

# Readme

I have implemented the features of the project by making a class for graph named myGraphClass.

**GitHub Link:** [https://github.com/Divy2000/CSE464\\_2023\\_dplate164](https://github.com/Divy2000/CSE464_2023_dplate164)

Run the **src/main/application/Main.java** file to access the command-line interface.  
To run the test cases run **test/myGraphClass.java**

Following are the commands I have used to run my code :-

“src/main/java/testGraph.dot” -> Parses the .dot file

“0” -> To print the graph details

“1” -> To enter the details of the graph to a file

“src/main/java/testGraph.txt” -> File-path for the output

“2” -> To add node to the graph

“node0” -> Label of the node to add

“3” -> To add multiple nodes to the graph

“node1,node2,node3” -> Three nodes to be added to the graph

“4” -> To add an edge to the graph

“white” -> First node of the edge

“node3” -> Second node of the edge

“5” -> To remove a node from the graph

“yellow” -> Label of the node to remove

“6” -> To remove multiple nodes from the graph

“node1,black” -> 2 nodes to remove

“7” -> To remove an edge from the graph

“blue” -> Label of the first node of the edge to remove

“cyan” -> Label of the second node of the edge to remove

“8” -> To output the graph to .dot file

“src/main/java/testGraph\_o.dot” -> Path of the output file

“9” -> To output the graph to an image

“src/main/java/testGraph.png” -> Path of the output image file

“Q” -> To exit the program

In addition to this, now we have added feature for graph search :-

“10” -> Perform BFS search or DFS search in the graph

“bfs” -> For Breadth First Search

OR

“dfs” -> For Depth First Search

# Running code screenshots and commit links :-

## Feature 1

- Parsing a DOT graph file

The screenshot shows the IntelliJ IDEA interface with the project 'CSE464\_2023\_dplate164' open. The 'Run' tool window is active, showing the command: '/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java ...'. The 'Run' dropdown menu is open, showing the path: 'src/main/testGraph.dot'. An error message from SLF4J is displayed: 'SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder". SLF4J: Defaulting to no-operation (NOP) logger implementation SLF4J: See <http://www.slf4j.org/codes.html#StaticLoggerBinder> for further details.' Below the error, the application's output is shown:

```
Choose one of the following options
0 -> Print graph details
1 -> Output graph details to a text file
2 -> Add a node to the graph
3 -> Add multiple nodes to the graph
4 -> Add an edge to the graph
5 -> Remove a node from the graph
6 -> Remove multiple nodes from the graph
7 -> Remove an edge from the graph
8 -> Output graph to .dot file
9 -> Output graph to an image file
10 -> Perform BFS search or DFS search in the graph
Q -> Exit the software
Enter your choice
0
There are 8 nodes in the graph.
The label of the nodes are :-
red
pink
green
blue
white
yellow
black
cyan
```

- Print the details of the graph

The screenshot shows the IntelliJ IDEA interface with the project 'CSE464\_2023\_dplate164' open. The 'Run' tool window is active, showing the command: '/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java ...'. The 'Run' dropdown menu is open, showing the path: 'src/main/testGraph.dot'. The application's output is displayed:

```
Enter your choice
0
There are 8 nodes in the graph.
The label of the nodes are :-
red
pink
green
blue
white
yellow
black
cyan

There are 12 edges in the graph.
The edges are :-
green--black
white--pink
white--cyan
cyan--green
yellow--red
white--yellow
pink--blue
yellow--green
pink--red
blue--black
red--black
cyan--blue
```

- Output the details to a file.

