CSE 464: Software QA and Testing

Project Part #2

Sabthagirivasan Vellingiri

1225471286

svellin3@asu.edu

Introduction:

Part 2 of the project was built using **IntelliJ** as the IDE, **Java 8** as the Java Development Kit, **Maven** for Dependency Manager, and the **graphviz-java** library shared by the professor. The **graphviz-java** was used to parse the **.dot** file into objects in java and vice versa. Custom classes such as Graph, Node, Edge, and Path were written to perform operations(such as adding or removing nodes or edges, searching source to destination path using DFS or BFS) on the given graph.

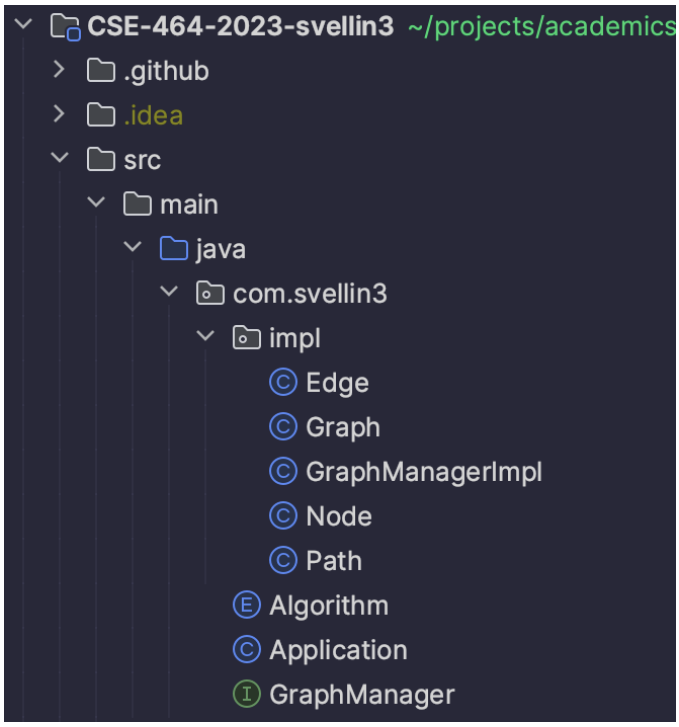
If there are any dependency issues, please resolve them using **mvn package** in the command line terminal or resolve them using maven in IntelliJ.

All the features required in part 1 and part 2 are added into an Interface named "**GraphManager.java**" which is in the folder: **src/main/java/com/svellin3.** The implementation of all these features is implemented in **"GraphManagerImpl.java"** which resides in the folder: **src/main/java/com/svellin3/impl.**

GitHub Link: **https://github.com/sabthagirivasanv/CSE-464-2023-svellin3.git**

How to run the application:

To run the application, run the main function that resides in the **Application.java** which can be located inside the folder: **src/main/java/com/svellin3**



The Command Line Terminal can be used to interact with the application. Initially, the application will ask for a **.dot** file to parse the graph. I have added my input file: **testInput.dot** to the project folder.

```
please enter the dot file to parse the graph:
testInput.dot
```

Once the dot file was parsed, the application will display a list of options to perform on the parsed graph. The screenshot of the options is attached below.

```
please enter the dot file to parse the graph:
testInput.dot


press the below options to perform actions:
1. print the graph
2. output to file
3. add a new node
4. add a list of new nodes
5. remove a node
6. remove a list of nodes
7. add an edge
8. remove an edge
9. output as DOT graph
10. output into graphics
11. search nodes by BFS
12. search nodes by DFS
13. exit
```

You can manipulate the graph using the options developed in Phase-1.

Option 11: Search the path from source to destination using BFS

Now, enter the source node and destination node between which you want to find the path using BFS. Our algorithm will return the optimum path using the BFS algorithm. If no such path exists, it will return null.

```
11
Please enter the source node:
a
Please enter the destination node:
g
The path is :
a -> b -> c -> d -> e -> f -> g
```

Option 12: Search the path from source to destination using DFS

Now, enter the source node and destination node between which you want to find the path using DFS. Our algorithm will return the optimum path using the DFS algorithm. If no such path exists, it will return null.

