CSE-464-2023-gsong20 README

Introduction

This README provides instructions for using the GraphFeatures Java class to manipulate and analyze directed graphs. It includes information on how to compile and run the code, example input files, and expected output.

Table of Contents

Prerequisites

Before running the code, make sure you have the following prerequisites installed on your system:

- Java Development Kit (JDK)
- Graphviz (for generating graphics)

Running the Code

1. Compile the `GraphFeatures.java` file:

   ```
   javac GraphFeatures.java
   ```

2. Run the code with your input file:

   ```
   java GraphFeatures input.dot
   ```

   Replace `input.dot` with the path to your input file. This will parse the graph, perform operations, and generate output files.

Example Inputs

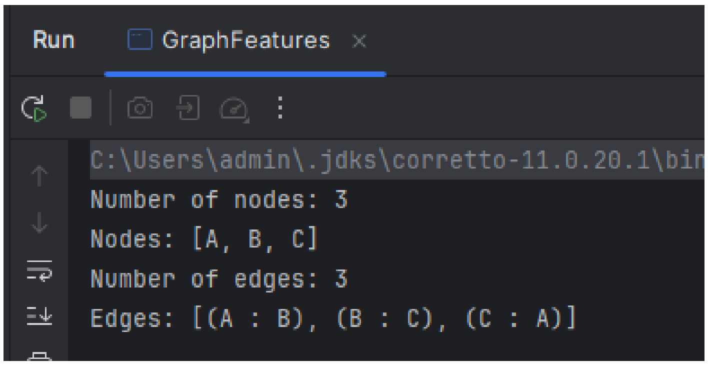
You can use the provided `input.dot` file as an example input for the code. This file defines nodes and edges in the DOT format. You can create your input files in a similar format.

```
digraph MyGraph {
    // Define nodes
    A
    B
    C
    // Define edges
    A -> B
    B -> C
    C -> A
}
```

Expected Outputs

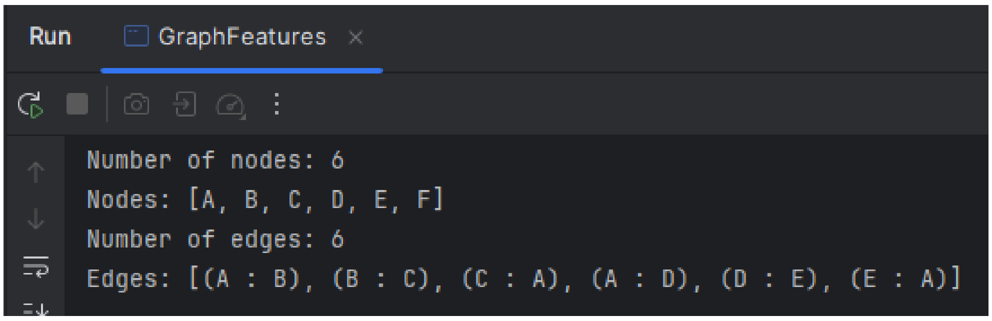
Screenshots of the expected output for each feature:

1. **Parsing the Graph:**



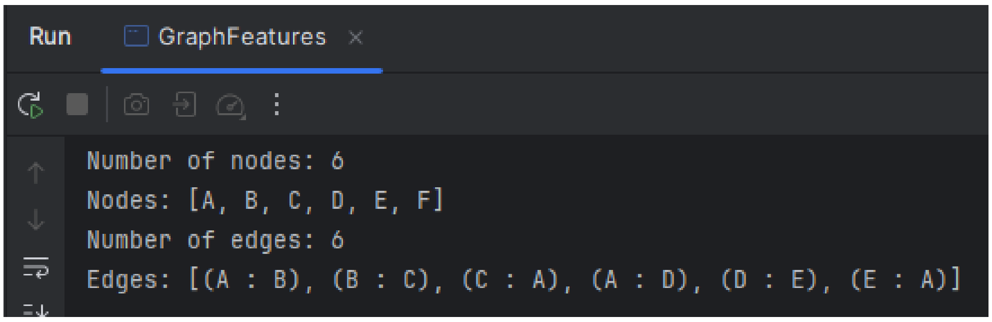
```
C:\Users\admin\.jdks\corretto-11.0.20.1\bin
Number of nodes: 3
Nodes: [A, B, C]
Number of edges: 3
Edges: [(A : B), (B : C), (C : A)]
```