The screenshot shows the IntelliJ IDEA interface with the project 'CSE464\_2023\_dpage164' open. The code editor displays the Main.java file, which contains a main method that reads a graph from a file named 'testGraph.dot'. A terminal window below the editor shows the execution of the program, starting with a menu of options (0-10) and then prompting for the output file 'output.txt'. The message 'Details about graph are successfully written to the file' is displayed at the end.

```

no usages  ↗ Divy2000 +1
public class Main {
    no usages  ↗ Divy2000 +1
    public static void main(String[] args) {
        myGraphClass g = new myGraphClass();
        boolean running = true;
        Scanner inputScanner = new Scanner(System.in);
        System.out.println("Enter the path to the dot file");
        String graphFilePath = inputScanner.nextLine();
        try {
            g.parseGraph(graphFilePath);
        }
        catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}

```

Run: application.Main

Choose one of the following options  
0 -> Print graph details  
1 -> Output graph details to a text file  
2 -> Add a node to the graph  
3 -> Add multiple nodes to the graph  
4 -> Add an edge to the graph  
5 -> Remove a node from the graph  
6 -> Remove multiple nodes from the graph  
7 -> Remove an edge from the graph  
8 -> Output graph to .dot file  
9 -> Output graph to an image file  
10 -> Perform BFS search or DFS search in the graph  
Q -> Exit the software  
Enter your choice  
1  
Enter the filepath of the output file:  
output.txt  
Details about graph are successfully written to the file

Git Run TODO Problems Terminal Services Build Dependencies

**Commit link:** [https://github.com/Divy2000/CSE464\\_2023\\_dpage164/commit/992acecb2f21363762b58f52b81f04d088277959](https://github.com/Divy2000/CSE464_2023_dpage164/commit/992acecb2f21363762b58f52b81f04d088277959)

## Feature 2

- Add a node to the graph

The screenshot shows the IntelliJ IDEA interface with the project 'CSE464\_2023\_dpage164' open. The code editor displays the Main.java file, which contains a main method that reads a graph from a file named 'testGraph.dot'. A terminal window below the editor shows the execution of the program, starting with a menu of options (0-10) and then prompting for the label of the node to add. A tooltip indicates that externally added files can be added to Git. The message 'Externally added files can be added to Git' is displayed at the bottom right.

```

# Project exclude paths
/target/
.idea/
/src/main/resources/
src/main/testGraph_o.dot
.DS_Store
/out/
/example/
/ex1.png
output.txt

```

Run: application.Main

Choose one of the following options  
0 -> Print graph details  
1 -> Output graph details to a text file  
2 -> Add a node to the graph  
3 -> Add multiple nodes to the graph  
4 -> Add an edge to the graph  
5 -> Remove a node from the graph  
6 -> Remove multiple nodes from the graph  
7 -> Remove an edge from the graph  
8 -> Output graph to .dot file  
9 -> Output graph to an image file  
10 -> Perform BFS search or DFS search in the graph  
Q -> Exit the software  
Enter your choice  
2  
Enter the label of the node you want to add:  
newNode1

Externally added files can be added to Git  
View Files Always Add Don't Ask Again

Git Run TODO Problems Terminal Services Build Dependencies

- Add list of nodes to the graph

```

CSE464_2023_ddate164 / .gitignore
Project Commit Pull Requests Bookmarks Structure
  myGraphClass.java myGraphClassTest.java .gitignore Main.java myGraphClass.java Path.java pom.xml (CSE464-GraphManager) C Maven Notifications
    myGraphClass SearchType Path testGraph.dot testGraph_o_dot
      test myGraphClassTest target
        .gitignore CSE464_2023_ddate164.iml ex1.png output.txt pom.xml Readme.pdf
          External Libraries Scratches and Consoles
Run: application.Main
  Choose one of the following options
  0 -> Print graph details
  1 -> Output graph details to a text file
  2 -> Add a node to the graph
  3 -> Add multiple nodes to the graph
  4 -> Add an edge to the graph
  5 -> Remove a node from the graph
  6 -> Remove multiple nodes from the graph
  7 -> Remove an edge from the graph
  8 -> Output graph to .dot file
  9 -> Output graph to an image file
  10 -> Perform BFS search or DFS search in the graph
  Q -> Exit the software
  Enter your choice
  3
  Enter the labels of node you want to add separated by comma ('a,b,c'):
  newNode2, node1beRemoved
  Externally added files can be added to Git View Files Always Add Don't Ask Again
  Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (5 minutes ago)
  119:1 LF UTF-8 4 spaces P main

