GitHub Repo: https://github.com/accfmax95/CSE-464-2023-mjberry4

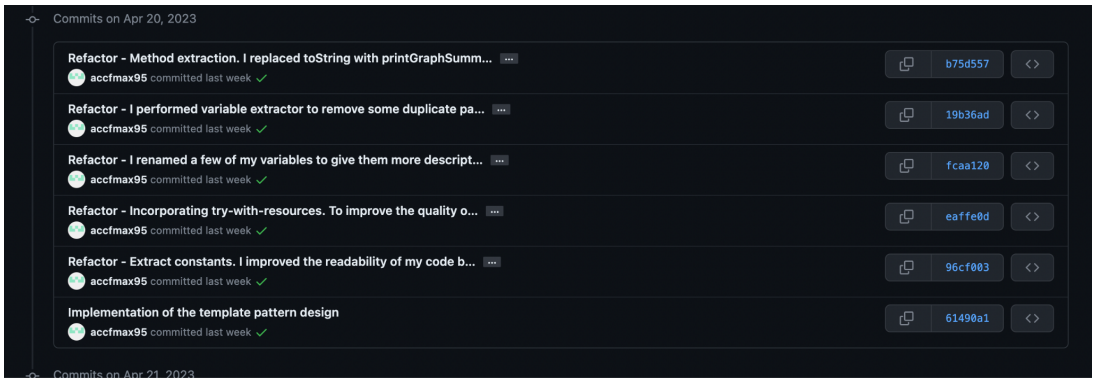
How to Run the BFS, DFS, and RandomWalk Functions -

1. Once the code has been downloaded and opened in IntelliJ (or whatever IDE you are using), type 'mvn package' into your IDE's terminal. This will install the necessary libraries onto your project structure. NOTE: If you want to run the program with the .jar file created by maven, type this command: 'mvn clean compile assembly:single'. This will install the .jar file with the necessary dependencies to execute the Part1.java file. This base file has just a few lines to test the GraphManager. You can modify this file for further testing.
2. Once you run 'mvn package', you will notice the test cases run in console. All test cases should pass if the BFS, DFS, and RandomWalk functions are working correctly.
3. If you would like to test the program manually, you can open the GraphManagerTest.java file and run the testGraphSearch function at the bottom. The inputs I am using for BFS, DFS, and RandomWalk are as follows:
    a. BFS - GraphSearch(src, dst, g.graph, Algorithm.BFS) for "a -> h"
        i. Output should be "a -> b -> e -> h"
    b. DFS - GraphSearch(src, dst, g.graph, Algorithm.DFS) for "a -> h"
        i. Output should be "a -> c -> e -> h"
    c. BFS - GraphSearch(src, dst, g.graph, Algorithm.BFS) for "d -> h"
        i. Output should be "d -> f -> h"
    d. DFS - GraphSearch(src, dst, g.graph, Algorithm.DFS) for "d -> h"
        i. Output should be "d -> g -> h"
    e. RANDOM_WALK - GraphSearch(src, dst, g.graph, Algorithm.BFS) for "a -> h"
        i. Output is not known, since each time the program runs it should be randomized.
    f. RANDOM_WALK - GraphSearch(src, dst, g.graph, Algorithm.DFS) for "a -> h"
        i. Output is not known, since each time the program runs it should be randomized.
    g. NOTE: If you want to test different inputs by modifying the test functions, you will also have to modify what the output should be in order for the new test case to pass. To change the src value and dst value, simply modify the letter of the assigned node for the src and dst variables in the test function (They are "a" and "h" as default).
4. Here is a screenshot of the correct output when running 'mvn package'. As you can see, all the test cases pass, and I have displayed the output of both the DFS and BFS algorithm for the inputs shown above. I have also included the output for the RandomWalk algorithm. Each step of the algorithm is displayed in order to verify the randomness of the algorithm.

```
Visiting Path: a
Visiting Path: a -> b
Visiting Path: a -> b -> e
Visiting Path: a -> b -> e -> h
BFS Output: a -> b -> e -> h
Visiting Path: a
Visiting Path: a -> c
Visiting Path: a -> c -> e
Visiting Path: a -> c -> e -> h
DFS Output: a -> c -> e -> h
Visiting Path: d
Visiting Path: d -> f
Visiting Path: d -> f -> h
BFS Output: d -> f -> h
Visiting Path: d
Visiting Path: d -> g
Visiting Path: d -> g -> h
DFS Output: d -> g -> h
Visiting Path: a
Visiting Path: a -> c
Visiting Path: a -> c -> e
Visiting Path: a -> c -> e -> h
RW Output: a -> c -> e -> h
Visiting Path: a
Visiting Path: a -> b
Visiting Path: a -> b -> f
Visiting Path: a -> b -> f -> h
RW Output: a -> b -> f -> h
```

