15.6	19.4	15.6
192	35.2	40.0	34.0	44.1	33.2	37.3
384	106.7	129.5	140.5	147.1	137.2	132.2
768	665.8	634.3	456.3	897.6	834.1	697.6
1536	5030.1	5360.6	2984.6	5342.5	5248.3	4793.3



The one side of table represent the node count and other side represent the time in milli second. To do this we run ledder_1.txt to ledder_9.txt and calculate the average time and we use Excal sheet to build the graph. So this graph suggest the edge capacity, edge count play a huge role in calculate the maximum flow.

The complexity to do a BFS O(E).The complexity of FF implementation is going to be O(E*No of Path). Then ,the number of times an edge can be critical(Edge holding the bottle neck capacity) should be found out. The BFS returns a shortest path first then gives longer paths. An edge can become a critical edge V times. If all edges becomes a critical edge then it will be VE . Therefore we can say that O(E*VE). Since we are doing a BFS search it will be O(V2). Thus giving the final notation as O(EV3).