```

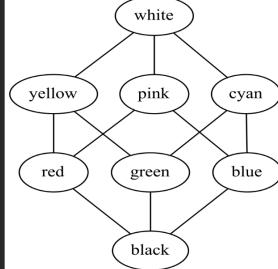
- Remove a node from the graph

```

CSE464_2023_ddate164 / .gitignore
Project Commit Pull Requests Bookmarks Structure
  myGraphClass.java myGraphClassTest.java .gitignore Main.java myGraphClass.java Path.java pom.xml (CSE464-GraphManager) C Maven Notifications
    myGraphClass SearchType Path testGraph.dot testGraph_o_dot
      test myGraphClassTest target
        .gitignore CSE464_2023_ddate164.iml ex1.png output.txt pom.xml Readme.pdf
          External Libraries Scratches and Consoles
Run: application.Main
  Choose one of the following options
  0 -> Print graph details
  1 -> Output graph details to a text file
  2 -> Add a node to the graph
  3 -> Add multiple nodes to the graph
  4 -> Add an edge to the graph
  5 -> Remove a node from the graph
  6 -> Remove multiple nodes from the graph
  7 -> Remove an edge from the graph
  8 -> Output graph to .dot file
  9 -> Output graph to an image file
  10 -> Perform BFS search or DFS search in the graph
  Q -> Exit the software
  Enter your choice
  5
  Enter the label of the node you want to remove:
  nodeToBeRemoved
  Externally added files can be added to Git View Files Always Add Don't Ask Again
  Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (6 minutes ago)
  137:1 LF UTF-8 4 spaces P main

```

- Remove multiple node from the graph



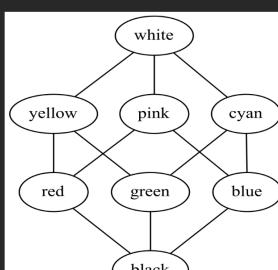
Choose one of the following options  
 0 -> Print graph details  
 1 -> Output graph details to a text file  
 2 -> Add a node to the graph  
 3 -> Add multiple nodes to the graph  
 4 -> Add an edge to the graph  
 5 -> Remove a node from the graph  
 6 -> Remove multiple nodes from the graph  
 7 -> Remove an edge from the graph  
 8 -> Output graph to .dot file  
 9 -> Output graph to an image file  
 10 -> Perform BFS search or DFS search in the graph  
 Q -> Exit the software  
 Enter your choice  
 6  
 Enter the labels of node you want to remove separated by comma ('a,b,c'): `newNode2, white`  
 Choose one of the following options

Externally added files can be added to Git  
[View Files](#) [Always Add](#) [Don't Ask Again](#)

**Commit Link:** [https://github.com/Divy2000/CSE464\\_2023\\_dpage164/commit/c2e5e0723c60eed8c52ad7ac50c373458055bff](https://github.com/Divy2000/CSE464_2023_dpage164/commit/c2e5e0723c60eed8c52ad7ac50c373458055bff)

