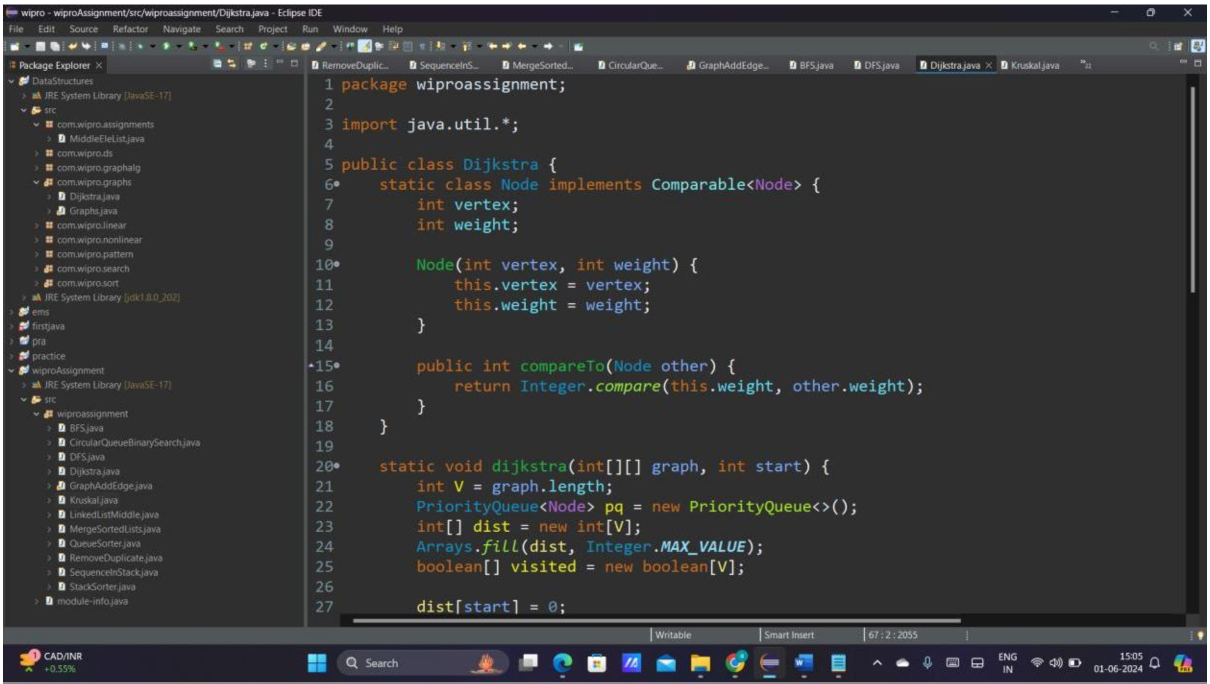
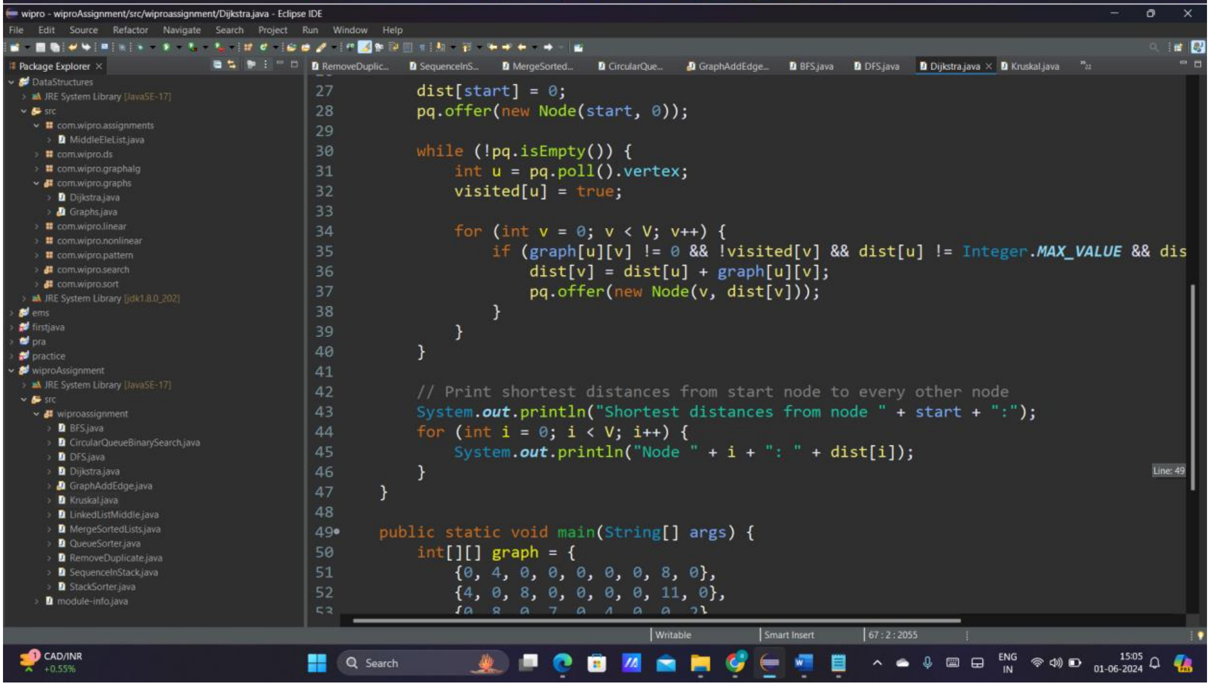
Assignment- Day 9 and 10

Task 1: Dijkstra's Shortest Path Finder

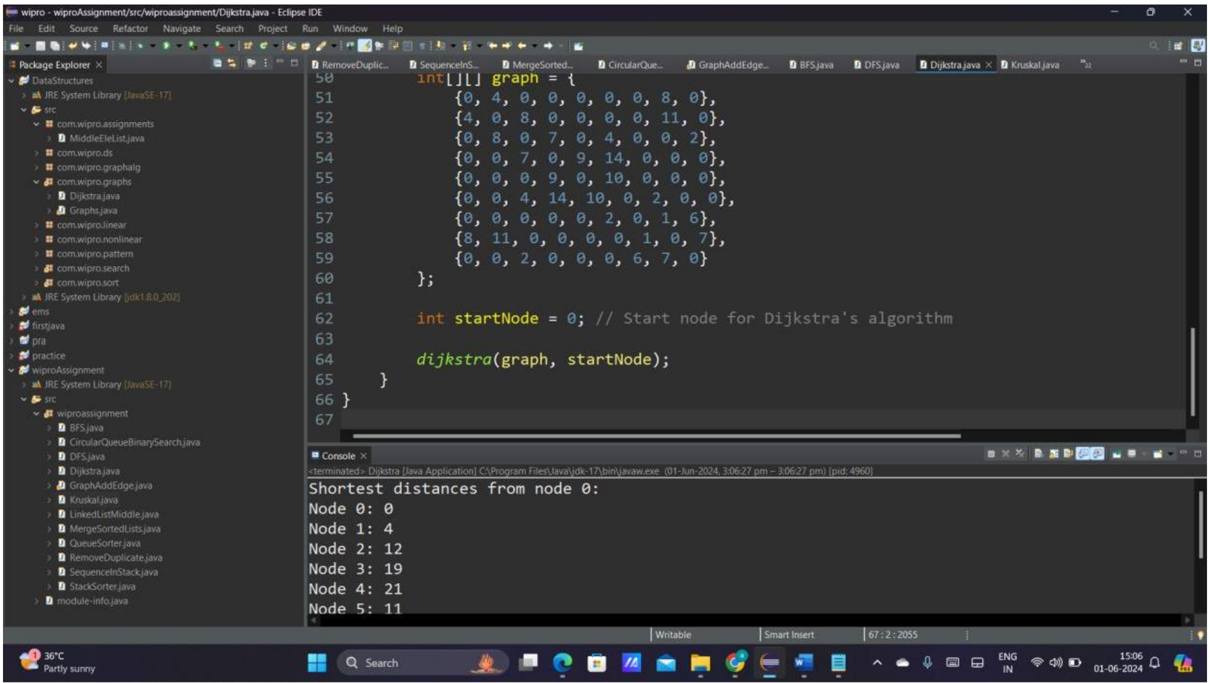
Code Dijkstra's algorithm to find the shortest path from a start node to every other node in aweighted graph with positive weights.



```java
package wiproassignment;

import java.util.*;

public class Dijkstra {
    static class Node implements Comparable<Node> {
        int vertex;
        int weight;

        Node(int vertex, int weight) {
            this.vertex = vertex;
            this.weight = weight;
        }

        public int compareTo(Node other) {
            return Integer.compare(this.weight, other.weight);
        }
    }

    static void dijkstra(int[][] graph, int start) {
        int V = graph.length;
        PriorityQueue<Node> pq = new PriorityQueue<>();
        int[] dist = new int[V];
        Arrays.fill(dist, Integer.MAX_VALUE);
        boolean[] visited = new boolean[V];

        dist[start] = 0;
```

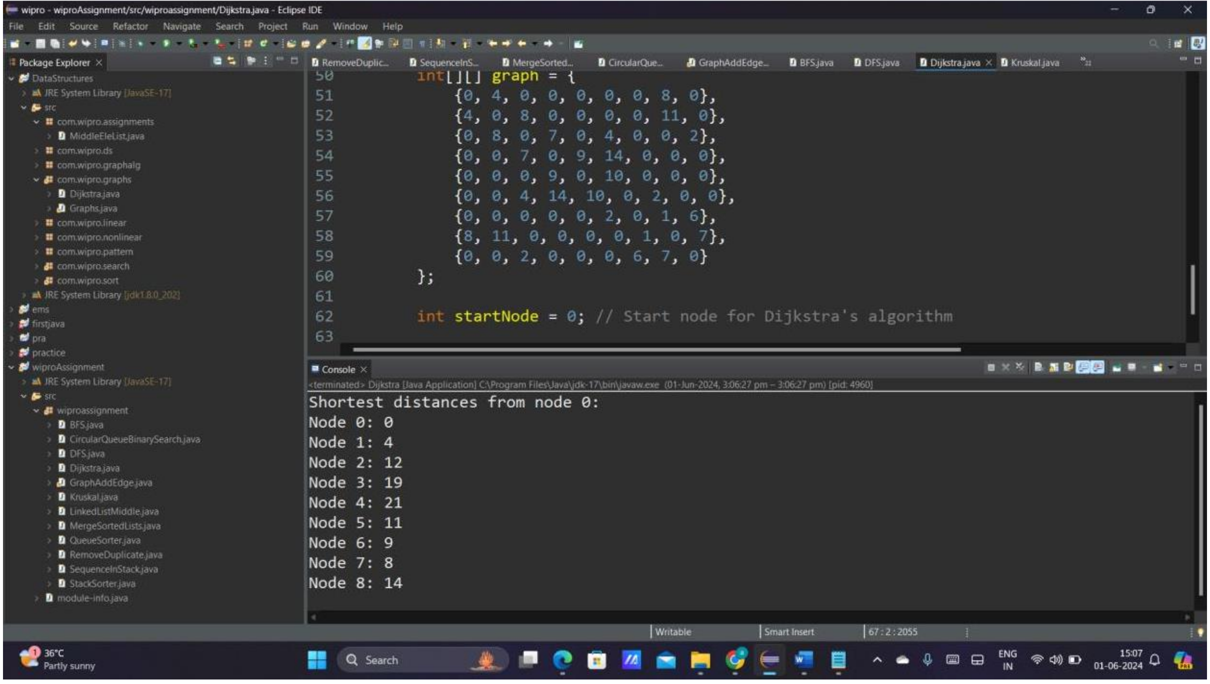


```java
        dist[start] = 0;
        pq.offer(new Node(start, 0));

        while (!pq.isEmpty()) {
            int u = pq.poll().vertex;
            visited[u] = true;

            for (int v = 0; v < V; v++) {
                if (graph[u][v] != 0 && !visited[v] && dist[u] != Integer.MAX_VALUE && dis
                    dist[v] = dist[u] + graph[u][v];
                    pq.offer(new Node(v, dist[v]));