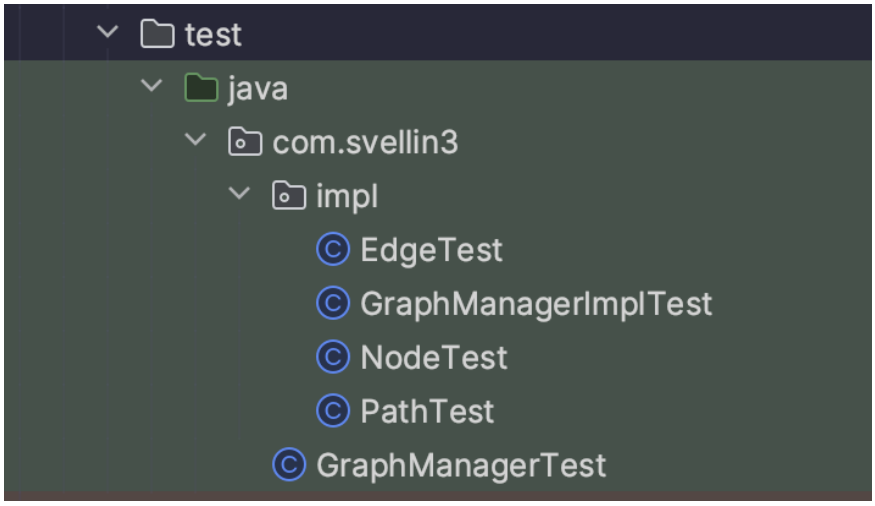
```
12
Please enter the source node:
d
Please enter the destination node:
g
The path is :
d -> e -> f -> g
```

How to run the Test:

All the features required in part-1 and part-2 are added into an Interface named "**GraphManager.java**" which is in the folder: **src/main/java/com/svellin3.** The implementation of all these features is implemented in **"GraphManagerImpl.java"** which resides in the folder: **src/main/java/com/svellin3/impl.**

The unit test cases were written for all the features implemented in part-1 and part-2. The test class is added inside the folder "**src/test/java/com/svellin3**"**.** All the required features for part-1 and part-2 are exclusively tested in the java file named "**GraphManagerTest.java**" which is located in the folder "**src/test/java/com/svellin3**". All other internal functions which are essential in achieving the required functionalities are tested in separate classes which reside inside **src/test/java/com/svellin3/impl**.

The test cases can be run using **IntelliJ** or **mvn test**. I am hereby attaching the folder structure for reference.



GitHub Commits:

I created two new branches: **bfs** and **dfs** to develop graph search using BFS and DFS respectively. I am hereby attaching all my commits in those branches and the **main** branch.

bfs branch commit link:

https://github.com/sabthagirivasanv/CSE-464-2023-svellin3/commits/bfs

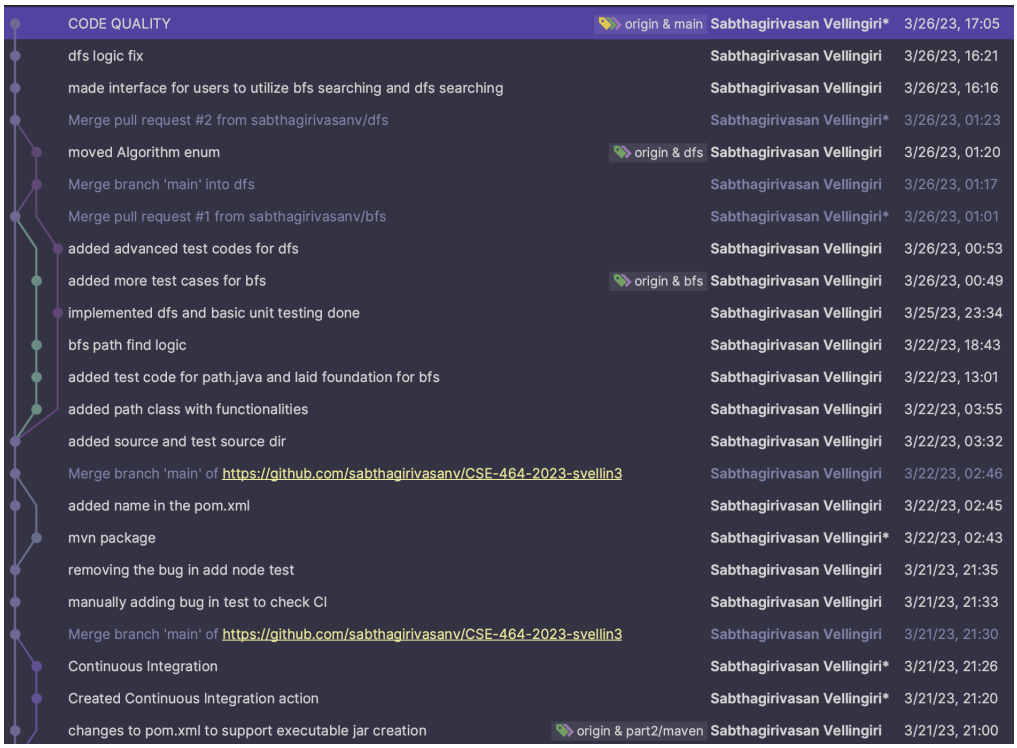
dfs branch commit link:

https://github.com/sabthagirivasanv/CSE-464-2023-svellin3/commits/dfs

main branch commit link:

https://github.com/sabthagirivasanv/CSE-464-2023-svellin3/commits/main

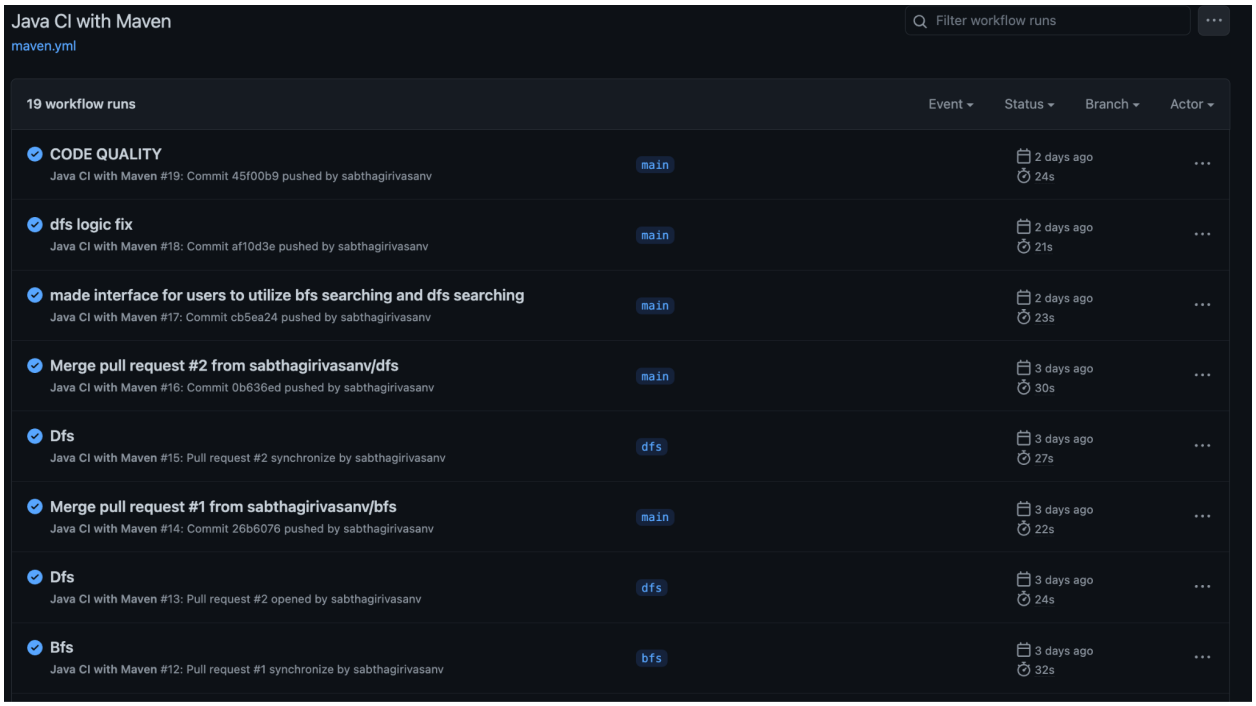
Git Commit Logs:



GitHub workflow results:

GitHub Action link:

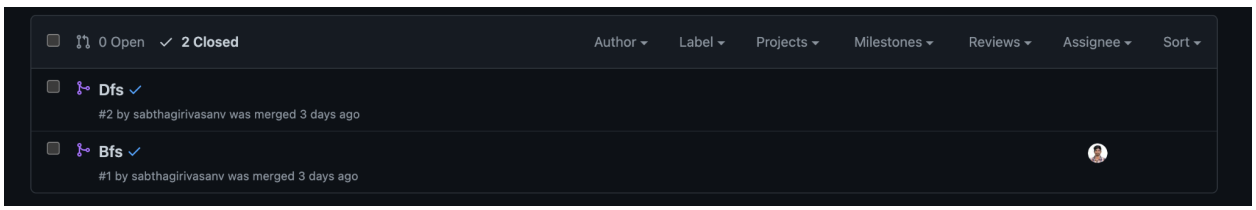
https://github.com/sabthagirivasanv/CSE-464-2023-svellin3/actions/workflows/maven.yml



Pull Requests:

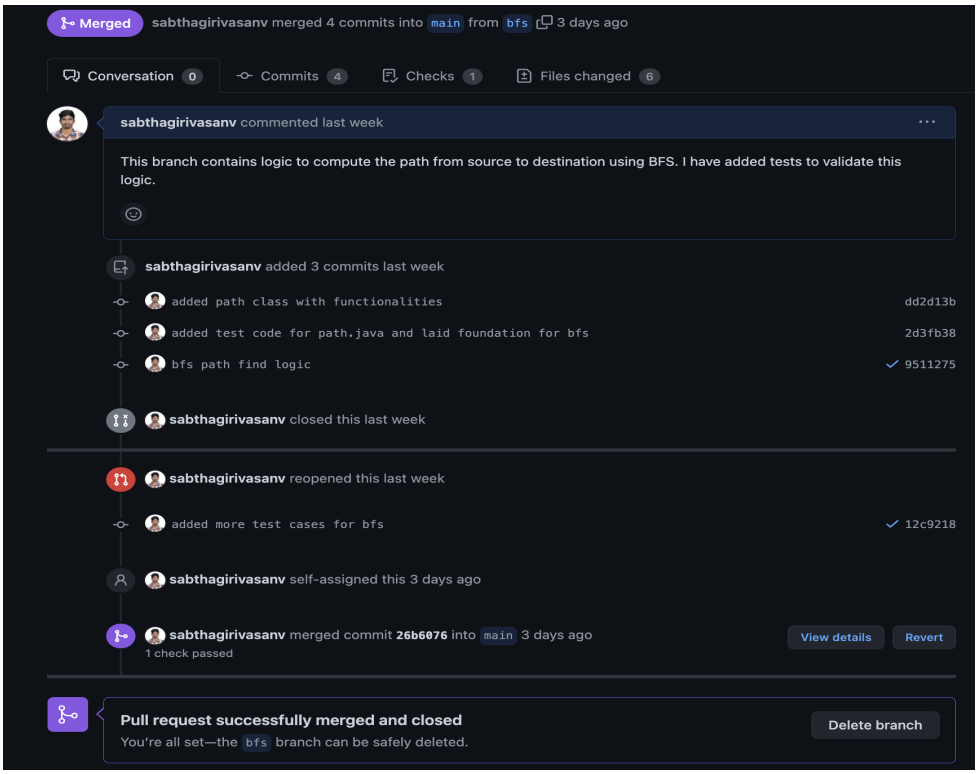
I created two pull requests to merge **bfs** branch and **dfs** branch changes into the main branch. Firstly, I merged the **bfs** branch into the **main** branch. Then, merged the latest changes from the **main** branch into the **dfs** branch. After resolving the conflicts in GraphSearch API using **Algorithm. enum,** I created another pull request to merge the dfs branch into the main branch.

I am hereby attaching the status of both Pull requests.



Pull Request of BFS branch:

Pull request link: https://github.com/sabthagirivasanv/CSE-464-2023-svellin3/pull/1



Pull Request of DFS branch:

Pull request link: https://github.com/sabthagirivasanv/CSE-464-2023-svellin3/pull/2