UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

In collaboration with

University of Westminster

BEng (Hons) in Software Engineering

5SENG003C.2

Algorithms: Theory, Design and Implementation

Algorithmic Analysis of Network Flow

Name: Olitha Samaraweera

Email: olitha.20230111@iit.ac.lk

Student Id: 20230111

UOW: w2052883

SE – G12

01. Data Structure and Algorithm Choice

The **Ford-Fulkerson algorithm** was selected to solve the maximum flow problem.

1.1. Graph Representation:

- An adjacency list was used to represent the graph in which each node has a list of outgoing edges. Each edge retains its destination node, capacity, current flow, and reference to its reverse edge to allow updates for the residual graph.

1.2. Reason for Choice:

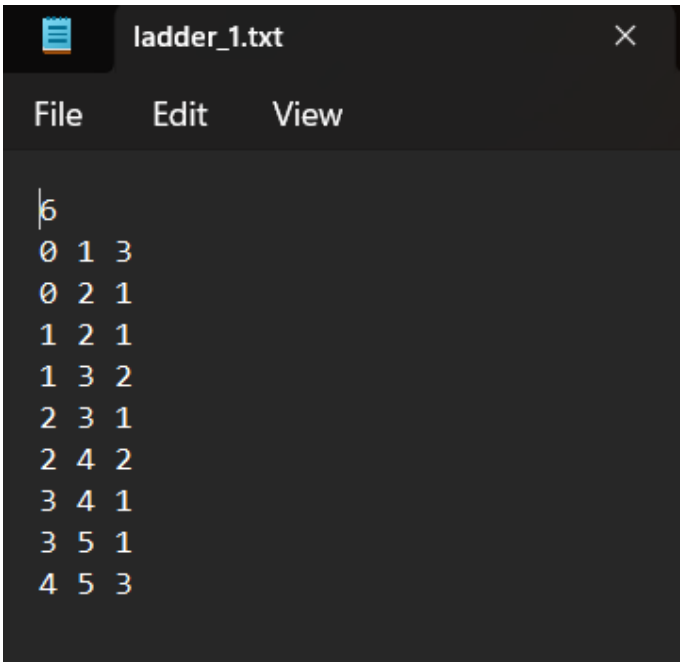
- Efficient memory usage, especially for sparse graphs.
- Quick neighbor lookup during depth-first searches (DFS).
- Adjacency lists make it easy to update flows and manage residual capacities.

1.3. Algorithm Overview:

- The Ford-Fulkerson method repeatedly finds **augmenting paths** from the source to the sink using **DFS**, augments the flow along the paths by the minimum residual capacity (bottleneck), and continues until no further augmenting paths exist.

02. Run on ladder_1.txt

Input Graph:



Execution Output:

```
**********************************
      Network Flow Algorithm
**********************************

Enter the benchmark file name (eg-: ladder_3): ladder_1

Source Node: 0, Sink Node: 5

Final Augmenting Paths and Flow Calculations:
Path 1: 0 -> 2 -> 4 -> 5 with flow = 1
Path 2: 0 -> 1 -> 3 -> 5 with flow = 1
Path 3: 0 -> 1 -> 3 -> 4 -> 5 with flow = 1
Path 4: 0 -> 1 -> 2 -> 4 -> 5 with flow = 1

Maximum Flow: 4
```

The output shows that the network from the file ladder_1.txt has a maximum flow of 4. The algorithm finds four augmenting paths from the source (node 0) to the sink (node 5), each carrying 1 unit of flow. After applying all paths, the total maximum flow reaches 4, matching the expected result and confirming that the implementation works correctly.

03. Performance Analysis

The implemented algorithm is Ford-Fulkerson using DFS for finding augmenting paths.

3.1 Time Complexity:

- O(E×F), where E is the number of edges and F is the maximum flow. Each DFS traversal takes O(E) time, and up to F augmentations may be needed.

3.2 Space Complexity:

- O(V+E), using an adjacency list for the graph and arrays for tracking visited nodes.

3.3 Summary:

- This approach is efficient for small networks with moderate maximum flow. However, for graphs with large capacities, the number of augmentations increases, leading to slower performance.

3.4 Order of Growth:

- The time complexity is O(E×F) and the space complexity is O(V+E).