Project 01 Part 1 README

Description

This project implements a Java application for parsing, manipulating, and visualizing DOT format graphs.

It also includes:
- Node and edge removal capabilities
- Path finding using BFS and DFS algorithms
- Continuous integration with GitHub Actions

Prerequisites:

Java17+
Maven 3.6+
IntelliJ IDEA/CE

Building Project:

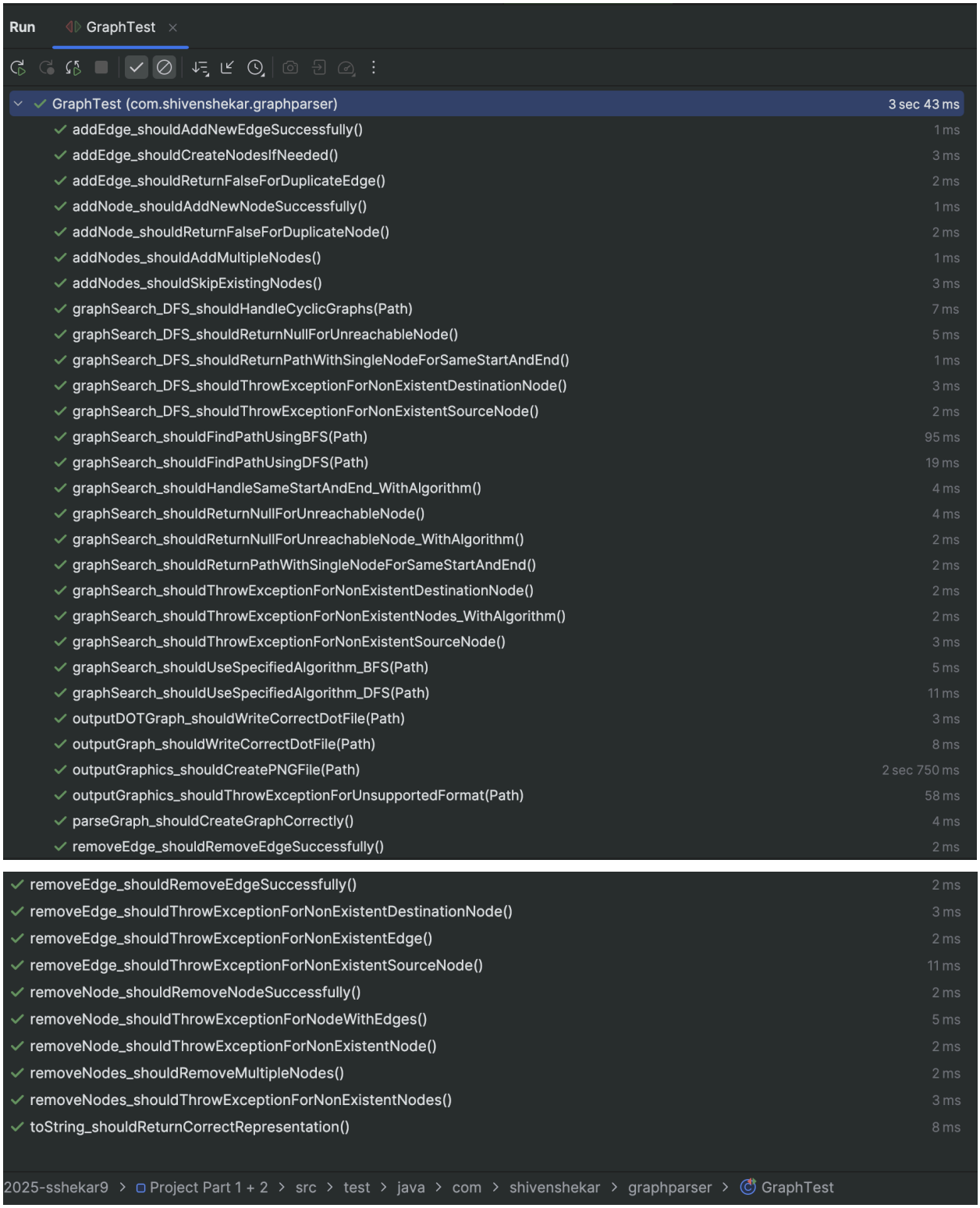
Run: mvn package

Complete Unit Testing

Run: mvn test

OR

1. Open in IntelliJ
2. Go to GraphTest.java
3. Click on run in class GraphTest {
4. Expected Output:


Run
GraphTest
GraphTest (com.shivenshekar.graphparser) 3 sec 43 ms
addEdge_shouldAddNewEdgeSuccessfully() 1 ms
addEdge_shouldCreateNodesIfNeeded() 3 ms
addEdge_shouldReturnFalseForDuplicateEdge() 2 ms
addNode_shouldAddNewNodeSuccessfully() 1 ms
addNode_shouldReturnFalseForDuplicateNode() 2 ms
addNodes_shouldAddMultipleNodes() 1 ms
addNodes_shouldSkipExistingNodes() 3 ms
graphSearch_DFS_shouldHandleCyclicGraphs(Path) 7 ms
graphSearch_DFS_shouldReturnNullForUnreachableNode() 5 ms
graphSearch_DFS_shouldReturnPathWithSingleNodeForSameStartAndEnd() 1 ms
graphSearch_DFS_shouldThrowExceptionForNonExistentDestinationNode() 3 ms
graphSearch_DFS_shouldThrowExceptionForNonExistentSourceNode() 2 ms
graphSearch_shouldFindPathUsingBFS(Path) 95 ms
graphSearch_shouldFindPathUsingDFS(Path) 19 ms
graphSearch_shouldHandleSameStartAndEnd_WithAlgorithm() 4 ms
graphSearch_shouldReturnNullForUnreachableNode() 4 ms
graphSearch_shouldReturnNullForUnreachableNode_WithAlgorithm() 2 ms
graphSearch_shouldReturnPathWithSingleNodeForSameStartAndEnd() 2 ms
graphSearch_shouldThrowExceptionForNonExistentDestinationNode() 2 ms
