**Informatics Institute of Technology**

**In collaboration with**

**University of Westminster**

**BEng (Hons) in Software Engineering**

# 5SENG003C.2 Algorithms: Algorithmic analysis of network flow

# Network Flow Algorithm Report

**Nimshan Munasinghe**

**20230020 / w2052877**

**SE - Group 08**

**thineth.20230020@iit.ac.lk**

## 1. Choice of Data Structure and Algorithm

### 1.1 Data Structure

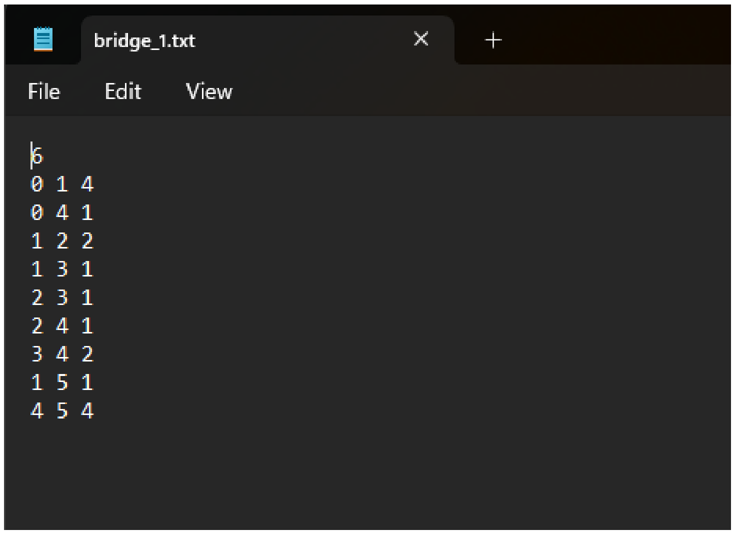
I represented the network using an adjacency list, where every node has a record of its outgoing edges. Backward edges are included to deal with residual capacities, and every edge has the source, target, capacity, and current flow stored. The adjacency list was chosen because it is easy to modify flows on execution, has quick access to outgoing edges, and space usage.

### 1.2 Algorithm: Ford-Fulkerson with DFS

The Ford-Fulkerson algorithm with Depth First Search (DFS) was employed to calculate the maximum flow, which repeatedly finds improving paths from the source to the washbasin. Each path improves flows and capacities in terms of the minimum residual capacity by pushing flow equal to that value. DFS was used because it is easy to implement, flexible enough to handle any paths, and appropriate for coursework-level problems with small graph sizes.

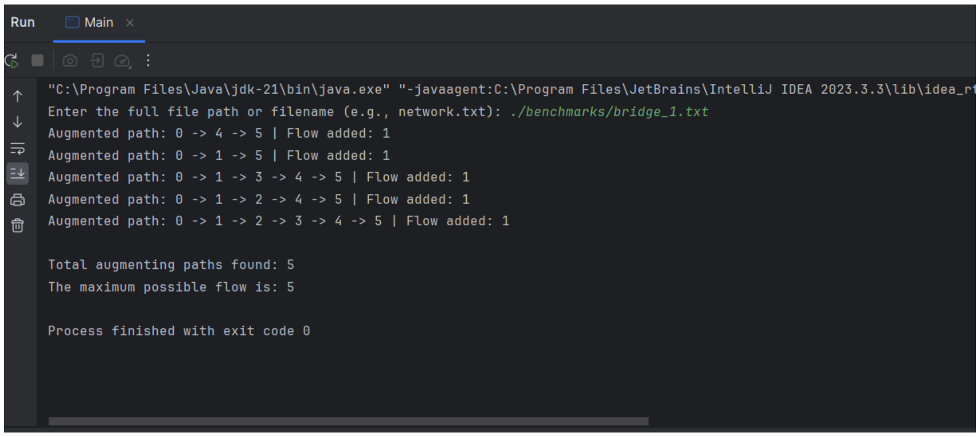
## 2. Example Run and Result

### 2.1 The Smallest Benchmark Example



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

## 2.2 Output of Smallest Benchmark Example



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.3\lib\idea_rt
Enter the full file path or filename (e.g., network.txt): ./benchmarks/bridge_1.txt
Augmented path: 0 -> 4 -> 5 | Flow added: 1
Augmented path: 0 -> 1 -> 5 | Flow added: 1
Augmented path: 0 -> 1 -> 3 -> 4 -> 5 | Flow added: 1
Augmented path: 0 -> 1 -> 2 -> 4 -> 5 | Flow added: 1
Augmented path: 0 -> 1 -> 2 -> 3 -> 4 -> 5 | Flow added: 1

Total augmenting paths found: 5
The maximum possible flow is: 5

Process finished with exit code 0
```

The network here has six nodes, and node 0 is the source node while node 5 is the sink node. Five augmenting paths, each having one unit of flow, were discovered by the program through DFS. Each path pushed very little flow because of its limited edge capabilities. The final result confirmed the algorithm and was as expected network behavior, displaying a maximum flow of 5 units and 5 total augmenting paths.

## 3. Performance Analysis

### 3.1 Theoretical Analysis

The Ford-Fulkerson method has a time complexity of O(max_flow × E), where max_flow is the value of the maximum flow and E is the number of edges. In the worst case, each unit of flow is augmented separately by a DFS that may touch all edges, making the total complexity dependent on both the maximum flow and the number of edges.

### 3.2 Empirical Observation

Compared to using BFS (like Edmonds-Karp), using DFS resulted in a somewhat larger number of iterations because numerous augmenting paths with small flow capacities, particularly flow = 1, exist. However, performance was satisfactory for networks with integral capacity and a small to moderate size.

### 3.3 Order-of-Growth

- The final Big-O classification of the algorithm is: **O(max_flow×E)**
- where the number of iterations depends heavily on the structure of the network and the initial capacity values.