## Feature 3

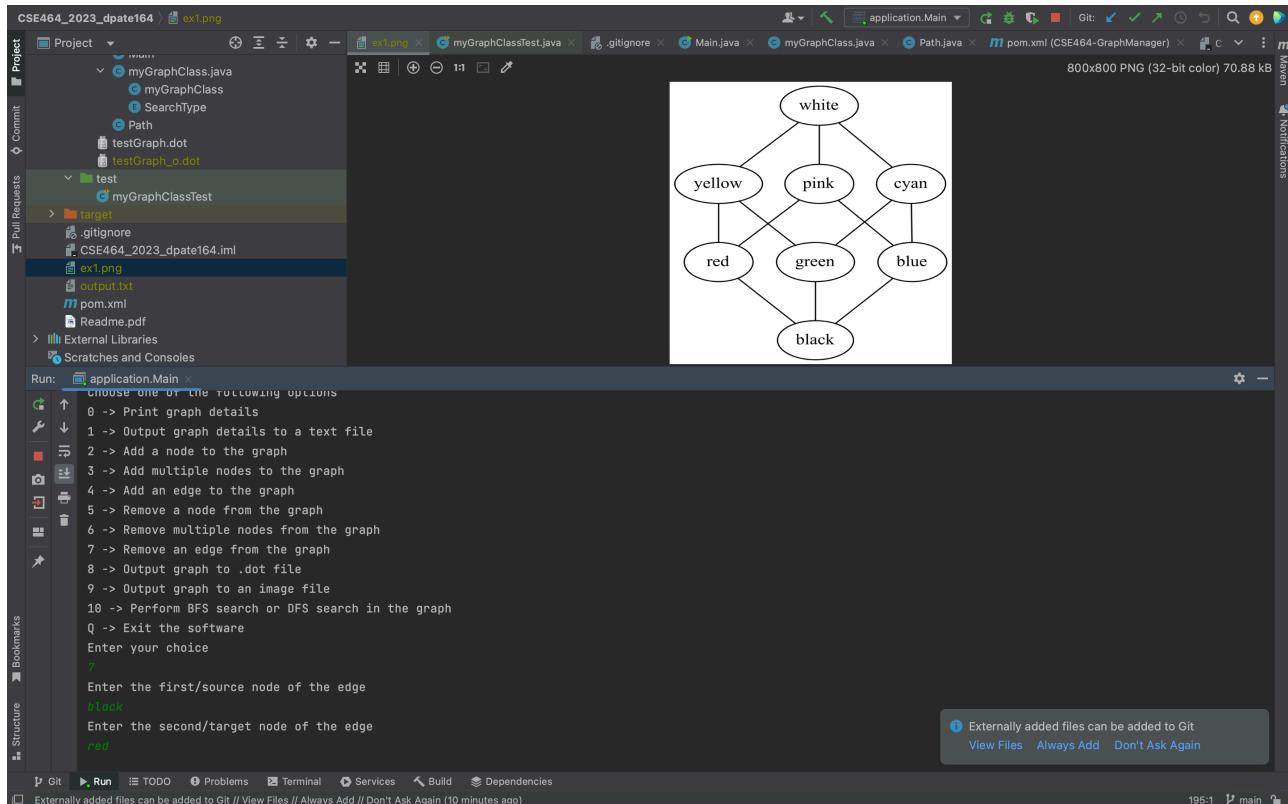
- Add an edge to the graph



Choose one of the following options  
 0 -> Print graph details  
 1 -> Output graph details to a text file  
 2 -> Add a node to the graph  
 3 -> Add multiple nodes to the graph  
 4 -> Add an edge to the graph  
 5 -> Remove a node from the graph  
 6 -> Remove multiple nodes from the graph  
 7 -> Remove an edge from the graph  
 8 -> Output graph to .dot file  
 9 -> Output graph to an image file  
 10 -> Perform BFS search or DFS search in the graph  
 Q -> Exit the software  
 Enter your choice  
 4  
 Enter the first/source node of the edge  
`newNode1`  
 Enter the second/target node of the edge  
`black`

Externally added files can be added to Git  
[View Files](#) [Always Add](#) [Don't Ask Again](#)

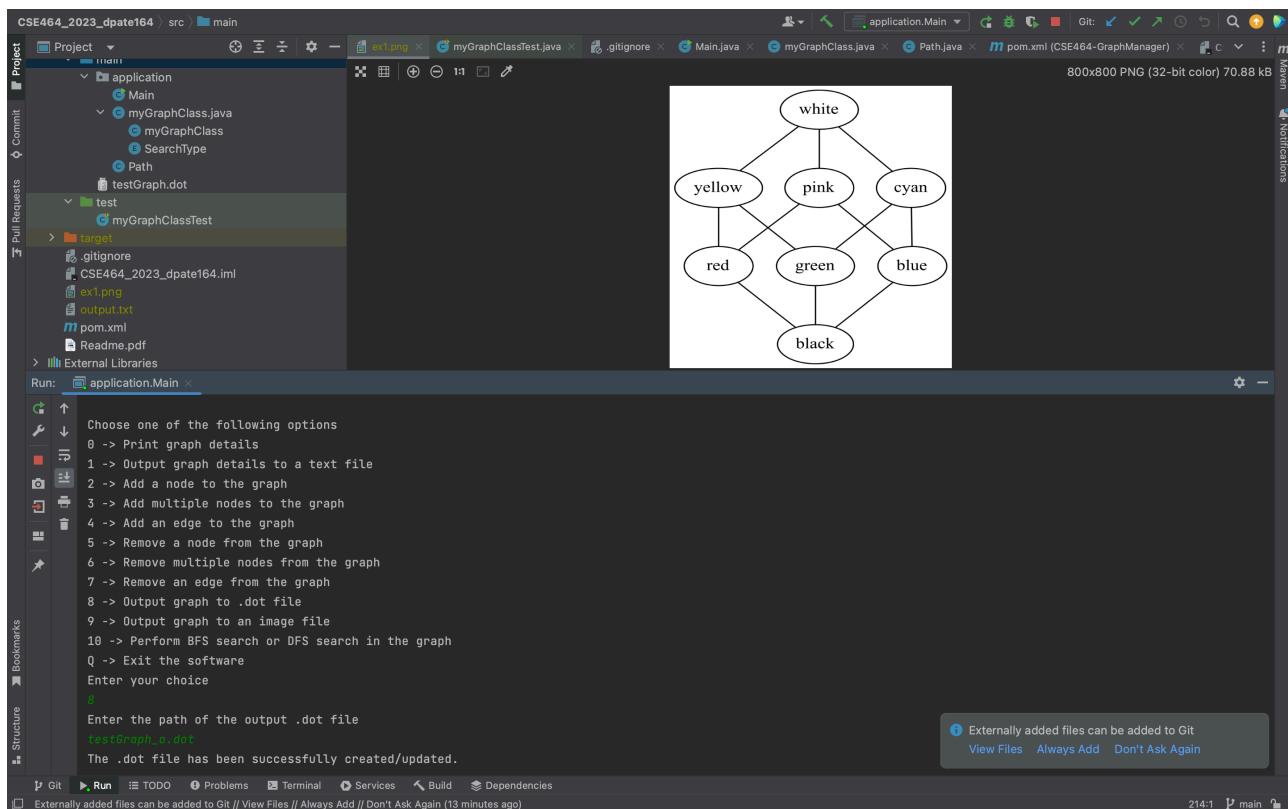
- Remove an edge from the graph



**Commit Link:** [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/commit/4fcf172714d5db0b8b19bc522782d1f3f1fa26b0](https://github.com/Divy2000/CSE464_2023_ddate164/commit/4fcf172714d5db0b8b19bc522782d1f3f1fa26b0)