graphSearch_shouldThrowExceptionForNonExistentNodes_WithAlgorithm() 2 ms
graphSearch_shouldThrowExceptionForNonExistentSourceNode() 3 ms
graphSearch_shouldUseSpecifiedAlgorithm_BFS(Path) 5 ms
graphSearch_shouldUseSpecifiedAlgorithm_DFS(Path) 11 ms
outputDOTGraph_shouldWriteCorrectDotFile(Path) 3 ms
outputGraph_shouldWriteCorrectDotFile(Path) 8 ms
outputGraphics_shouldCreatePNGFile(Path) 2 sec 750 ms
outputGraphics_shouldThrowExceptionForUnsupportedFormat(Path) 58 ms
parseGraph_shouldCreateGraphCorrectly() 4 ms
removeEdge_shouldRemoveEdgeSuccessfully() 2 ms

removeEdge_shouldRemoveEdgeSuccessfully() 2 ms
removeEdge_shouldThrowExceptionForNonExistentDestinationNode() 3 ms
removeEdge_shouldThrowExceptionForNonExistentEdge() 2 ms
removeEdge_shouldThrowExceptionForNonExistentSourceNode() 11 ms
removeNode_shouldRemoveNodeSuccessfully() 2 ms
removeNode_shouldThrowExceptionForNodeWithEdges() 5 ms
removeNode_shouldThrowExceptionForNonExistentNode() 2 ms
removeNodes_shouldRemoveMultipleNodes() 2 ms
removeNodes_shouldThrowExceptionForNonExistentNodes() 3 ms
toString_shouldReturnCorrectRepresentation() 8 ms
2025-sshekar9 › Project Part 1 + 2 › src › test › java › com › shivenshekar › graphparser › GraphTest

Features:

1. Parsing DOT File

Commit Link:
https://github.com/shiven01/CSE-464-2025-sshekar9/commit/901b5989750f3d0262a40c079a2058e872991765

Expected Output:
After running mvn test, look for the passing test cases:

- parseGraph_shouldCreateGraphCorrectly() - Tests the core parsing functionality
- toString_shouldReturnCorrectRepresentation() - Tests the string representation
- outputGraph_shouldWriteCorrectDotFile(Path) - Tests writing the graph to a DOT file

