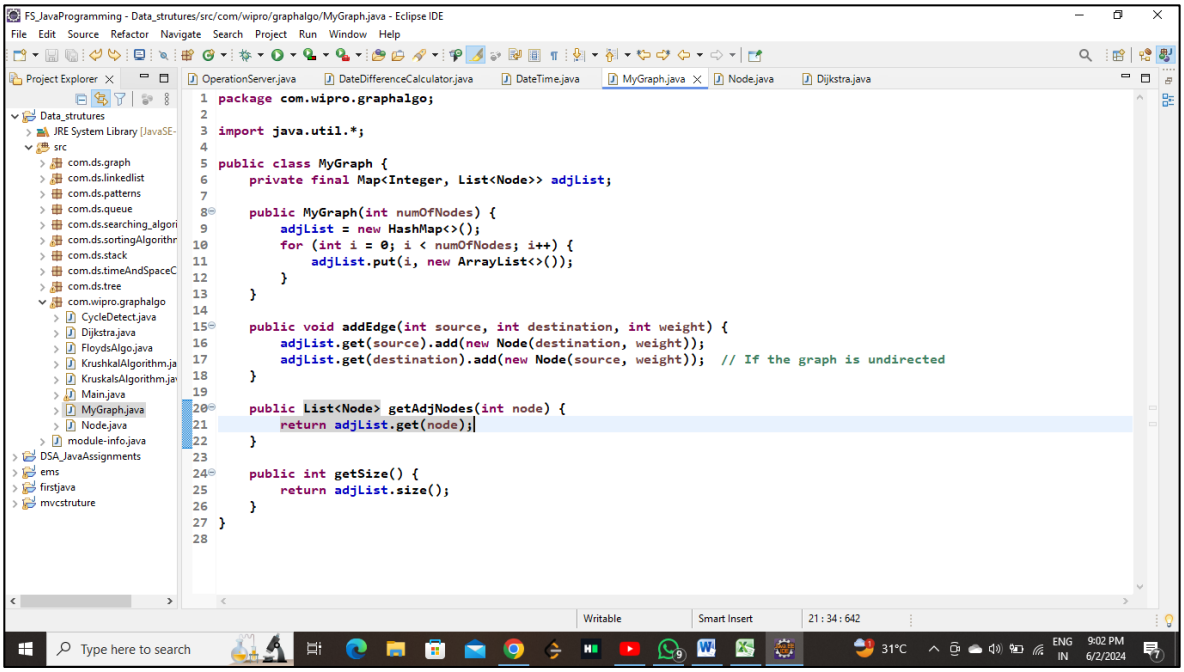
Name : Tejaswini Anil Kamble
Email : teju000kamble@gmail.com