2. **Adding Nodes:**



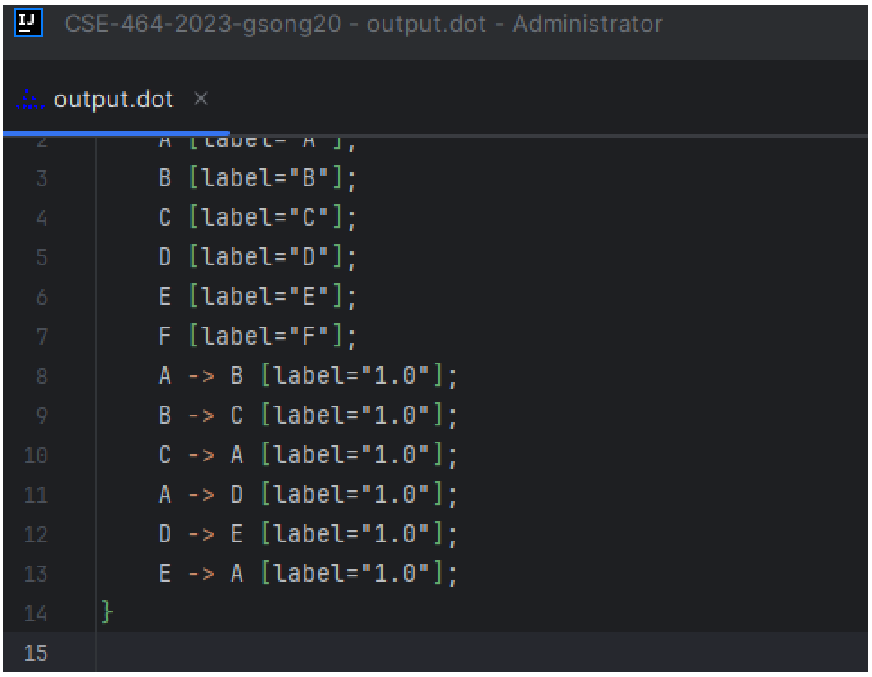
```
Number of nodes: 6
Nodes: [A, B, C, D, E, F]
Number of edges: 6
Edges: [(A : B), (B : C), (C : A), (A : D), (D : E), (E : A)]
```

3. **Adding Edges:**



```
Number of nodes: 6
Nodes: [A, B, C, D, E, F]
Number of edges: 6
Edges: [(A : B), (B : C), (C : A), (A : D), (D : E), (E : A)]
```

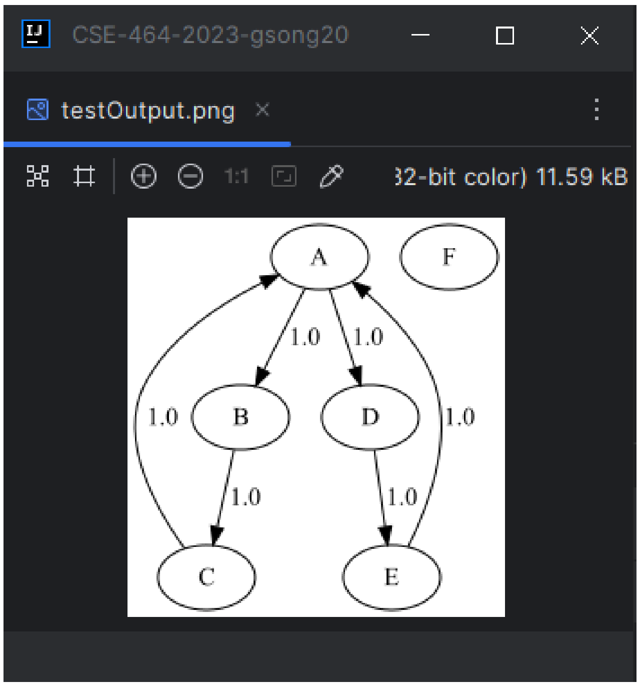
4. **Outputting the Graph to a DOT file:**