2. Adding Nodes to Graph

Commit Link:
https://github.com/shiven01/CSE-464-2025-sshekar9/commit/10198d6e2b3b40aa8eb2b7fb02c0d104c73df152

Expected Output:
After running mvn test, look for:

- addNode_shouldAddNewNodeSuccessfully() - Tests adding a single new node
- addNode_shouldReturnFalseForDuplicateNode() - Tests duplicate node handling
- addNodes_shouldAddMultipleNodes() - Tests adding multiple nodes at once
- addNodes_shouldSkipExistingNodes() - Tests adding a mix of new and existing nodes

3. Adding Edges to Graph

Commit Link:
https://github.com/shiven01/CSE-464-2025-sshekar9/commit/10c0f66cabbdc5112bc3387b6e0045b774ab9866

Expected Output:
After running mvn test, look for:

- addEdge_shouldAddNewEdgeSuccessfully() - Tests adding a new edge between nodes
- addEdge_shouldReturnFalseForDuplicateEdge() - Tests duplicate edge handling

- addEdge_shouldCreateNodesIfNeeded() - Tests creating missing nodes automatically when adding an edge

4. Output to Graph

Commit Link:
https://github.com/shiven01/CSE-464-2025-sshekar9/commit/1f86be4e4bd971880e5b9ad6a3ad34d485aca051

Expected Output:
After running mvn test, look for:

- outputDOTGraph_shouldWriteCorrectDotFile(Path) - Tests DOT format output
- outputGraphics_shouldCreatePNGFile(Path) - Tests PNG image generation
- outputGraphics_shouldThrowExceptionForUnsupportedFormat(Path) - Tests error handling for unsupported formats

5. Path Finding with BFS/DFS

Commit Link:
https://github.com/shiven01/CSE-464-2025-sshekar9/commit/25e5beb907e3424ed65448e54d01dcdf37493fd8

Expected Output:
After running mvn test, look for:

- graphSearch_shouldFindPathUsingBFS(Path) - Tests BFS path finding between nodes
- graphSearch_shouldReturnPathWithSingleNodeForSameStartAndEnd() - Tests BFS when source and destination are the same
- graphSearch_shouldReturnNullForUnreachableNode() - Tests BFS when destination is unreachable
- graphSearch_shouldThrowExceptionForNonExistentSourceNode() - Tests BFS error handling for non-existent source
- graphSearch_shouldThrowExceptionForNonExistentDestinationNode() - Tests BFS error handling for non-existent destination
- graphSearch_shouldFindPathUsingDFS(Path) - Tests DFS path finding between nodes
- graphSearch_DFS_shouldReturnPathWithSingleNodeForSameStartAndEnd() - Tests DFS when source and destination are the same
- graphSearch_DFS_shouldReturnNullForUnreachableNode() - Tests DFS when destination is unreachable
- graphSearch_DFS_shouldThrowExceptionForNonExistentSourceNode() - Tests DFS error handling for non-existent source

- graphSearch_DFS_shouldThrowExceptionForNonExistentDestinationNode() - Tests DFS error handling for non-existent destination
- graphSearch_DFS_shouldHandleCyclicGraphs(Path) - Tests DFS with graphs containing cycles
- graphSearch_shouldUseSpecifiedAlgorithm_BFS(Path) - Tests path finding using explicit BFS algorithm parameter
- graphSearch_shouldUseSpecifiedAlgorithm_DFS(Path) - Tests path finding using explicit DFS algorithm parameter
- graphSearch_shouldHandleSameStartAndEnd_WithAlgorithm() - Tests both algorithms when source and destination are the same
- graphSearch_shouldReturnNullForUnreachableNode_WithAlgorithm() - Tests both algorithms when destination is unreachable
- graphSearch_shouldThrowExceptionForNonExistentNodes_WithAlgorithm() - Tests both algorithms' error handling for non-existent nodes

Project Structure

```
src/
  main/
    java/
      com/
        shivenshekar/
          graphparser/
            Graph.java
            Main.java
            Algorithm.java
            Path.java
    resources/
      sample.dot         // Sample DOT file for testing
  test/
    java/
      com/
        shivenshekar/
          graphparser/
            GraphTest.java // Unit tests
  .github/
    workflows/
        maven.yml
```