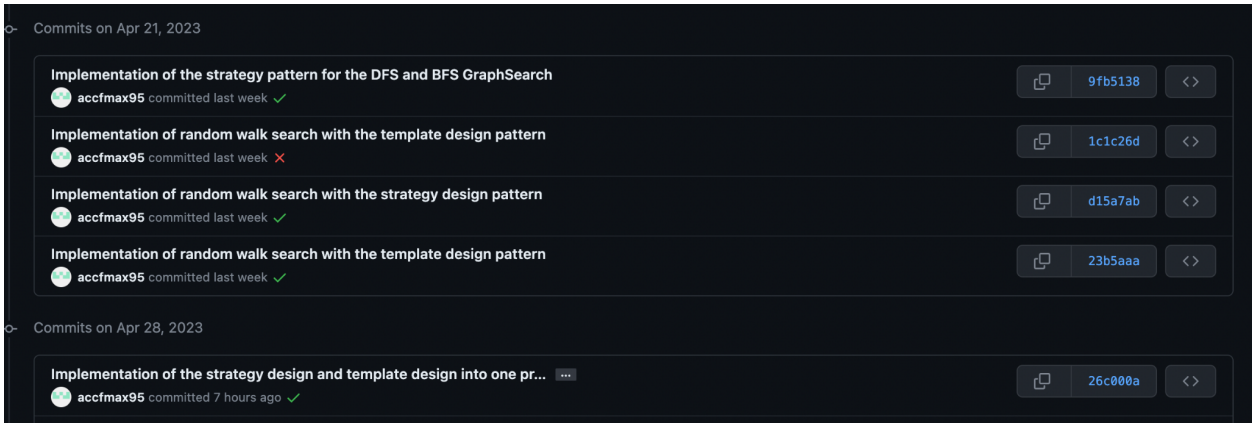
GitHub Commits, Branches, and Merges -

1. Here is a history of my commits, merges and everything else listed in the Actions tab of my GitHub: https://github.com/accfmax95/CSE-464-2023-mjberry4/actions
2. You can view a specific history of my commits here: https://github.com/accfmax95/CSE-464-2023-mjberry4/compare/master...refactor
3. Here is a list of my refactor commits. Each commit description will have a detailed description of what I did to refactor my code:

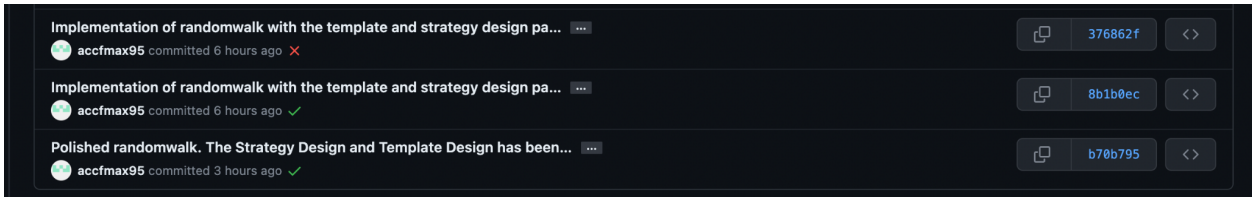


   a. https://github.com/accfmax95/CSE-464-2023-mjberry4/commit/b75d55743d0187b0653b6f80405fbe3d24cca55f
   b. https://github.com/accfmax95/CSE-464-2023-mjberry4/commit/19b36ad38c53ad5e90f478874680163829c5c850

    c. https://github.com/accfmax95/CSE-464-2023-mjberry4/commit/fcaa1204f6f0b94cec12058ea8f0556ecf48894c
    d. https://github.com/accfmax95/CSE-464-2023-mjberry4/commit/eaffe0dee0f69fcbf4575dc698aa49cdba0b65a7
    e. https://github.com/accfmax95/CSE-464-2023-mjberry4/commit/96cf0038cff56c83790f9b95380ecc3b8b327413
4. Here is the commit for my original template design for the DFS and BFS algorithms:
    a. https://github.com/accfmax95/CSE-464-2023-mjberry4/commit/61490a17f15d49f6e0fb849f90b685f849f8cf0c
5. For the strategy design, I had a long list of commits. I originally committed the code separately from the template design, as I assumed they would be graded separately. I then realized the template design and strategy design needed to be implemented together. Here is my commit for that code:



    a. https://github.com/accfmax95/CSE-464-2023-mjberry4/commit/26c000a06f719674aeafbe94828916d24b9266f5
6. Finally, I implemented my RandomWalk algorithm using both the template design and the strategy design. I went through a few different commits to get a design I was happy with. I will link the most recent design that I consider to be the finished product:



    a. https://github.com/accfmax95/CSE-464-2023-mjberry4/commit/b70b795eb98754f21a37c8c6977b99725a88cff7

7. In this screenshot, you can see that my Continuous Integration is set up properly. This is one example of the build being complete.