Github: https://github.com/Fonz-Hamilton/CSE-464-2024-Cahamil1

Course Project Part 3 For CSE 464

Features:

Commit 1:
Refactor 1: Added documentation. In this refactor I went through and gave detailed comments of all the methods in each class.
https://github.com/Fonz-Hamilton/CSE-464-2024-Cahamil1/commit/c8c738eec72f036d6a36f71d93c2baeb8303b4f6

Refactor 2: Added helper method. In this refactor I added a helper method for BFS and DFS to remove repeated code.
https://github.com/Fonz-Hamilton/CSE-464-2024-Cahamil1/commit/06801ff72d91651329730d2e25e66f048f2b0a76

Refactor 3: Cleaned up code. In this refactor I removed unnecessary imports and white space / comments
https://github.com/Fonz-Hamilton/CSE-464-2024-Cahamil1/commit/8c23d27448b6895c46a814200211c4b07b68872f

Refactor 4: Added more error handling. In this refactor I added more error handling for the input. This will prevent a crash if the searched nodes do not exist.
https://github.com/Fonz-Hamilton/CSE-464-2024-Cahamil1/commit/295c048a38d7f465f62634e3dc10dfe4ca226131

Refactor 5: Added tests for BFS and DFS. These were added to make sure the searches run as expected
https://github.com/Fonz-Hamilton/CSE-464-2024-Cahamil1/commit/fabdc35569fb1fe02acc65aa409c1c3552b87ba6

Commit 2:
Added a template design pattern for the BFS and DFS searches.
https://github.com/Fonz-Hamilton/CSE-464-2024-Cahamil1/commit/c29b86fb41d7db0f0dfa46e31d05b4fc9d936a39

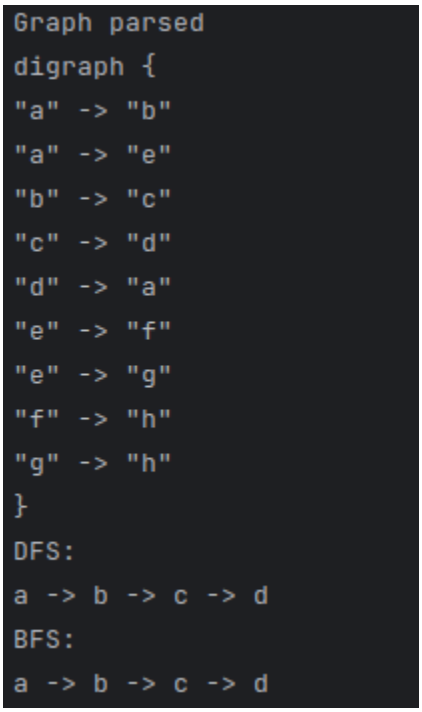
Commit 3:
Added a strategy design pattern for BFS and DFS searches
https://github.com/Fonz-Hamilton/CSE-464-2024-Cahamil1/commit/e18b02f9360f3d03a05403b9ad46097c19dad8b9

Commit 4:
Added random walk search. This searches randomly until it finds the required path
https://github.com/Fonz-Hamilton/CSE-464-2024-Cahamil1/commit/f8ab358693e8372b30b5162082835897b5503c29

To run BFS and DFS, simply provide the graph and use the graphSearch function:

```
Path path = dotGraph.graphSearch(dotGraph.getNode("a"), dotGraph.getNode("d"),
DOTGraph.Algorithm.DFS);
System.out.println("DFS: \n" + path.printPath());


path = dotGraph.graphSearch(dotGraph.getNode("a"), dotGraph.getNode("d"),
DOTGraph.Algorithm.BFS);
System.out.println("BFS: \n" + path.printPath());
```



The same goes for RANDOM_WALK

```
path = dotGraph.graphSearch(dotGraph.getNode("a"), dotGraph.getNode("c"),
DOTGraph.Algorithm.RANDOM_WALK);
System.out.println("Random Walk: \n" + path.printPath());
```

```
visted path: a
visted path: a e
visted path: a e f
visted path: a e f h
No path to destination, restarting
visted path: a
visted path: a b
visted path: a b c
Random Walk:
a -> b -> c

Process finished with exit code 0
```

```
visted path: a
visted path: a b
visted path: a b c
Random Walk:
a -> b -> c
```