DAA LAB ASSIGNMENT

Name:- Sanidhya Varshney
Section:- A(53)
Univ. Roll.No.:- 191500714

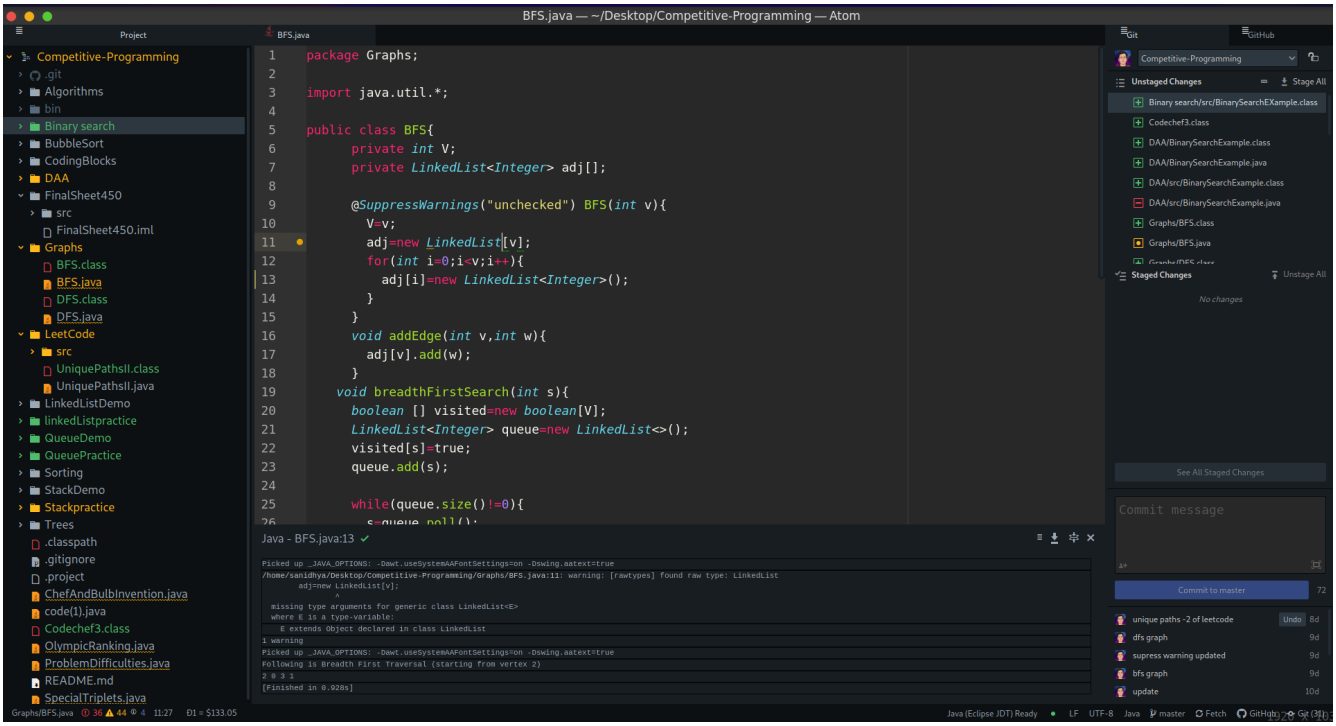
Question:- Implement and analyze the compexity of Breadth First Search and Depth First Search.

Breadth First Search

Code:-

```
package Graphs;

import java.util.*;

public class BFS{
      private int V;
      private LinkedList<Integer> adj[];

      @SuppressWarnings("unchecked") BFS(int v){
        V=v;
        adj=new LinkedList[v];
        for(int i=0;i<v;i++){
          adj[i]=new LinkedList<Integer>();
        }
      }
      void addEdge(int v,int w){
        adj[v].add(w);
      }
    void breadthFirstSearch(int s){
      boolean [] visited=new boolean[V];
      LinkedList<Integer> queue=new LinkedList<>();
```

```
      visited[s]=true;
      queue.add(s);

      while(queue.size()!=0){
        s=queue.poll();
        System.out.print(s+" ");

        Iterator<Integer> i=adj[s].listIterator();
        while(i.hasNext()){
          int n=i.next();
          if(!visited[n]){
            visited[n]=true;
            queue.add(n);
          }
        }
      }
    }
  public static void main(String[] args) {
    BFS bfs=new BFS(4);
    bfs.addEdge(0, 1);
    bfs.addEdge(0, 2);
    bfs.addEdge(1, 2);
    bfs.addEdge(2, 0);
    bfs.addEdge(2, 3);
    bfs.addEdge(3, 3);
    System.out.println("Following is Breadth First Traversal "+"(starting from
vertex 2)");
    bfs.breadthFirstSearch(2);
  }
}
```