CSE-464-2023-gsong20 - output.dot - Administrator

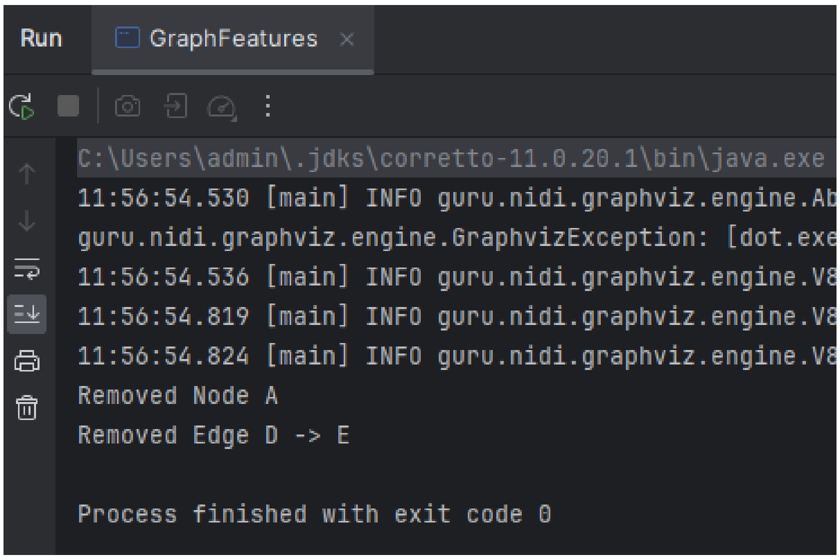
output.dot

```
    A [label="A"];
    B [label="B"];
    C [label="C"];
    D [label="D"];
    E [label="E"];
    F [label="F"];
    A -> B [label="1.0"];
    B -> C [label="1.0"];
    C -> A [label="1.0"];
    A -> D [label="1.0"];
    D -> E [label="1.0"];
    E -> A [label="1.0"];
}
```

5. **Outputting the Graph to a PNG image:**



6. **Deleting Nodes and Edges:**



```
C:\Users\admin\.jdks\corretto-11.0.20.1\bin\java.exe
11:56:54.530 [main] INFO guru.nidi.graphviz.engine.Ab
guru.nidi.graphviz.engine.GraphvizException: [dot.exe
11:56:54.536 [main] INFO guru.nidi.graphviz.engine.V8
11:56:54.819 [main] INFO guru.nidi.graphviz.engine.V8
11:56:54.824 [main] INFO guru.nidi.graphviz.engine.V8
Removed Node A
Removed Edge D -> E

Process finished with exit code 0
```

Running Tests

The `GraphFeaturesTest.java` file contains test cases for various functions in the code. You can run the tests to verify the correctness of the code.

1. Compile the `GraphFeaturesTest.java` file:

   ```
   javac GraphFeaturesTest.java
   ```

2. Run the tests:

   ```
   java -cp .:junit-<version>.jar org.junit.runner.JUnitCore org.example.GraphFeature
   sTest
   ```

   Replace `<version>` with the version of JUnit.

GitHub Links

GitHub links for each feature and test:

1. **Feature 1: Parsing the Graph**

   Commit for Parsing the Graph

2. **Feature 2: Adding Nodes**

   Commit for Adding Nodes

3. **Feature 3: Adding Edges**

   Commit for Adding Edges

4. **Feature 4: Outputting the Graph to a DOT file**

   Commit for Outputting the Graph to a DOT file

5. **Unit Tests**

   Commit for Unit Test Cases

6. **Adding README PDF**

   Commit for generating the README PDF

7. **Adding Continuous Integration and Build Automation**

   Commit for adding continuous integration (CI) and build automation

8. **Feature 5: Removing nodes and edges API**

   Commit for removing nodes and edges API

9. **Implementing Graph Search using Breadth-First Search (BFS)**

   Commit for implementing graph search using Breadth-First Search (BFS)

10. **Implementing Graph Search using Depth-First Search (DFS)**

    Commit for implementing graph search using Depth-First Search (DFS)

11. **Merging Branch 'dfs' and Introducing Enum `Algorithm`**

    Merge branch 'dfs' and introduce enum class `Algorithm`

12. **Updating the Maven Workflow for fixing Build Automation**

    Commit for updating the Maven workflow for improved build automation