

Salvador Gomez

1218885590

Test cases are given in test folder located in the project folder ProjectTest1 should include all the unit testing.

Github:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47>

Tree Branch BFS:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/bfs>

Tree Branch DFS:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/dfs>

Continuous Integration:

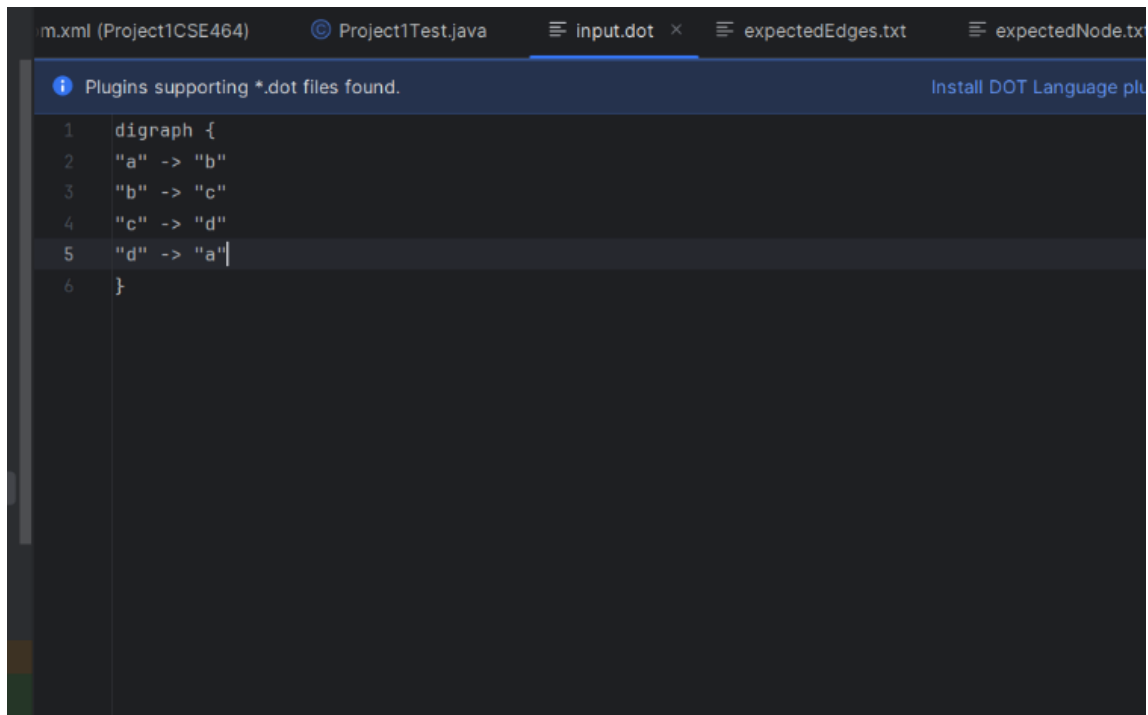
<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/actions/runs/6765986554>

Merge bfs:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/53df010698f5c4423fff91ff37d12cd14b05034f>

Expected outputs:

