Project README for CSE 464: Software QA and Testing

Project Overview

This project is part of CSE 464: Software QA and Testing. The goal is to implement and test a graph management system with functionalities for adding, removing, and searching nodes and edges in a directed graph. The project also includes continuous integration (CI) to automatically build and test code upon each commit to the GitHub repository.

1. GitHub Repository Link

https://github.com/sdingwan/CSE-464-2024-sdingwan

2. Prerequisites

Before running the project, ensure the following software is installed:

- **Java JDK** (version 8 or higher)
- **Maven** (for building and testing the project)

3. Cloning the Repository

To download the project, clone the repository using Git:

git clone <repository-link>
cd <repository-folder>

4. Building the Project

Build the project using Maven:

mvn package

This command will compile the code and package it as a JAR file if successful.

5. Running the Code

Here are example commands to demonstrate the primary functionalities of the GraphParser class.

Example Code for Running GraphParser

Initialize the GraphParser:

```
GraphParser graph = new GraphParser();
```

Adding Nodes and Edges:

```
graph.addNode(new Node("A"));
graph.addNode(new Node("B"));
graph.addEdge(new Node("A"), new Node("B"));
```

Searching for a Path:

```
Path pathBFS = graph.GraphSearch(new Node("A"), new Node("B"),
GraphParser.Algorithm.BFS);
System.out.println("Path (BFS): " + pathBFS);

Path pathDFS = graph.GraphSearch(new Node("A"), new Node("B"),
GraphParser.Algorithm.DFS);
System.out.println("Path (DFS): " + pathDFS);
```

Removing Nodes and Edges:

```
graph.removeNode(new Node("A"));
graph.removeEdge(new Node("A"), new Node("B"));
```

6. APIs Documentation

List of APIs

1. **addNode(Node node)**: Adds a node to the graph.
2. **addEdge(Node src, Node dst)**: Adds a directed edge from src to dst.
3. **removeNode(Node node)**: Removes a specified node and its associated edges.
4. **removeEdge(Node src, Node dst)**: Removes a specified edge from src to dst.
5. **GraphSearch(Node src, Node dst, Algorithm algo)**: Searches for a path from src to dst using either BFS or DFS (selected by algo parameter).

API Usage Examples

```
// Adding nodes
graph.addNode(new Node("X"));

// Adding an edge
graph.addEdge(new Node("X"), new Node("Y"));

// Searching for a path using BFS
Path path = graph.GraphSearch(new Node("X"), new Node("Y"), GraphParser.Algorithm.BFS);

// Removing nodes
graph.removeNode(new Node("X"));

// Removing an edge
graph.removeEdge(new Node("X"), new Node("Y"));
```

7. Running Test Cases

To run the test suite, execute:

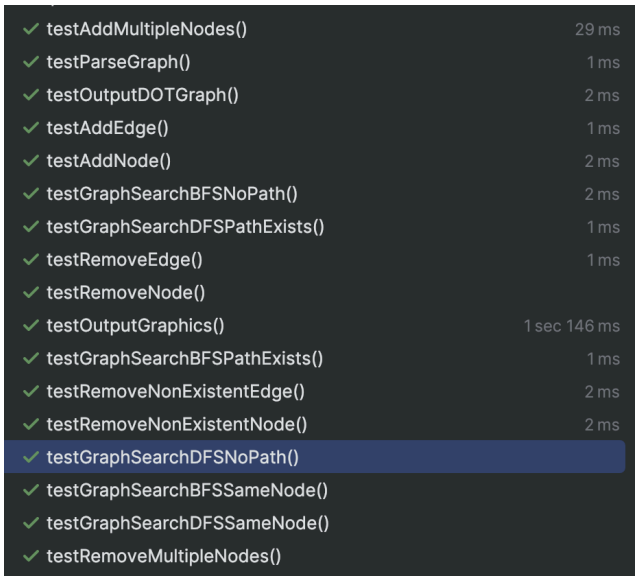
```
mvn test
```

Test Coverage

The tests cover:

- Adding nodes and edges.
- Removing nodes and edges (with expected exception handling).
- Graph search using BFS and DFS.

