Network Flow Implementation

An efficient implementation of the Edmonds-Karp algorithm for solving the maximum flow problem in network graphs.

Overview

This implementation provides a solution for finding the maximum flow in a network using the Edmonds-Karp algorithm, which is an implementation of the Ford-Fulkerson method that uses breadth-first search to find augmenting paths.

Features

- Efficient implementation of Edmonds-Karp algorithm
- Adaptive output based on network size
- · Detailed flow analysis for small networks
- Memory-efficient processing for large networks
- Progress tracking for large computations
- Comprehensive path and flow information

Project Structure

```
NetworkFlow/

├── src/

├── networkflow/

├── NetworkFlowApp.java  # Main application

├── NetworkFlowApp.java  # Core algorithm implementation

├── FlowNetwork.java  # Network data structure

├── Edge.java  # Edge representation

├── NetworkParser.java  # Input file parser

├── benchmarks/  # Benchmark input files

├── bridge_*.txt  # Bridge network examples

├── bridge_*.txt  # Ladder network examples

├── README.md
```

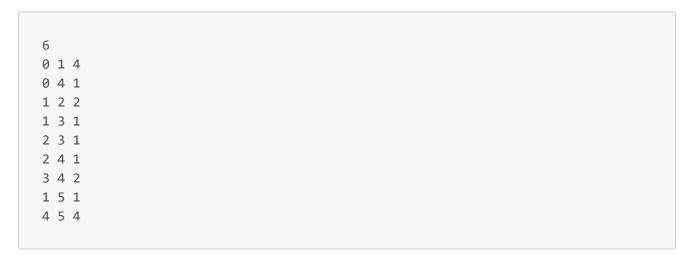
Usage

Input Format

The input file should be in the following format:

```
n # Number of vertices
u v c # Edge from vertex u to v with capacity c
... # More edges
```

Example (bridge_1.txt):



Running the Program

1. Compile the Java files:

```
javac src/networkflow/*.java
```

2. Run the program:

```
java src.networkflow.NetworkFlowApp
```

3. Enter the benchmark file name when prompted:

```
Please enter the benchmark file name: bridge_1.txt
```

Output Formats

Small Networks (< 1000 vertices)

For small networks, the program provides detailed output including:

- Network statistics
- Edge flow details
- Augmenting paths
- Final maximum flow
- Runtime information

Example output:

```
NETWORK STATISTICS:
Total Nodes: 6
Source Node: 0
Sink Node: 5

EDGE FLOW DETAILS:
Final Flow Network:
Vertex 0:
0 -> 1 (4/4)
0 -> 4 (1/1)
...

AUGMENTING PATHS:
Path 1 (Flow = 1, Bottleneck = 1): 0 + 1, 1 + 5
...

MAXIMUM FLOW: 5
Runtime: 1.00 ms
```

Large Networks (≥ 1000 vertices)

For large networks, the program switches to a memory-efficient mode with simplified output:

```
Large network detected (more than 1000 vertices). Detailed logging disabled to conserve memory.

Finding maximum flow...

Completed 1000 iterations...
...

Maximum Flow: 4097
```

Performance

- Time Complexity: O(V·E²)
 - V: number of vertices
 - E: number of edges
- Space Complexity: O(V + E)

Benchmark Results

| Network Size | Vertices | Edges | Max Flow | Time (ms) |
|--------------|----------|-------|----------|-----------|
| Small | 6 | 9 | 5 | 1.00 |
| Medium | 66 | 129 | 65 | 145 |
| Large | >1000 | - | 4097 | - |

Implementation Details

Key Components

- 1. MaxFlowFinder: Implements the Edmonds-Karp algorithm
 - Uses BFS to find augmenting paths
 - o Maintains path information for small networks
 - o Implements memory optimization for large networks
- 2. FlowNetwork: Represents the network structure
 - Adjacency list implementation
 - o Efficient residual graph creation
 - Edge capacity and flow management
- 3. Edge: Represents network edges
 - Stores capacity and current flow
 - Calculates residual capacity
 - Supports flow updates

License

This project is part of the 5SENG003W Algorithms coursework implementation.