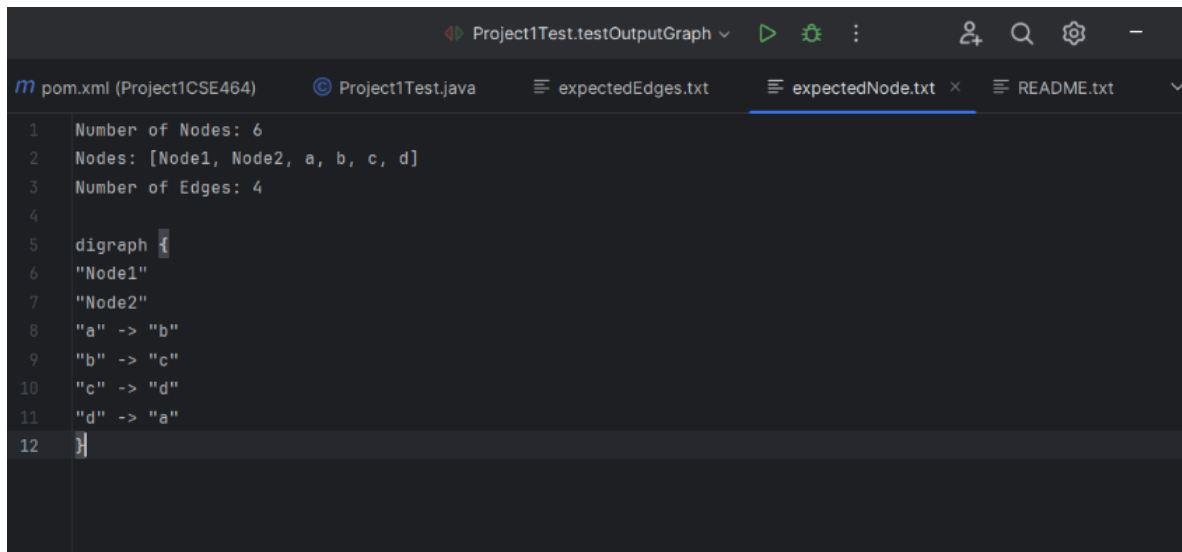
Expected for feature 4 testOutputDotGraph



The screenshot shows an IDE with several tabs: pom.xml (Project1CSE464), Project1Test.java, input.dot, expectedEdges.txt, and expectedNode.txt. The 'input.dot' tab is active, displaying a DOT language definition for a directed graph with four nodes (a, b, c, d) and four edges forming a cycle. A notification bar at the top states 'Plugins supporting *.dot files found.' with a link to 'Install DOT Language plugin'.

```
1 digraph {
2   "a" -> "b"
3   "b" -> "c"
4   "c" -> "d"
5   "d" -> "a"
6 }
```