## Feature 4

- Output the DOT graph file



- Output the image file

The screenshot shows the IntelliJ IDEA interface with the project 'CSE464\_2023\_ddate164' open. The code editor displays `Main.java` which contains logic for handling user input and performing graph operations. Below the code editor is a terminal window showing a menu of options for graph manipulation, such as printing details, adding nodes or edges, and outputting graphs to various formats. The bottom of the screen shows the IntelliJ navigation bar with icons for Git, Run, TODO, Problems, Terminal, Services, Build, and Dependencies.

```

case "6":
    System.out.println("Enter the labels of node you want to remove separated by comma ('a,b,c'):");
    nodeLabels = inputScanner.nextLine();
    nodeLabelList = nodeLabels.split( regex: "," );
    g.removeNodes(nodeLabelList);
    break;
case "7":
    System.out.println("Enter the first/source node of the edge");
    node1 = inputScanner.nextLine();
    System.out.println("Enter the second/target node of the edge");
    node2 = inputScanner.nextLine();
    g.removeEdge(node1, node2);
    break;
case "8":
    System.out.println("Enter the path of the output .dot file");
    filepath = inputScanner.nextLine();

```

**Commit Link:** [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/commit/f36ef90b4ae229af14958bf57da1d9c8179503c5](https://github.com/Divy2000/CSE464_2023_ddate164/commit/f36ef90b4ae229af14958bf57da1d9c8179503c5)

Final commit of finishing the testing portion: [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/commit/82b8581aa5068c9804c18beb58abe7cc01e21766](https://github.com/Divy2000/CSE464_2023_ddate164/commit/82b8581aa5068c9804c18beb58abe7cc01e21766)

Final commit for adding maven support: [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/commit/8c7363bda391c40d0865358f37e1ee66b59063dc](https://github.com/Divy2000/CSE464_2023_ddate164/commit/8c7363bda391c40d0865358f37e1ee66b59063dc)

Final commit for completing the continuous integration: [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/commit/669e2d62a7a94fbc0a038a561ab0c1f0aa705246](https://github.com/Divy2000/CSE464_2023_ddate164/commit/669e2d62a7a94fbc0a038a561ab0c1f0aa705246)

BFS Branch: [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/tree/bfs](https://github.com/Divy2000/CSE464_2023_ddate164/tree/bfs)

DFS Branch: [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/tree/dfs](https://github.com/Divy2000/CSE464_2023_ddate164/tree/dfs)

Adding BFS implementation: [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/commit/c65d58291562086c2838ff501920aae9e405d5bf](https://github.com/Divy2000/CSE464_2023_ddate164/commit/c65d58291562086c2838ff501920aae9e405d5bf)

Adding DFS implementation: [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/commit/b6c9a76f85e7a4d1f695e29629fe0a0943fe39b5](https://github.com/Divy2000/CSE464_2023_ddate164/commit/b6c9a76f85e7a4d1f695e29629fe0a0943fe39b5)

Merge branch BFS to main: [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/commit/119d64372419fbe526ec6756e19c5ca274fc16e6](https://github.com/Divy2000/CSE464_2023_ddate164/commit/119d64372419fbe526ec6756e19c5ca274fc16e6)

Merge branch DFS to main: [https://github.com/Divy2000/CSE464\\_2023\\_ddate164/commit/0192df91cf231a6e8ce38f5ef6508d49a49af2bc](https://github.com/Divy2000/CSE464_2023_ddate164/commit/0192df91cf231a6e8ce38f5ef6508d49a49af2bc)

- BFS search example

The screenshot shows the IntelliJ IDEA interface with the project 'CSE464\_2023\_dpage164' open. The Main.java file is the active editor. The code implements a menu-driven application for graph operations, including BFS search. The user has selected 'BFS search' (line 10) and entered 'white' as the source node (line 11). The search type is set to 'bfs' (line 12).

```

case "10":
    System.out.println("Performing BFS search");
    SearchType searchType = SearchType.BFS;
    String sourceNode = "white";
    String searchTypeStr = "bfs";
    search();
}

```

- DFS search example

The screenshot shows the IntelliJ IDEA interface with the project 'CSE464\_2023\_dpage164' open. The Main.java file is the active editor. The code implements a menu-driven application for graph operations, including DFS search. The user has selected 'DFS search' (line 10) and entered 'white' as the source node (line 11). The search type is set to 'dfs' (line 12).

```

case "10":
    System.out.println("Performing DFS search");
    SearchType searchType = SearchType.DFS;
    String sourceNode = "white";
    String searchTypeStr = "dfs";
    search();
}