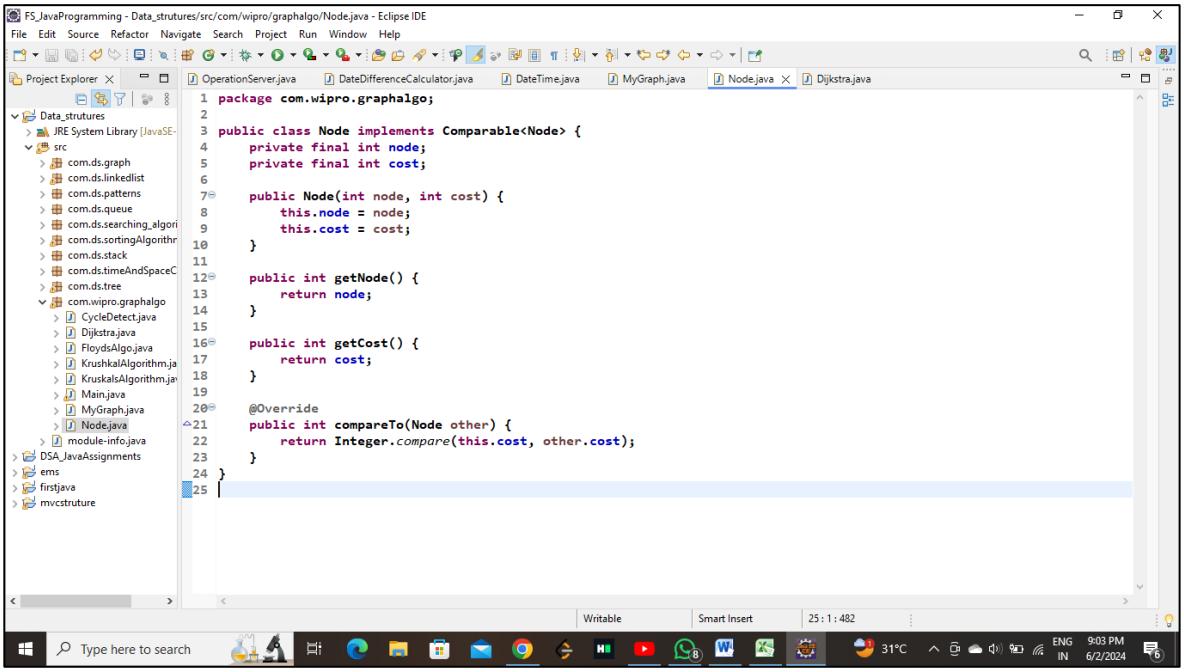
Day 9 and 10 : Assignments

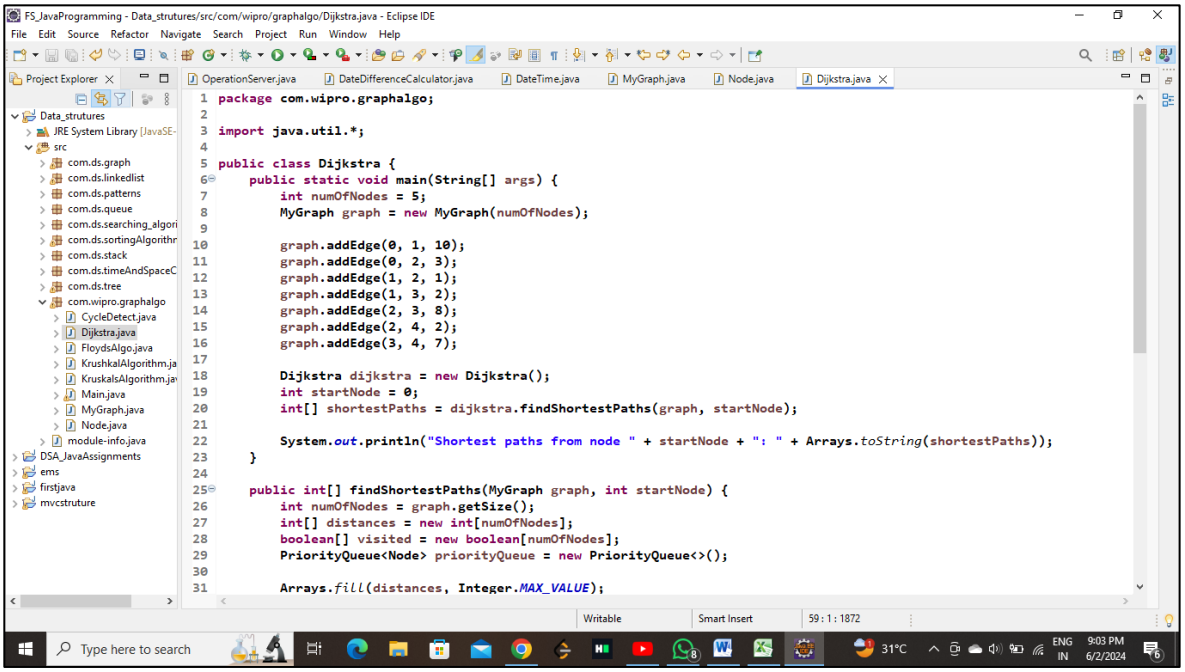
Task 1: Dijkstra's Shortest Path Finder

Code Dijkstra's algorithm to find the shortest path from a start nodeto every other node in a weighted graph with positive weights.

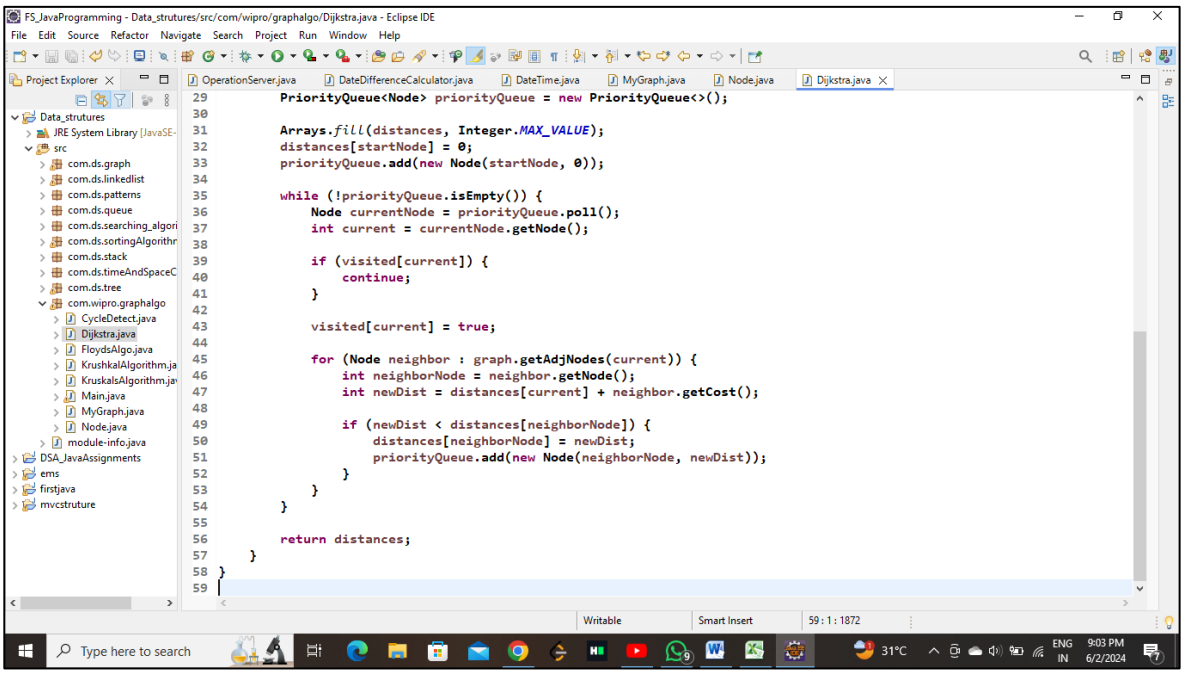
Ans: Source Code





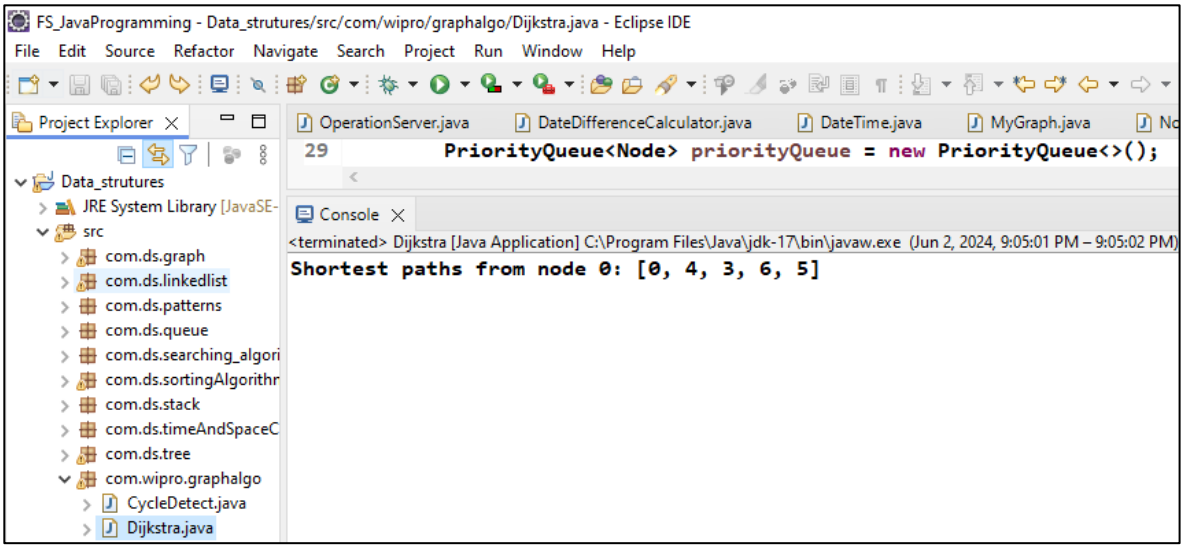



```java
package com.wipro.graphalgo;

import java.util.*;

public class Dijkstra {
    public static void main(String[] args) {
        int numOfNodes = 5;
        MyGraph graph = new MyGraph(numOfNodes);

        graph.addEdge(0, 1, 10);
        graph.addEdge(0, 2, 3);
        graph.addEdge(1, 2, 1);
        graph.addEdge(1, 3, 2);
        graph.addEdge(2, 3, 8);
        graph.addEdge(2, 4, 2);
        graph.addEdge(3, 4, 7);

        Dijkstra dijkstra = new Dijkstra();
        int startNode = 0;
        int[] shortestPaths = dijkstra.findShortestPaths(graph, startNode);

        System.out.println("Shortest paths from node " + startNode + ": " + Arrays.toString(shortestPaths));
    }

    public int[] findShortestPaths(MyGraph graph, int startNode) {
        int numOfNodes = graph.getSize();
        int[] distances = new int[numOfNodes];
        boolean[] visited = new boolean[numOfNodes];
        PriorityQueue<Node> priorityQueue = new PriorityQueue<>();

        Arrays.fill(distances, Integer.MAX_VALUE);
```




```java
        PriorityQueue<Node> priorityQueue = new PriorityQueue<>();

        Arrays.fill(distances, Integer.MAX_VALUE);
        distances[startNode] = 0;
        priorityQueue.add(new Node(startNode, 0));

        while (!priorityQueue.isEmpty()) {
            Node currentNode = priorityQueue.poll();
            int current = currentNode.getNode();

            if (visited[current]) {
                continue;
            }

            visited[current] = true;

            for (Node neighbor : graph.getAdjNodes(current)) {
                int neighborNode = neighbor.getNode();
                int newDist = distances[current] + neighbor.getCost();

                if (newDist < distances[neighborNode]) {
                    distances[neighborNode] = newDist;
                    priorityQueue.add(new Node(neighborNode, newDist));
                }
            }
        }

        return distances;
    }
}
```

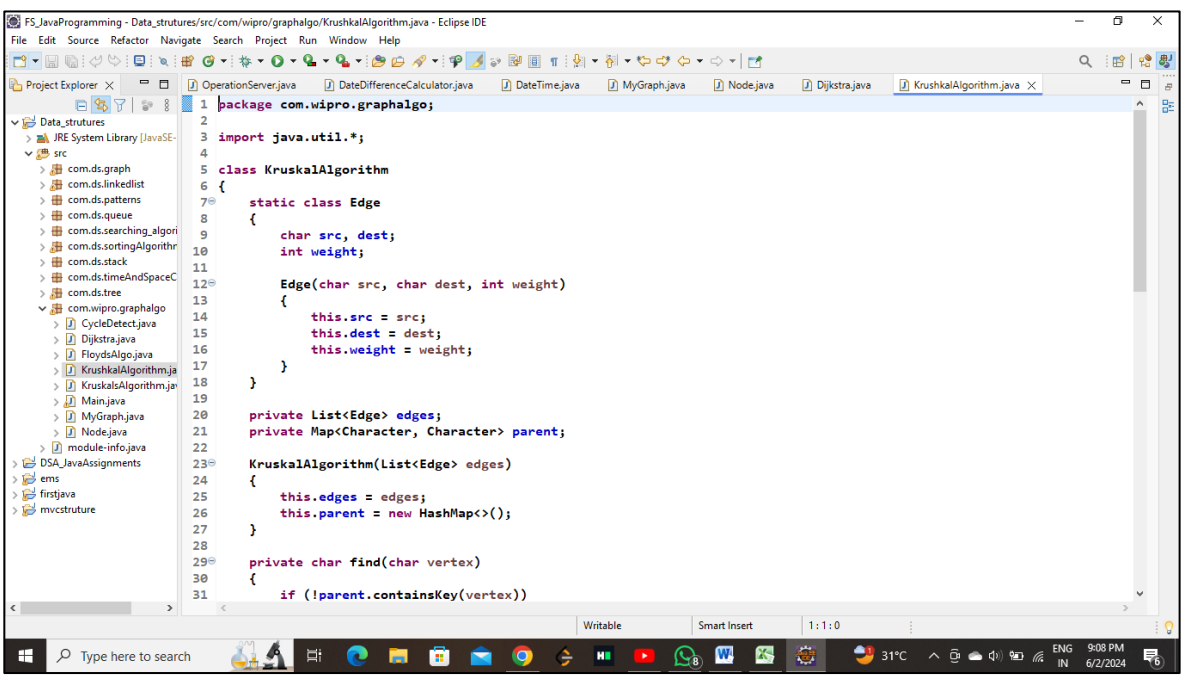
Output:

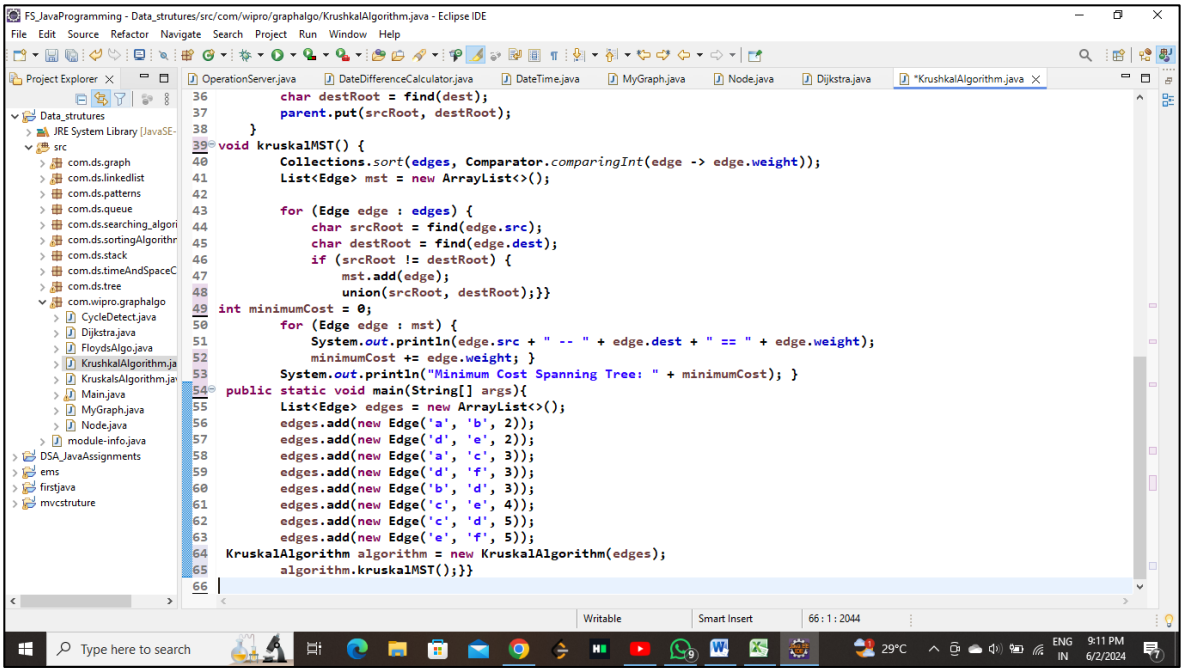


Task 2: Kruskal's Algorithm for MST

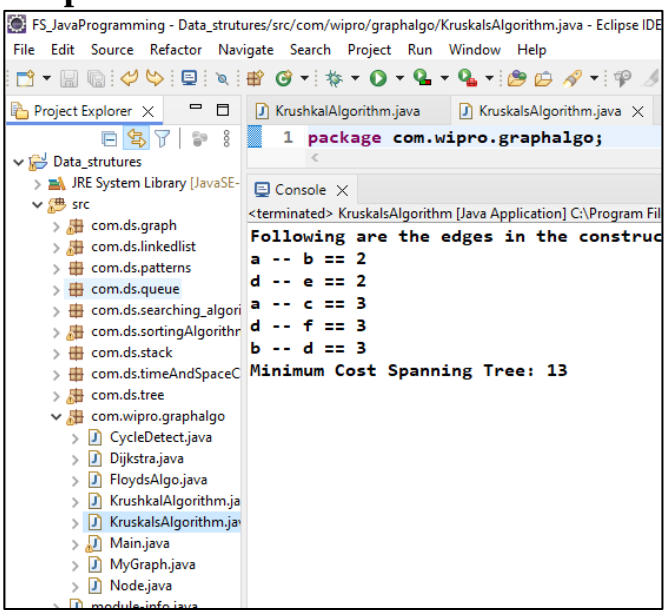
Implement Kruskal's algorithm to find the minimum spanning tree of a given connected, undirected graph with non-negative edge weights.

Ans: Source Code




FS_JavaProgramming - Data_strutures/src/com/wipro/graphalgo/KrushkalAlgorithm.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
OperationServer.java
DateDifferenceCalculator.java
DateTime.java
MyGraph.java
Node.java
Dijkstra.java
*KrushkalAlgorithm.java
Data_strutures
JRE System Library [JavaSE-
src
com.ds.graph
com.ds.linkedlist
com.ds.patterns
com.ds.queue
com.ds.searching_algori
com.ds.sortingAlgorithr
com.ds.stack
com.ds.timeAndSpaceC
com.ds.tree
com.wipro.graphalgo
CycleDetect.java
Dijkstra.java
FloydsAlgo.java
KrushkalAlgorithm.ja
KruskalsAlgorithm.ja
Main.java
MyGraph.java
Node.java
module-info.java
DSA_JavaAssignments
ems
firstjava
mvcstruture
36 char destRoot = find(dest);
37 parent.put(srcRoot, destRoot);
38 }
39 void kruskalMST() {
40 Collections.sort(edges, Comparator.comparingInt(edge -> edge.weight));
41 List<Edge> mst = new ArrayList<>();
42
43 for (Edge edge : edges) {
44 char srcRoot = find(edge.src);
45 char destRoot = find(edge.dest);
46 if (srcRoot != destRoot) {
47 mst.add(edge);
48 union(srcRoot, destRoot);}}
49 int minimumCost = 0;
50 for (Edge edge : mst) {
51 System.out.println(edge.src + " -- " + edge.dest + " == " + edge.weight);
52 minimumCost += edge.weight; }
53 System.out.println("Minimum Cost Spanning Tree: " + minimumCost); }
54 public static void main(String[] args){
55 List<Edge> edges = new ArrayList<>();
56 edges.add(new Edge('a', 'b', 2));
57 edges.add(new Edge('d', 'e', 2));
58 edges.add(new Edge('a', 'c', 3));
59 edges.add(new Edge('d', 'f', 3));
60 edges.add(new Edge('b', 'd', 3));
61 edges.add(new Edge('c', 'e', 4));
62 edges.add(new Edge('c', 'd', 5));
63 edges.add(new Edge('e', 'f', 5));
64 KruskalAlgorithm algorithm = new KruskalAlgorithm(edges);
65 algorithm.kruskalMST();}}
66
Writable
Smart Insert
66 : 1 : 2044
Type here to search
29°C
ENG IN
9:11 PM 6/2/2024

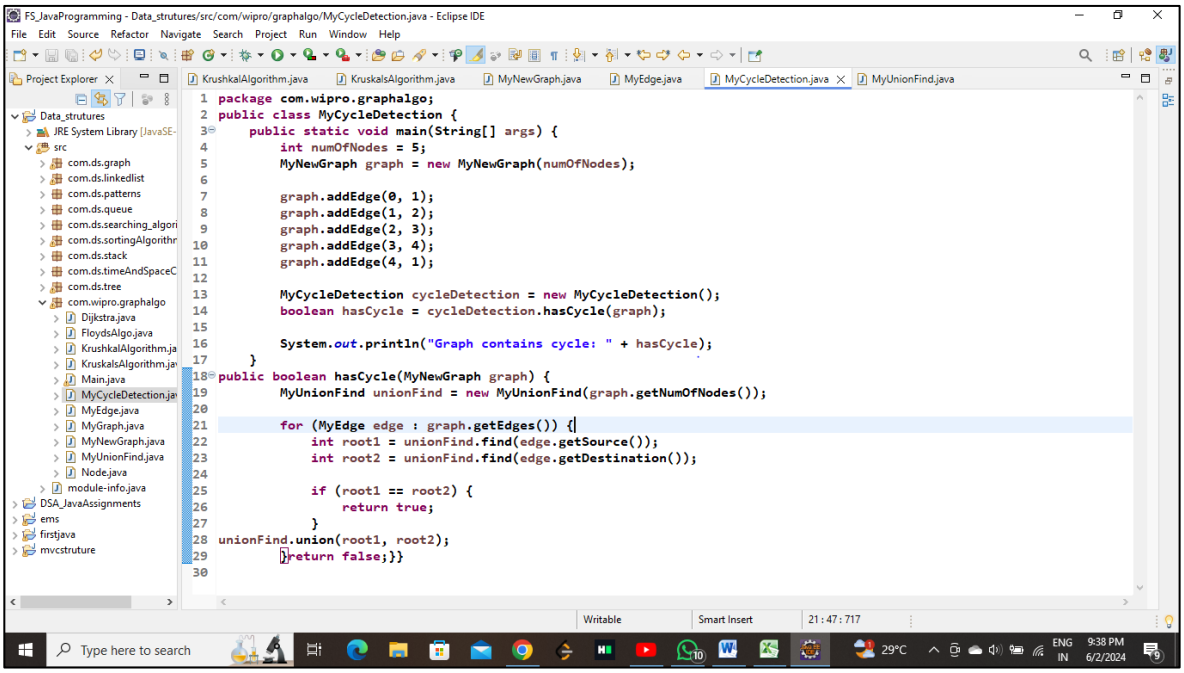
Output:

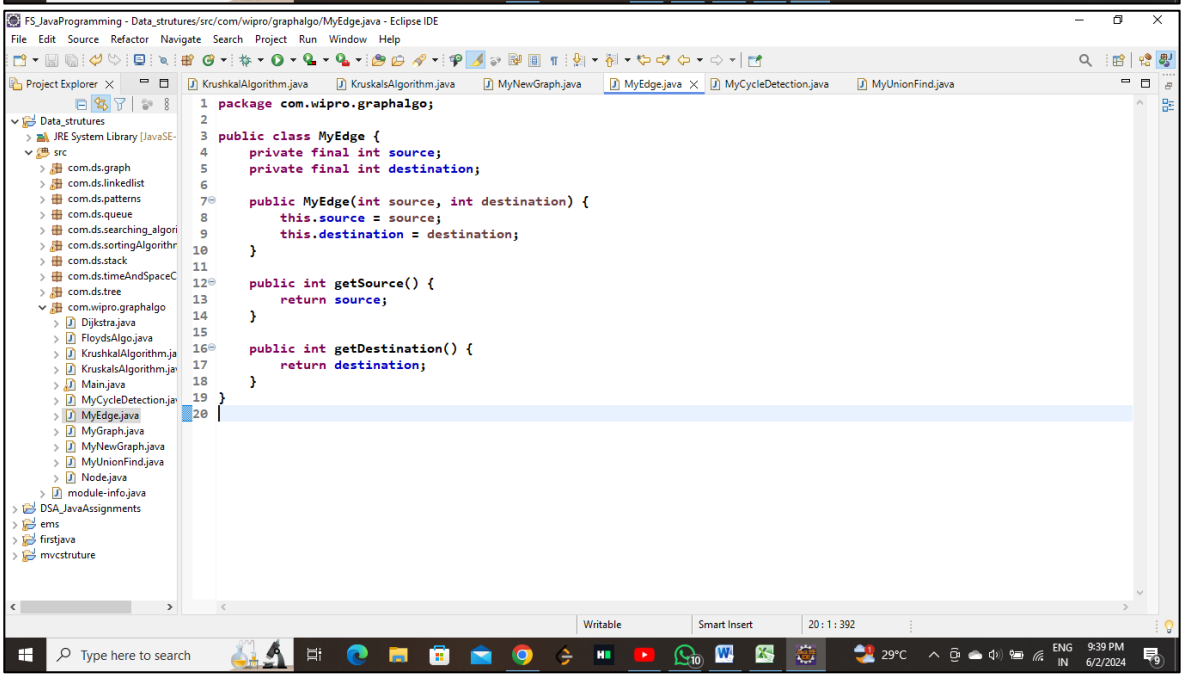

FS_JavaProgramming - Data_strutures/src/com/wipro/graphalgo/KruskalsAlgorithm.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
KrushkalAlgorithm.java
KruskalsAlgorithm.java
1 package com.wipro.graphalgo;
Console
<terminated> KruskalsAlgorithm [Java Application] C:\Program Fil
Following are the edges in the construc
a -- b == 2
d -- e == 2
a -- c == 3
d -- f == 3
b -- d == 3
Minimum Cost Spanning Tree: 13
Data_strutures
JRE System Library [JavaSE-
src
com.ds.graph
com.ds.linkedlist
com.ds.patterns
com.ds.queue
com.ds.searching_algori
com.ds.sortingAlgorithr
com.ds.stack
com.ds.timeAndSpaceC
com.ds.tree
com.wipro.graphalgo
CycleDetect.java
Dijkstra.java
FloydsAlgo.java
KrushkalAlgorithm.ja
KruskalsAlgorithm.ja
Main.java
MyGraph.java
Node.java

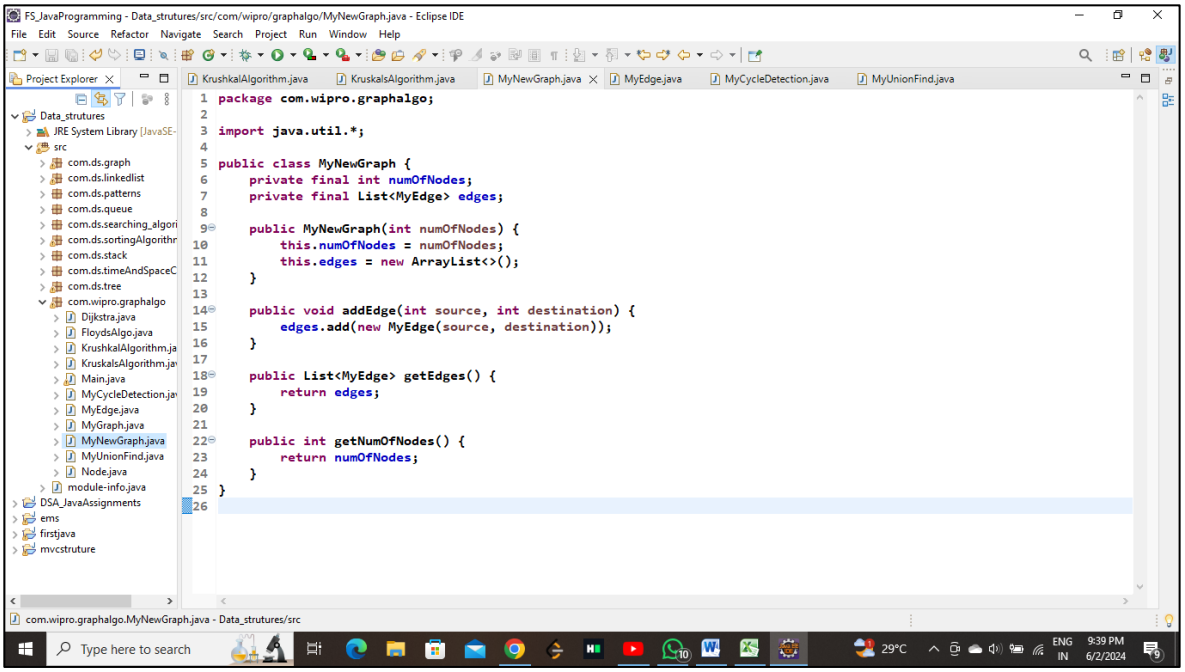
Task 3: Union-Find for Cycle Detection

Write a Union-Find data structure with path compression. Use this data structure to detect a cycle in an undirected graph.

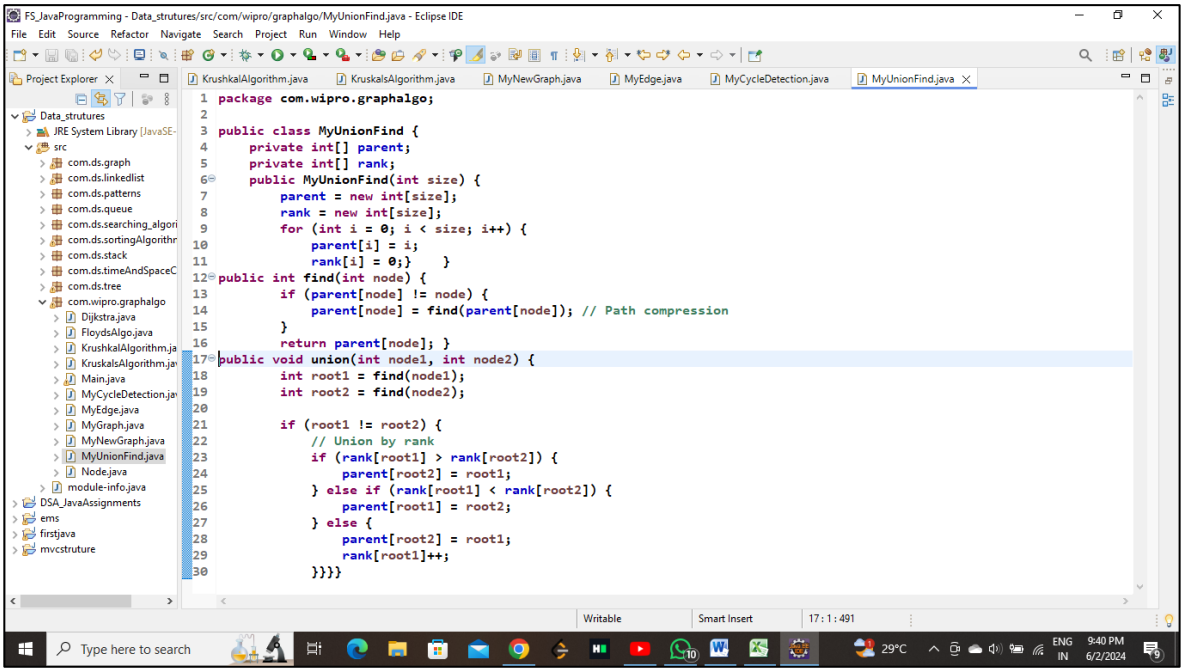
Ans: Source Code





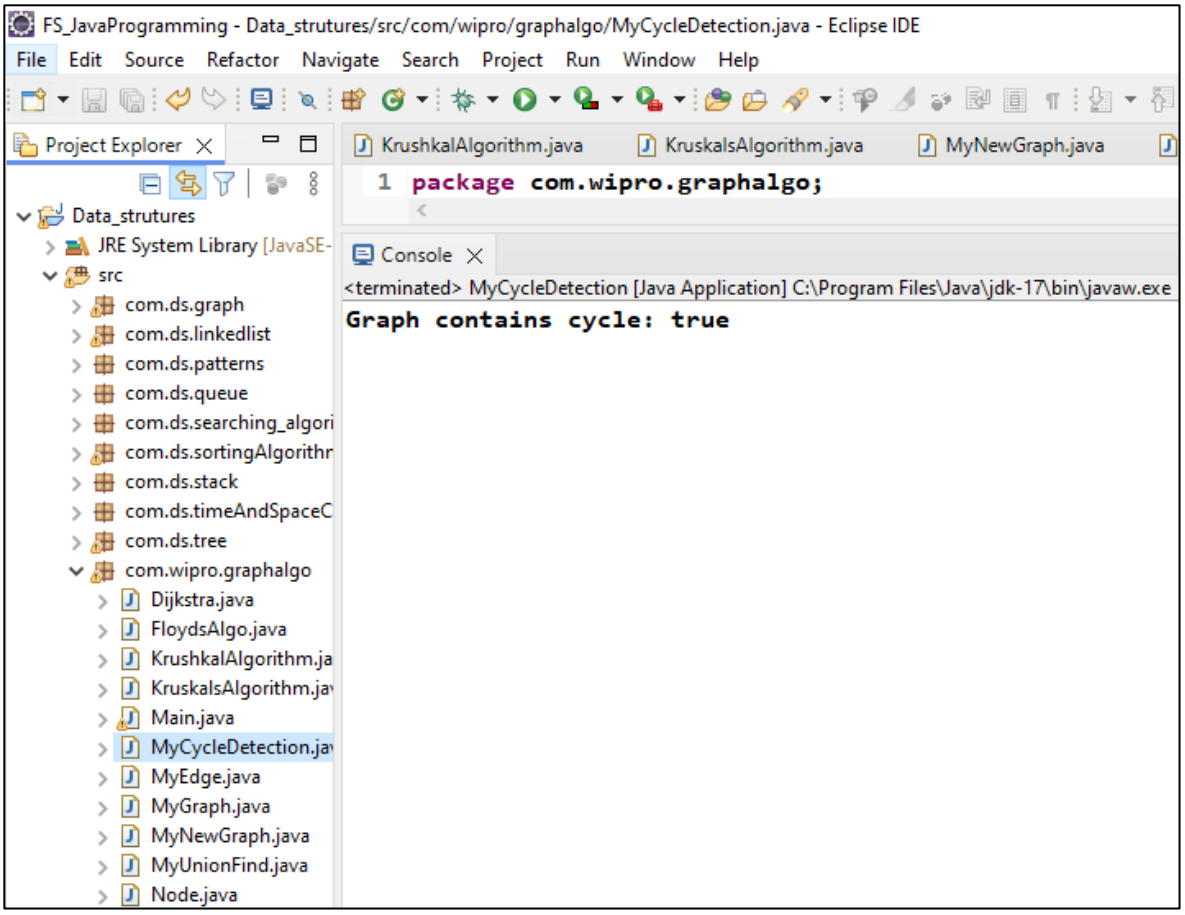



```java
package com.wipro.graphalgo;

import java.util.*;

public class MyNewGraph {
    private final int numOfNodes;
    private final List<MyEdge> edges;

    public MyNewGraph(int numOfNodes) {
        this.numOfNodes = numOfNodes;
        this.edges = new ArrayList<>();
    }

    public void addEdge(int source, int destination) {
        edges.add(new MyEdge(source, destination));
    }

    public List<MyEdge> getEdges() {
        return edges;
    }

    public int getNumOfNodes() {
        return numOfNodes;
    }
}
```




```java
package com.wipro.graphalgo;

public class MyUnionFind {
    private int[] parent;
    private int[] rank;
    public MyUnionFind(int size) {
        parent = new int[size];
        rank = new int[size];
        for (int i = 0; i < size; i++) {
            parent[i] = i;
            rank[i] = 0;}    }
public int find(int node) {
        if (parent[node] != node) {
            parent[node] = find(parent[node]); // Path compression
        }
        return parent[node]; }
public void union(int node1, int node2) {
        int root1 = find(node1);
        int root2 = find(node2);

        if (root1 != root2) {
            // Union by rank
            if (rank[root1] > rank[root2]) {
                parent[root2] = root1;
            } else if (rank[root1] < rank[root2]) {
                parent[root1] = root2;
            } else {
                parent[root2] = root1;
                rank[root1]++;
            }}}}
```

Output:


FS_JavaProgramming - Data_strutures/src/com/wipro/graphalgo/MyCycleDetection.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Data_strutures
JRE System Library [JavaSE-
src
com.ds.graph
com.ds.linkedlist
com.ds.patterns
com.ds.queue
com.ds.searching_algori
com.ds.sortingAlgorithr
com.ds.stack
com.ds.timeAndSpaceC
com.ds.tree
com.wipro.graphalgo
Dijkstra.java
FloydsAlgo.java
KrushkalAlgorithm.ja
KruskalsAlgorithm.jav
Main.java
MyCycleDetection.jav
MyEdge.java
MyGraph.java
MyNewGraph.java
MyUnionFind.java
Node.java
KrushkalAlgorithm.java
KruskalsAlgorithm.java
MyNewGraph.java
1 package com.wipro.graphalgo;
Console
<terminated> MyCycleDetection [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe
Graph contains cycle: true