UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

INFORMATICS INSTITUTE OF TECHNOLOGY

Algorithms: Theory, Design and Implementation

5SENG003C.2

Coursework Report
Task 5

Name : Kamkanamge Dona Dulani Linara

UoW ID : w2051819

IIT ID : 20231068

a) A short explanation of your choice of data structure and algorithm.

Data Structures:

To represent the network flow graph, we implemented an **adjacency list** using Java's ArrayList<Edge>[]:

- Each node holds a list of Edge objects, where each Edge contains:
    - to (destination node)
    - rev (index of the reverse edge in the destination node's list)
    - capacity and current flow
- For every edge, we create both:
    - A **forward edge** with initial capacity
    - A **reverse (residual) edge** with 0 capacity

This design is memory-efficient and supports quick lookups and updates for flow and residual capacities, aligning well with standard flow network representations.

Algorithm:

We used the **Edmonds-Karp algorithm**, a BFS-based implementation of the **Ford-Fulkerson method**. The key features of this algorithm include:

- Repeated BFS to find the shortest augmenting path (in terms of number of edges)
- Updating flow along each path by the bottleneck (minimum residual capacity)
- Updating both forward and reverse flows for each edge

This method guarantees polynomial-time performance and is easy to debug and verify.

b) A run of your algorithm on the smallest benchmark example.

Input File bridge_1.txt :

```
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
```

Sourcr: Node 0
Sink: Node 5

Execution Output:

```
Augmented path with bottleneck: 1, total flow: 1
Augmented path with bottleneck: 1, total flow: 2
Augmented path with bottleneck: 1, total flow: 3
Augmented path with bottleneck: 1, total flow: 4
Augmented path with bottleneck: 1, total flow: 5
Maximum Flow: 5

Process finished with exit code 0
```

Explanation of Paths Discovered:

The Edmonds-Karp algorithm identified five augmenting paths from the source node (0) to the sink node (5), each supplying one unit of flow as a result of separate network bottlenecks:

1. **Path 1:** 0 → 1 → 5

The flow was directed via this direct path from node 0 to node 1 and finally to the sink. The capacity of edge 1 → 5 acted as the bottleneck, limiting the flow to 1 unit.

2. **Path 2:** 0 → 1 → 2 → 4 → 5

This route passed through nodes two and four before arriving at the washbasin. The flow was limited to 1 unit by the edge 2 → 4, even if the majority of the edges along this path had adequate capacity.

3. **Path 3:** 0 → 1 → 2 → 3 → 4 → 5

Next, the algorithm found a longer path that went through nodes 2, 3, and 4. The limiting factor was the edge 2 → 3, which had a capacity of 1 and caused a bottleneck of 1 unit.

4. **Path 4:** 0 → 4 → 5

Through node 4, this way offered a more straightforward path. However, the contribution of this path was limited to a single unit of flow because the edge 0 → 4 only had a capacity of 1.

5. **Path 5:** 0 → 1 → 3 → 4 → 5

The method used node 3 as a substitute to get to node 4 in the last enhancing phase. Once more permitting a flow of one unit, the bottleneck on this path was the edge 1 → 3.

After each of these five augmenting paths was identified in a different iteration, no more paths with residual capacity were located. This demonstrates that the maximum flow from node 0 to node 5 is 5 units, which is consistent with the theoretical limit determined by the network's structure and capacity limitations.

c) A performance analysis of your algorithmic design and implementation.

The **Edmonds-Karp algorithm**, a breadth-first search (BFS) based enhancement of the traditional **Ford-Fulkerson method** for calculating maximum flow in a directed graph, is the approach used in this course.

Theoretical Time Complexity

The Edmonds-Karp algorithm guarantees polynomial time complexity.

Its runtime is: $O(V.E^2)$

Where:

- V is the number of vertices (nodes),
- E is the number of edges in the graph.

This complexity arises because:

- Each BFS to find an augmenting path takes O(E) time,
- In the worst case, the number of augmenting paths is O(V.E).

Edmonds-Karp is more dependable than the original Ford-Fulkerson algorithm, which might run endlessly if irrational capacities are utilised. This is especially true for graphs with integer capacities, which is what this curriculum requires.

Empirical Performance and Observations

The algorithm was tested on benchmark graphs of varying sizes and structures, including files like bridge_1.txt and ladder_10.txt. Across all cases:

- The BFS efficiently identified the shortest augmenting paths in terms of edge count.
- The algorithm terminated after a predictable number of iterations, with each augmentation step clearly traceable.
- The runtime increased gradually as graph size grew, aligning well with the expected order of growth.

Real-time logging of every augmentation is another element of the technology that aids in tracking cumulative flow and bottleneck data, which is crucial for debugging and verifying accuracy.

Design Efficiency

The overall design separates concerns clearly through modular classes:

- FlowGraph handles network construction,
- Edge encapsulates capacity and flow management,
- EdmondsKarp contains the core algorithm logic.

This division enhances testing, maintainability, and readability of the code. Adjacency lists guarantee space efficiency, particularly for sparse graphs, which are prevalent in real-world network flow situations.

Conclusion

The constructed algorithm successfully strikes a balance between accuracy, efficiency, and simplicity. When it comes to addressing maximum flow issues in academic and real-world contexts where flow capacities are integers, it excels both conceptually and practically.