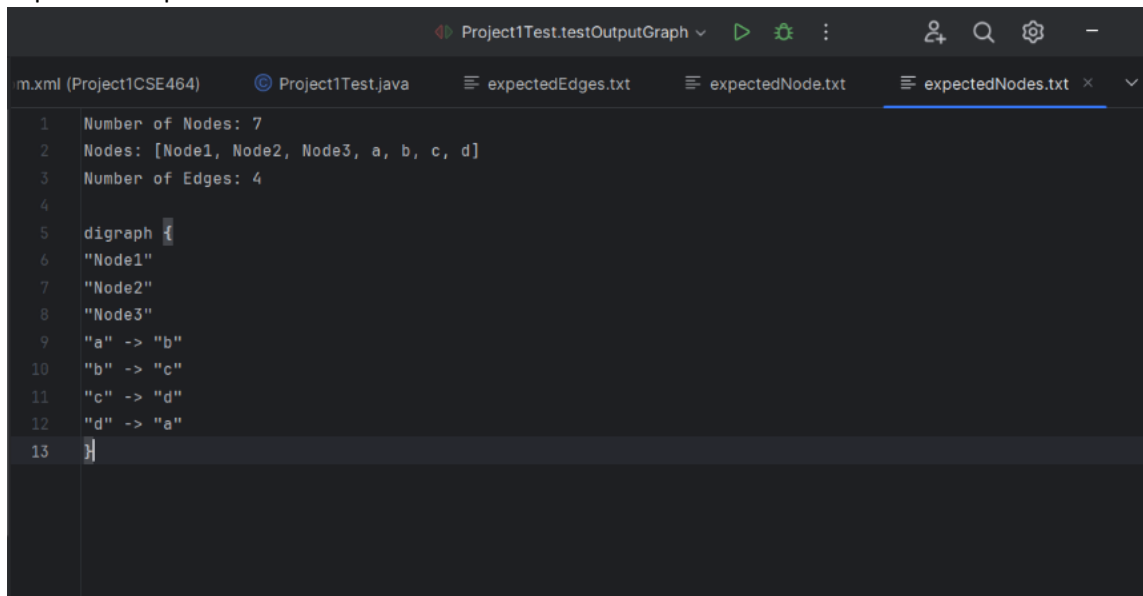
Expected output or feature 2 addNode



The screenshot shows an IDE with tabs: pom.xml (Project1CSE464), Project1Test.java, expectedEdges.txt, expectedNode.txt, and README.txt. The 'expectedNode.txt' tab is active, displaying the expected output of the graph processing. The output includes the number of nodes (6), the list of nodes (Node1, Node2, a, b, c, d), the number of edges (4), and the DOT language definition for the graph.

```
1 Number of Nodes: 6
2 Nodes: [Node1, Node2, a, b, c, d]
3 Number of Edges: 4
4
5 digraph {
6   "Node1"
7   "Node2"
8   "a" -> "b"
9   "b" -> "c"
10  "c" -> "d"
11  "d" -> "a"
12 }
```

Expected output for feature 2 addNodes

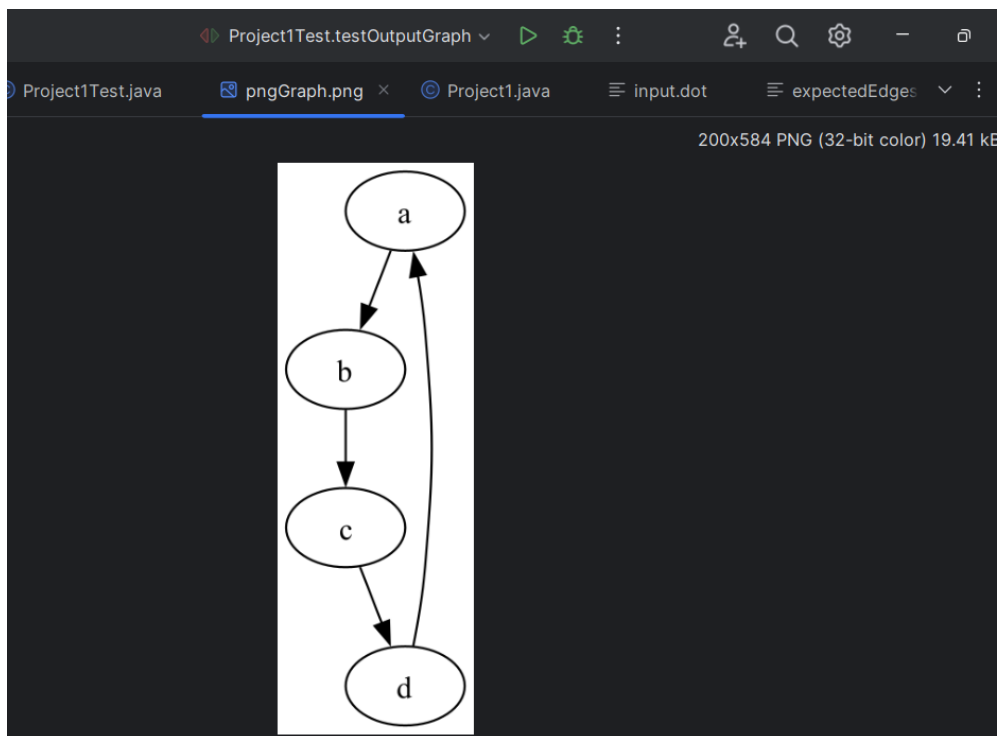
A screenshot of an IDE window showing a Java file named Project1Test.java. The file contains a series of print statements and a digraph object. The output of the program is displayed in the console. The output shows the number of nodes (7), the list of nodes (Node1, Node2, Node3, a, b, c, d), the number of edges (4), and a digraph object with nodes Node1, Node2, Node3, a, b, c, d and edges a -> b, b -> c, c -> d, d -> a.

```
1 Number of Nodes: 7
2 Nodes: [Node1, Node2, Node3, a, b, c, d]
3 Number of Edges: 4
4
5 digraph {
6   "Node1"
7   "Node2"
8   "Node3"
9   "a" -> "b"
10  "b" -> "c"
11  "c" -> "d"
12  "d" -> "a"
13 }
```