Example Output from Running Tests




```
] T E S T S
] -------------------------------------------------------
] Running GraphParserTest
: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
: Defaulting to no-operation (NOP) logger implementation
: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
] Tests run: 17, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.984 s - in GraphParserTest
]
] Results:
]
] Tests run: 17, Failures: 0, Errors: 0, Skipped: 0
]
]
] --- jar:3.4.1:jar (default-jar) @ CSE-464-2024-sdingwan ---
] ------------------------------------------------------------------------
] BUILD SUCCESS
] ------------------------------------------------------------------------
] Total time:  2.726 s
```

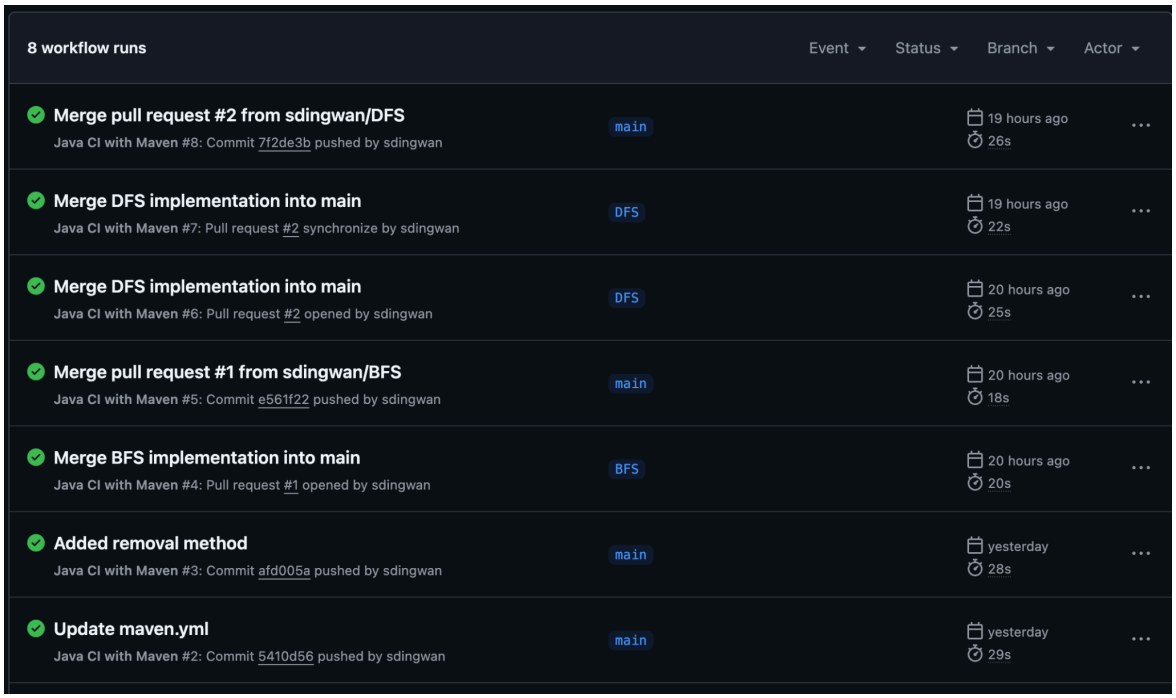
8. Continuous Integration (CI)

This project uses GitHub Actions to automate building and testing the code on each commit.

GitHub Actions Workflow

- **Workflow Link**:
  https://github.com/sdingwan/CSE-464-2024-sdingwan/actions/workflows/maven.yml

Screenshot of Successful CI Execution



9. Branches and Merges

Branches

- **main**: Contains the final unified code with all features and tests.

  https://github.com/sdingwan/CSE-464-2024-sdingwan/tree/main

- **bfs**: Implements the GraphSearch API using BFS.

  https://github.com/sdingwan/CSE-464-2024-sdingwan/tree/BFS

- **dfs**: Implements the GraphSearch API using DFS.

  https://github.com/sdingwan/CSE-464-2024-sdingwan/tree/DFS

Resolving Merge Conflicts

The bfs and dfs branches were merged into main, with conflicts resolved by creating a unified GraphSearch method that accepts an Algorithm parameter (BFS or DFS).

11. GitHub Commits and Links

Key Commits

- **Adding APIs for node and edge removal**: https://github.com/sdingwan/CSE-464-2024-sdingwan/commit/afd005ad0284974f9ca5991ffb3bc182b462b307
- **Implementing GraphSearch with BFS**: https://github.com/sdingwan/CSE-464-2024-sdingwan/commit/c9c1a60274a554053f68799d5732ac51bcd3fa1a
- **Implementing GraphSearch with DFS**: https://github.com/sdingwan/CSE-464-2024-sdingwan/commit/43df39f9d7cda3b78259fa1caac7b5a4308f89ec
- **Merging and Resolving Conflicts with Unified GraphSearch**: https://github.com/sdingwan/CSE-464-2024-sdingwan/commit/2e1062e65a3c67835277a7eede8a9de811d427fa

  https://github.com/sdingwan/CSE-464-2024-sdingwan/commit/e561f22108e571466d2d86e78526fe9eafa8e756

  https://github.com/sdingwan/CSE-464-2024-sdingwan/commit/7f2de3b12c86c79d129dd87cd04a4cb5c32143bb