Time Complexiry of BFS:-

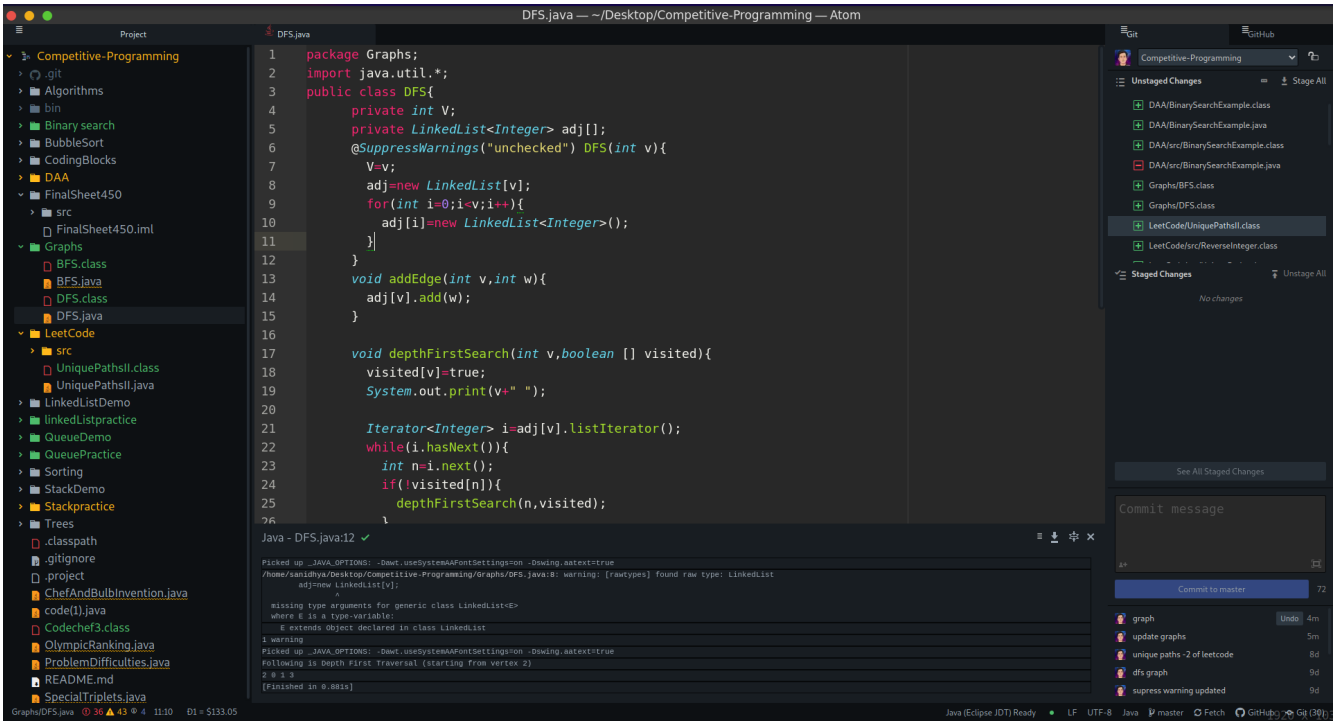
The Time complexity of BFS is O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges.

Depth First Search

Code:-

```
package Graphs;
import java.util.*;
public class DFS{
      private int V;
      private LinkedList<Integer> adj[];
      @SuppressWarnings("unchecked") DFS(int v){
        V=v;
        adj=new LinkedList[v];
        for(int i=0;i<v;i++){
          adj[i]=new LinkedList<Integer>();
        }
      }
      void addEdge(int v,int w){
        adj[v].add(w);
      }

      void depthFirstSearch(int v,boolean [] visited){
        visited[v]=true;
        System.out.print(v+" ");

        Iterator<Integer> i=adj[v].listIterator();
        while(i.hasNext()){
          int n=i.next();
          if(!visited[n]){
            depthFirstSearch(n,visited);
          }
        }
      }
      void depth(int v){
        boolean [] visited=new boolean[V];
```

```
    depthFirstSearch(v, visited);
  }
  public static void main(String[] args) {
    DFS dfs=new DFS(4);
    dfs.addEdge(0, 1);
    dfs.addEdge(0, 2);
    dfs.addEdge(1, 2);
    dfs.addEdge(2, 0);
    dfs.addEdge(2, 3);
    dfs.addEdge(3, 3);
    System.out.println("Following is Depth First Traversal "+"(starting from vertex 2)");
    dfs.depth(2);
  }
}
```



Time Complexity of DFS:-

The Time complexity of DFS is also O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges.