

GraphDot Project - CSE 464

Brady Flahart

Repo: <https://github.com/btflahar/CSE-464-btflahar-graph.git>

mvn package command works and will properly build

There should be a total of 7 Tests that run when package is run

I used the Maven Sideboard and did Lifecycle -> Package but mvn package is the same so it should work

From here, you can either click the StartArrow to the left of GraphApp.main() located in CSE464-btflahar-graph/src/main/java/btflahar/asu/edu/graphDot/GraphApp

Or

You can run the command

java -cp target/classes btflahar.asu.edu.graphDot.GraphApp input.dot input.png

It should produce a graph-report.txt file (a text report of what the expected node and edge count is as it is meant to be a parseGraph test.)

As well as an input.dot (DOT file)

And an input.png (the png of the dot file).

It is important to note that the input.Dot file will be overwritten after being parsed to contain the normal form of the dot graph. It will list all unique nodes, then their relationships (edges) after. This was so it would be easier to print and test later on. The example input.dot I gave is very simple

```
digraph G {  
    A -> B;  
    B -> C;  
    D;  
}
```

The expected number of nodes would be 4, and the number of edges would be 2.

All of my Test Functions are located in

CSE464-btflahar-graph/src/test/java/btflahar/asu/edu/graphDot/GraphAppTest.java

Below are the screenshots for the feature 1 function tests showing it works as it should, (first two pictures are parseGraph and the last is outputGraph)

(You may have to zoom in I apologize)

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows the project structure with packages `main`, `btflahar.asu.edu`, and `graphDot`. Inside `graphDot`, there are files `GraphApp.java`, `Main.java`, and `GraphAppTest.java`.
- Code Editor:** The main editor window displays `GraphApp.java` containing Java code for a command-line application. The code reads input from `input.dot` and outputs a graph report to `graph-report.txt` and `input.png`.
- Run Output:** Below the editor, the terminal pane shows the execution of the application. It prints:

```
C:\Users\brady\.jdks\openjdk-28\bin\java.exe ...
Nodes: 4
Edges: 2
A -> B
B -> C
```
- Toolbars and Icons:** Standard IntelliJ IDEA toolbars and icons are visible along the top and sides of the interface.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows the project structure with package `main` containing files `GraphApp.java`, `Main.java`, and `GraphAppTest.java`.
- Maven:** The right sidebar shows the Maven lifecycle with the `package` goal selected.
- Code Editor:** The main editor window displays `graph-report.txt` containing the output of the application:

```
Nodes: 4
Edges: 2
A -> B
B -> C
```
- Run Output:** Below the editor, the terminal pane shows the execution of the application. It prints:

```
C:\Users\brady\.jdks\openjdk-28\bin\java.exe ...
Nodes: 4
Edges: 2
A -> B
B -> C
```
- Toolbars and Icons:** Standard IntelliJ IDEA toolbars and icons are visible along the top and sides of the interface.

The screenshot shows an IDE interface with a code editor, a file tree, and a terminal window.

File Tree:

- .mvn
- src
 - main
 - java
 - btflahar.asu.edu.graphDot
 - GraphApp
 - Main
 - resources
 - test
 - java
 - btflahar.asu.edu.graphDot
- target
- .gitignore
- expected.txt
- graph-report.txt
- input.dot
- input.png
- pom.xml
- README.md
- External Libraries

From here we move to Feature 2, addNode, addNodes Tests. For these I simply gave a test graph, then added duplicates to ensure my program wouldn't add them, but instead ignore. The expected results are just below it in the same class.

The screenshot shows an IDE interface with a code editor, a file tree, and a terminal window.

File Tree:

- test
 - java
 - btflahar.asu.edu.graphDot
 - GraphAppTest
- target
- .gitignore
- expected.txt
- graph-report.txt
- input.dot
- input.png
- pom.xml
- README.md
- External Libraries

The screenshot shows an IDE interface with a project structure on the left and code editor on the right.

Project Structure:

- btflahar.asu.edu.graphDot
- GraphAppTest
- target
- .gitignore
- expected.txt
- graph-report.txt
- input.dot
- input.png
- pom.xml
- README.md
- External Libraries

Code Editor (GraphAppTest.java):

```
57  
58  
59 @Test new *  
void addNodes_duplicatesTestDot() {  
    GraphApp app = new GraphApp();  
    app.addNode(new String[]{"P", "Q", "R", "P", "Q"});  
    String out = app.toString();  
    assertTrue(out.contains("Nodes: 3"), out); // P,Q,R  
    assertTrue(out.contains("Edges: 0"), out);  
}  
// ----- Feature 3: add edges (new case) -----  
@Test new *  
void addEdge_duplicateIgnoreTest1() {
```

Run Tab:

GraphAppTest.addNodes_duplicatesTestDot 50ms

1 test passed 1 test total, 50ms

C:\Users\brady\.jdks\openjdk-25\bin\java.exe ...

Process finished with exit code 0

Next we have Feature 3, addEdge where I followed a similar pattern. I tried to add duplicate edges and made an example dot to test the function. I placed the simple expected output below, where is counts the number of edges (shouldnt add the duplicate edge so 2)

The screenshot shows an IDE interface with a project structure on the left and code editor on the right.

Project Structure:

- java
- btflahar.asu.edu
- graphDot
- GraphApp
- Main
- resources
- test
- java
- btflahar.asu.edu.graphDot
- GraphAppTest
- target
- .gitignore
- expected.txt
- graph-report.txt
- input.dot
- input.png
- pom.xml
- README.md
- External Libraries

Code Editor (GraphAppTest.java):

```
67  
68  
69 @Test new *  
void addEdge_duplicateIgnoreTest1() {  
    GraphApp app = new GraphApp();  
    app.addNode(new String[]{"L", "M", "N"});  
  
    app.addEdge(srcLabel: "L", dstLabel: "M");  
    app.addEdge(srcLabel: "L", dstLabel: "M"); // test case duplicate  
    app.addEdge(srcLabel: "L", dstLabel: "N"); // different after duplicate test case  
  
    String out = app.toString();  
  
    assertTrue(out.contains("Edges: 2"), out);  
    assertTrue(out.contains("L -> M"), out);  
    assertTrue(out.contains("L -> N"), out);  
}  
// ----- Feature 4: output DOT (new case) -----  
@Test new *  
void outputDOTGraphCreatesProperTextTest(@TempDir Path tmp) throws IOException {  
    GraphApp app = new GraphApp();
```

Run Tab:

GraphAppTest.addEdge_duplicateIgnoreTest1 50ms

1 test passed 1 test total, 50ms

C:\Users\brady\.jdks\openjdk-25\bin\java.exe ...

Process finished with exit code 0

Finally we move onto Feature 4 which is more advanced as it needs to output a png. The sample file I gave (input.dot) can be modified to make a different png but I simply used a 4 node, 2 edge dot graph for my testing. The first pictures show the function properly, the last shows the proper PNG output.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left pane shows a Java project structure with packages like `main.java` and `test.java` containing classes `GraphApp` and `Main`, and resources like `input.dot` and `input.png`.
- Code Editor:** The right pane displays a Java test class `GraphAppTest` with a single test method `outputDOTGraphCreatesProperTextTest`. The code sets up a `GraphApp` instance with nodes A1, B2, C3, D4 and edges A1-B2, B2-C3, then outputs the graph to a temporary file and reads its content to verify it matches the expected DOT format.
- Run Tab:** Below the editor, the "Run" tab shows the test has passed in 79ms.
- Output Tab:** The bottom pane shows the command-line output of the test execution.

```

src
└── main
    └── java
        └── btflahar.asu.edu
            └── graphDot
                ├── GraphApp
                └── Main
            └── resources
        └── test
            └── java
                └── btflahar.asu.edu.graphDot
                    └── GraphAppTest
    > target
        └── .gitignore
    └── expected.txt
    └── graph-report.txt
    └── input.dot
    └── input.png
    └── pom.xml
    └── README.md
> External Libraries
Run GraphAppTest.outputDOTGraphCreatesProperTextTest ×
Run GraphAppTest.outputDOTGraphCreatesProperTextTest 79 ms
    ✓ 1 test passed 1 test total, 79 ms
    C:\Users\brady\.jdks\openjdk-25\bin\java.exe ...
    Process finished with exit code 0
}

```

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left pane shows a Java project structure with packages like `main.java` and `test.java` containing classes `GraphApp` and `Main`, and resources like `input.dot` and `input.png`.
- Code Editor:** The right pane displays a Java test class `GraphAppTest` with a single test method `outputGraph_textTest`. The code sets up a `GraphApp` instance with nodes X, Y, Z and edges X-Y, Y-Z, then outputs the graph to a temporary file and reads its content to verify it matches the expected text representation.
- Run Tab:** Below the editor, the "Run" tab shows the test has passed in 81ms.
- Output Tab:** The bottom pane shows the command-line output of the test execution.

```

.mvn
└── src
    └── main
        └── java
            └── btflahar.asu.edu
                └── graphDot
                    ├── GraphApp
                    └── Main
                └── resources
            └── test
                └── java
                    └── btflahar.asu.edu.graphDot
                        └── GraphAppTest
    > target
        └── .gitignore
    └── expected.txt
    └── graph-report.txt
    └── input.dot
    └── input.png
    └── pom.xml
    └── README.md
> External Libraries
Run GraphAppTest.outputGraph_textTest ×
Run GraphAppTest.outputGraph_textTest 81 ms
    ✓ 1 test passed 1 test total, 81 ms
    C:\Users\brady\.jdks\openjdk-25\bin\java.exe ...
    Process finished with exit code 0
}

```

File tree:

```

> .idea
  .mvn
  src
    main
      java
        btflahar.asu.edu
          graphDot
            GraphApp
            Main
      resources
    test
      java
        btflahar.asu.edu.graphDot
          GraphAppTest
  target
  .gitignore
  expected.txt
  graph-report.txt
  input.dot
  input.png
  pom.xml
  README.md

```

Run outputGraphics_pngTest

GraphAppTest (btflahar.asu.edu.graphDot) 407 ms ✓ 1 test passed 1 test total, 407 ms

```

outputGraphics_pngTest(Path) 407 ms
  C:\Users\bbrady\jdk\openjdk-25\bin\java.exe
  [main] INFO guru.nidi.graphviz.engine.GraphvizCommandLineEngine - input file:///C:/Users/bbrady/AppData/Local/Temp/GraphvizJava/DotEngine259836281714099742/dotfile.dot
  [main] INFO guru.nidi.graphviz.service.CommandlineExecutor - executing command [cmd, /C, dot.exe -Kdot -Tsvg C:/Users/bbrady/AppData/Local/Temp/GraphvizJava/DotEngine259836281714099742/outfile.svg]
  [main] INFO guru.nidi.graphviz.engine.GraphvizCommandLineEngine - output file:///C:/Users/bbrady/AppData/Local/Temp/GraphvizJava/DotEngine259836281714099742/outfile.svg

```

Process finished with exit code 0

```

graph TD
  A((A)) --> B((B))
  B --> C((C))
  B --> D((D))

```

I simply did A → B and B → C then added an additional node that was not an edge.

For the last images, it is showing all test cases in the test path passing.

The screenshot shows the IntelliJ IDEA interface with the Maven tool window open. The 'test' goal is selected in the Maven tree. The terminal window at the bottom displays the build log:

```
[INFO] Results:
[INFO]
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.839 s
[INFO] Finished at: 2025-10-26T22:35:45-07:00
[INFO] -----
```

And lastly, showing that mvn package builds successfully

The screenshot shows the IntelliJ IDEA interface with the Maven tool window open. The 'package' goal is selected in the Maven tree. The terminal window at the bottom displays the build log:

```
[INFO]
[INFO]
[INFO] --- jar:3.4.1:jar (default-jar) @ CSE464-btflahar-graph ---
[INFO] Building jar: C:\Users\brady\IdeaProjects\CSE464-btflahar-graph\target\CSE464-btflahar-graph-1.0-SNAPSHOT.jar
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.298 s
[INFO] Finished at: 2025-10-26T22:47:09-07:00
[INFO] -----
```

So to recap -

Feature 1: Tested using expected.txt and graph-report.txt, can simply click start on main function or type command I gave

Feature 2: Tests are built into test functions, they can be easily modified with desired values but are relatively simple currently. The function not only performs the action, but also compares to what the correct output should be and fails if it does not obtain the correct answer. Can run by simply going to the test path and clicking on any of the start buttons next to any function to run individually.

Feature 3: Very similar to Feature 2: Tests are simple and easy to modify and work with, already have expected outputs and test them when running them.

Feature 4: Outputs graph when program is run. The file it outputs will be input.png, the input file is input.dot. I have the expectation for both in the screenshots.

Commit Links:

Began Working on Project, mapping first few features out Commit 1 and editing pom.xml:

<https://github.com/btflahar/CSE-464-btflahar-graph/commit/647925f2a562917d90befedd2e29aa6620b061de>

Tried to get Feature 4 started along with a test case (finally got pom.xml working too):

<https://github.com/btflahar/CSE-464-btflahar-graph/commit/8b2f644c314c5f16dbab74d321b05a3d26a7d06d>

Feature 1: Full implementation and functionality

<https://github.com/btflahar/CSE-464-btflahar-graph/commit/a52127e3e585b8a984a80b93e88b9a8ec011b4ec>

Feature 2: Full implementation and functionality (also did .gitignore + started test case)

<https://github.com/btflahar/CSE-464-btflahar-graph/commit/a40db89e18bfdd925dbf1c5ff23af39fdc7336dd>

Feature 3: Full implementation and functionality (minor test case implementation)

<https://github.com/btflahar/CSE-464-btflahar-graph/commit/8cd2851e7b8bc27fe5e64fbfc7029e5a799cf89>

Feature 4: Full implementation and functionality

<https://github.com/btflahar/CSE-464-btflahar-graph/commit/502547b40cb91a2d4ff266ed7a7083e655f16c45>

Final Commit - All Tests finished, everything functioning

<https://github.com/btflahar/CSE-464-btflahar-graph/commit/fd91722e1e824cc4581802a80c776b92679eb7fa>