



**INFORMATICS  
INSTITUTE OF  
TECHNOLOGY**

**UNIVERSITY OF  
WESTMINSTER**

## **Informatics Institute of Technology**

Department of Computing  
(B.Sc.) in Software Engineering

# **Algorithm**

**Module Leader: MR. Sivaraman Ragu**

**Module code: 5COSC0012C.2**  
**Tutorial Group F**

**Student Name: W.T.N.Peiris**

**Student ID: 2018422**

**UOW ID: W1761915**

**Submission Date: 08/04/2021**

## Contents

<b>Data structure &amp; Algorithm .....</b>	<b>3</b>
<b>Performance of the Algorithm .....</b>	<b>4</b>

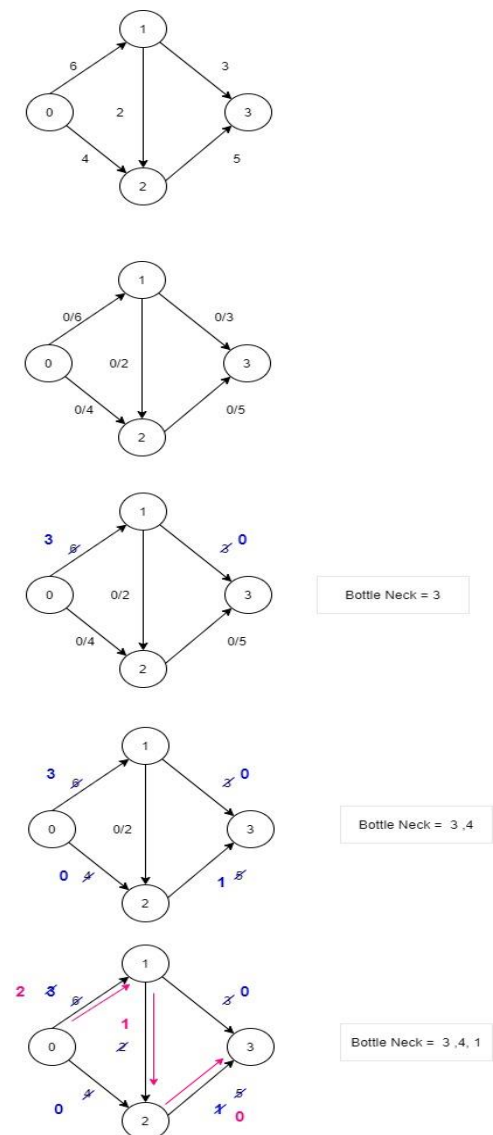
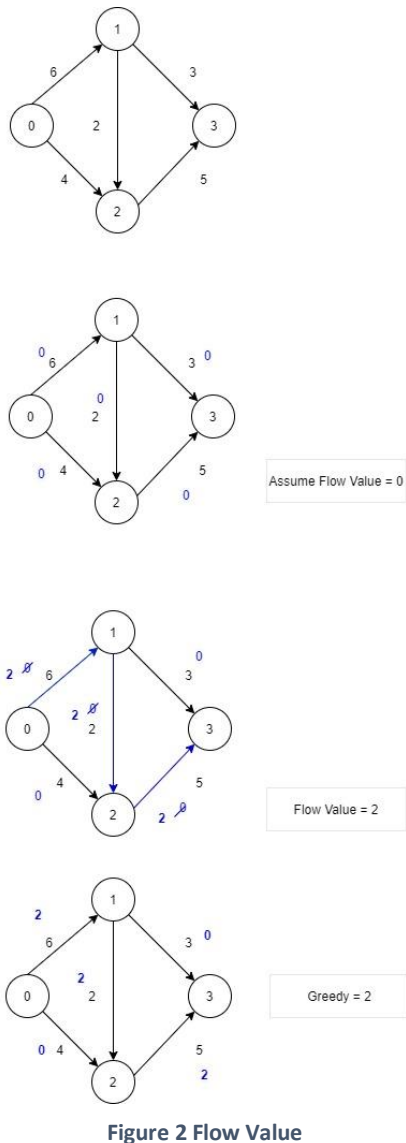
## Figures

<b>Figure 2 Maximum Flow Value.....</b>	<b>3</b>
<b>Figure 1 Flow Value.....</b>	<b>3</b>
<b>Figure 3 Flow Value.....</b>	<b>4</b>
<b>Figure 4 Coding Output.....</b>	<b>4</b>

## Data structure & Algorithm

Breadth First Search (BFS) is the data structure and the Ford-Fulkerson is the algorithm which is used to implement the code. BFS is used because of it is a traversal algorithm that starts traversing the graph from the root node and explores all the neighboring nodes. Then it chooses the closest node and explores all the nodes that have to be discovered. The algorithm repeats the loop for each closest node until it reaches the target. The maximum possible flow between a source and a sink in a network is computed using the Ford-Fulkerson algorithm. The Ford-Fulkerson algorithm make use of residual graphs which are an extension of the naive greedy approach that allows undoing operations.

Finding the flow value with Ford-Fulkerson algorithm is shown in below (Figure1). Finding the maximum flow value is shown in below (Figure 2).



The final Bottleneck values should be added to find the maximum flow from figure 2. So, according to the diagram the maximum flow value should be eight (8).

## Performance of the Algorithm

```
2018422 src \ FlowNetwork
Project \ FlowNetwork.java
Run: FlowNetwork
"C:\Program Files\Java\jdk1.8.0_251\bin\java.exe" ...
Nodes 12
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Edge from node 0 to node 1 with capacity 6
Edge from node 0 to node 2 with capacity 1
Edge from node 1 to node 2 with capacity 1
Edge from node 1 to node 3 with capacity 5
Edge from node 2 to node 3 with capacity 1
Edge from node 2 to node 4 with capacity 2
Edge from node 3 to node 4 with capacity 1
Edge from node 3 to node 5 with capacity 4
Edge from node 4 to node 5 with capacity 1
Edge from node 4 to node 6 with capacity 3
Edge from node 5 to node 6 with capacity 1

Edge from node 4 to node 5 with capacity 1
Edge from node 4 to node 6 with capacity 3
Edge from node 5 to node 6 with capacity 1
Edge from node 5 to node 7 with capacity 3
Edge from node 6 to node 7 with capacity 1
Edge from node 6 to node 8 with capacity 4
Edge from node 7 to node 8 with capacity 1
Edge from node 7 to node 9 with capacity 2
Edge from node 8 to node 9 with capacity 1
Edge from node 8 to node 10 with capacity 5
Edge from node 9 to node 10 with capacity 1
Edge from node 9 to node 11 with capacity 1
Edge from node 10 to node 11 with capacity 6

Source : 0
Sink : 11
Max Flow :7
Time :20
Process finished with exit code 0
```

Figure 4 Coding Output