Expected output for feature 3 addEdge

```
Project1Test.testOutputGraph
pom.xml (Project1CSE464) Project1Test.java expectedEdges.txt README.txt Project1.java
1 Number of Nodes: 6
2 Nodes: [Node1, Node2, a, b, c, d]
3 Number of Edges: 5
4
5 digraph {
6   "a" -> "b"
7   "b" -> "c"
8   "c" -> "d"
9   "d" -> "a"
10  "Node1" -> "Node2"
11 }
```

Expected output for feature 4 outputGraphics



Expected Output for removeNode

```
Number of Nodes: 6
Nodes: [Node1, Node3, a, b, c, d]
Number of Edges: 4

digraph {
  "Node3"
  "Node1"
  "c" -> "d"
  "d" -> "a"
  "a" -> "b"
  "b" -> "c"
}
```

Expected Output for removeNodes

```
Number of Nodes: 5
Nodes: [Node3, a, b, c, d]
Number of Edges: 4

digraph {
  "Node3"
  "c" -> "d"
  "d" -> "a"
  "a" -> "b"
  "b" -> "c"
}
```

Expected output for removeEdge

```
Number of Nodes: 6
Nodes: [Node1, Node2, a, b, c, d]
Number of Edges: 5

digraph {
  "a" -> "b"
  "b" -> "c"
  "c" -> "d"
  "d" -> "a"
  "Node2" -> "Node1"
}
```

The error files like `removeNodeError`, `removeNodesError`, and `removeEdgeError` should all return the error message into the terminal. As well as just compare the output

```
e does not exist in graph
```

```
e does not exist in graph
```

```
Edge from a to e does not exist
```

Github Commits:

Feature 1:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/13da5df15d70f137599f0d0548c08deca4fbdecf>

Feature 2:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/82dfd0cf6478be38794919aa65f49d4b758ccca6>

Feature 3:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/66fa44ecc89a2b389f3e56fd61a6584742bd2ac1>

Feature 4:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/21d75d8c94ba273ff7bbd3bed34fe07e4f2d3cc1>

Feature Remove:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/52c998e285aa93415b7ad7906e72960d2268cbeb>

1st Refactor:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/caffc21b4a15021d73befc831654b6dff0d17ca8>

2nd, 3rd, 4th Refactor:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/1f664ae68174a492238adc8f973147b8ae710abf>

5 Refactors:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/afd7239346c91139e9b6df03d8e6188e29b5ed12>

BFS and DFS Template Pattern:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/d52112709187ebc2cf601f66e10d87d64fef4357>

Explanation:

Created two different classes for the BFS and DFS template pattern. They both get their template from the class SearchTemplate.java, and they are called within the Path.java and Project1.java. Separating both of the code.

BFS Strategy Pattern:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/fb5fab0a156a8ae7df9b3d523529027b4910d540>

Strategy pattern uses the implementation of Path search from graphStrategy to run the bfs

Code Review Pull Request:

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/pull/4>

RandomWalk Strategy and Template Pattern

<https://github.com/Sgomez47-ASU/CSE464-2023-Sgomez47/tree/eb89a68c76d0ea555f7510f30dec121cf70fb9c9>

Template Pattern: RandomWalkTemplate

Strategy Pattern: RandomWalkAlgorithm

Template Pattern uses the SearchTemplate.java file and StrategyTemplate uses the GraphStrategy.java, to separate the search methods between the randomwalk in order to recursively call it.