```

## Project Part 3

### Part 1 - Code Refactoring

#### **1st Commit**

Extract Method Refactoring: [https://github.com/Divy2000/CSE464\\_2023\\_dplate164/commit/df43dd44ff8433b66ca8657331e6dc46126e20eb](https://github.com/Divy2000/CSE464_2023_dplate164/commit/df43dd44ff8433b66ca8657331e6dc46126e20eb)

#### **2nd Commit**

Inline Refactoring: [https://github.com/Divy2000/CSE464\\_2023\\_dplate164/commit/d3f43769b68d95cb1adb44e9cc0315068fc2512b](https://github.com/Divy2000/CSE464_2023_dplate164/commit/d3f43769b68d95cb1adb44e9cc0315068fc2512b)

#### **3rd Commit**

Renaming Variables: [https://github.com/Divy2000/CSE464\\_2023\\_dplate164/commit/da24acdaf5b617fd184807aed35d445859a9f898](https://github.com/Divy2000/CSE464_2023_dplate164/commit/da24acdaf5b617fd184807aed35d445859a9f898)

#### **4th Commit**

Replaced control flag with return statement: [https://github.com/Divy2000/CSE464\\_2023\\_dplate164/commit/20f9b45535ba22629360444e9d89ee9c0f0cb905](https://github.com/Divy2000/CSE464_2023_dplate164/commit/20f9b45535ba22629360444e9d89ee9c0f0cb905)

#### **5th Commit**

Removing the dead code: [https://github.com/Divy2000/CSE464\\_2023\\_dplate164/commit/9d85ac57609bcbfa33e74323049b1a3b75d63fd8](https://github.com/Divy2000/CSE464_2023_dplate164/commit/9d85ac57609bcbfa33e74323049b1a3b75d63fd8)

### Part 2 - Template Design Pattern

- In implementing template design pattern, I have extracted the common steps in “**src/main/application/GraphSearch.java**”.
- Both BFS and DFS have only one difference, BFS uses Queue while DFS uses stack.
- Therefor I have implemented everything in GraphSearch.java except the part where we add and remove from the queue/stack and the part where we check if queue/stack is empty.
- I have made 3 different abstract methods for that part :-
  - addToDataStructure
  - removeFromDataStructure
  - isEmpty (**Note: I changed the method name to isEmpty in next commit**)
- In BFS class I declare a Queue and implement abstract methods according to that.
- In DFS class I declare a Stack and implement abstract methods according to that.

**Commit Link:** [https://github.com/Divy2000/CSE464\\_2023\\_dplate164/commit/a79a63ad9249bf99742825b2c7acda88e04ec72f](https://github.com/Divy2000/CSE464_2023_dplate164/commit/a79a63ad9249bf99742825b2c7acda88e04ec72f)

### Part 3 - Strategy Design Pattern

- For this implementation I have created an abstract class GraphSearch\_Stragey, which takes MutableGraph and GraphSearch\_interface as its input to constructor.
- GraphSearch\_interface have 3 methods, addToDataStructure, removeFromDataStructure and isEmpty.
- Classes BFS and DFS implements these methods.
- The GraphSearch class extends the abstract class of GraphSearch\_Stragey and calls the super constructor.
- In GraphSearch method in muGraphClass, I am creating objects of the type of search and passing them to constructor of GraphSearch class.

**Commit link:** [https://github.com/Divy2000/CSE464\\_2023\\_dplate164/commit/e746659a58d4015f3e24ef76d6fa79e427f70aab](https://github.com/Divy2000/CSE464_2023_dplate164/commit/e746659a58d4015f3e24ef76d6fa79e427f70aab)