                }
            }
        }

        // Print shortest distances from start node to every other node
        System.out.println("Shortest distances from node " + start + ":");
        for (int i = 0; i < V; i++) {
            System.out.println("Node " + i + ": " + dist[i]);
        }
    }

    public static void main(String[] args) {
        int[][] graph = {
            {0, 4, 0, 0, 0, 0, 0, 8, 0},
            {4, 0, 8, 0, 0, 0, 0, 11, 0},
```



```java
        int[][] graph = {
            {0, 4, 0, 0, 0, 0, 0, 8, 0},
            {4, 0, 8, 0, 0, 0, 0, 11, 0},
            {0, 8, 0, 7, 0, 4, 0, 0, 2},
            {0, 0, 7, 0, 9, 14, 0, 0, 0},
            {0, 0, 0, 9, 0, 10, 0, 0, 0},
            {0, 0, 4, 14, 10, 0, 2, 0, 0},
            {0, 0, 0, 0, 0, 2, 0, 1, 6},
            {8, 11, 0, 0, 0, 0, 1, 0, 7},
            {0, 0, 2, 0, 0, 0, 6, 7, 0}
        };

        int startNode = 0; // Start node for Dijkstra's algorithm

        dijkstra(graph, startNode);
    }
}
```

```
<terminated> Dijkstra [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (01-Jun-2024, 3:06:27 pm – 3:06:27 pm) [pid: 4960]
Shortest distances from node 0:
Node 0: 0
Node 1: 4
Node 2: 12
Node 3: 19
Node 4: 21
Node 5: 11
```

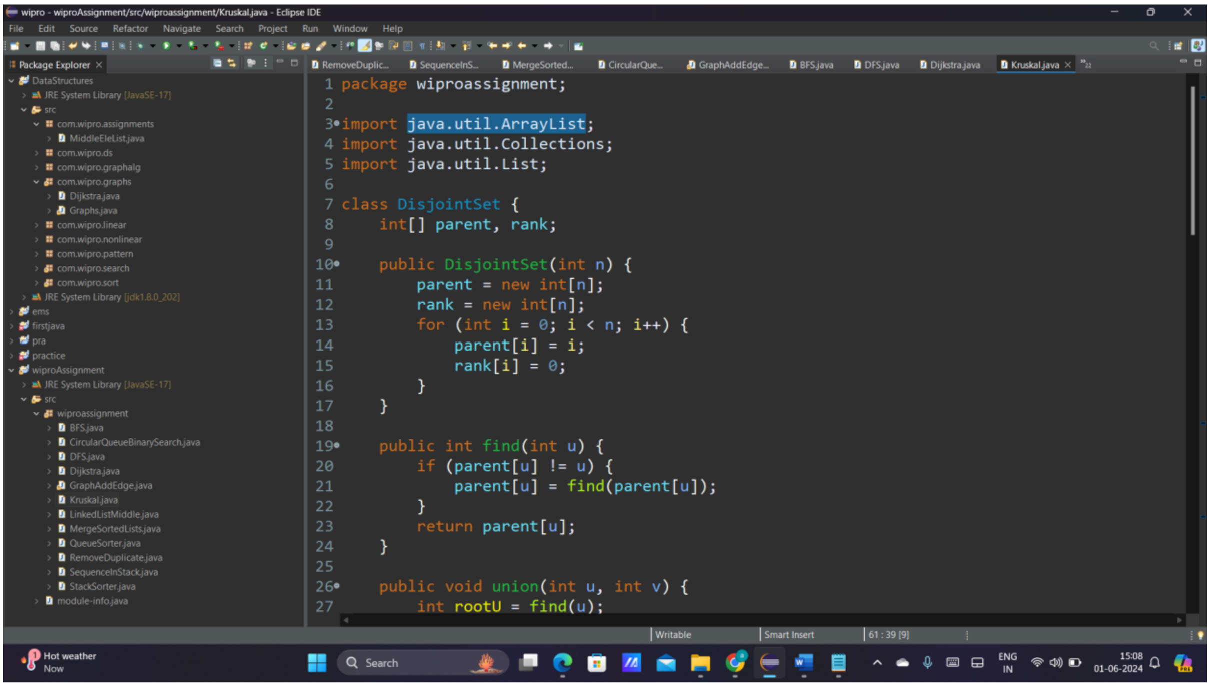

wipro - wiproAssignment/src/wiproassignment/Dijkstra.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Console
Writable
Smart Insert
67 : 2 : 2055
36°C
Partly sunny
Search
ENG
IN
15:07
01-06-2024

```java
        int[][] graph = {
            {0, 4, 0, 0, 0, 0, 0, 8, 0},
            {4, 0, 8, 0, 0, 0, 0, 11, 0},
            {0, 8, 0, 7, 0, 4, 0, 0, 2},
            {0, 0, 7, 0, 9, 14, 0, 0, 0},
            {0, 0, 0, 9, 0, 10, 0, 0, 0},
            {0, 0, 4, 14, 10, 0, 2, 0, 0},
            {0, 0, 0, 0, 0, 2, 0, 1, 6},
            {8, 11, 0, 0, 0, 0, 1, 0, 7},
            {0, 0, 2, 0, 0, 0, 6, 7, 0}
        };

        int startNode = 0; // Start node for Dijkstra's algorithm
```

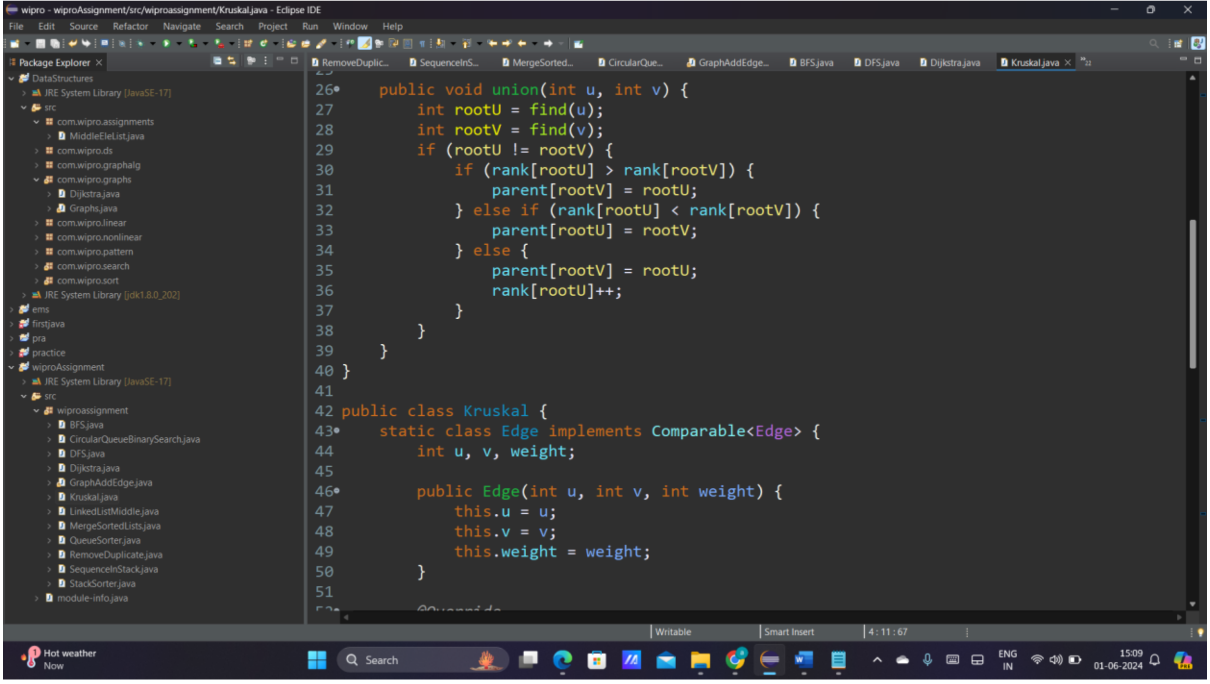
```
<terminated> Dijkstra [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (01-Jun-2024, 3:06:27 pm – 3:06:27 pm) [pid: 4960]
Shortest distances from node 0:
Node 0: 0
Node 1: 4
Node 2: 12
Node 3: 19
Node 4: 21
Node 5: 11
Node 6: 9
Node 7: 8
Node 8: 14
```

Task 2: Kruskal’s Algorithm for MST

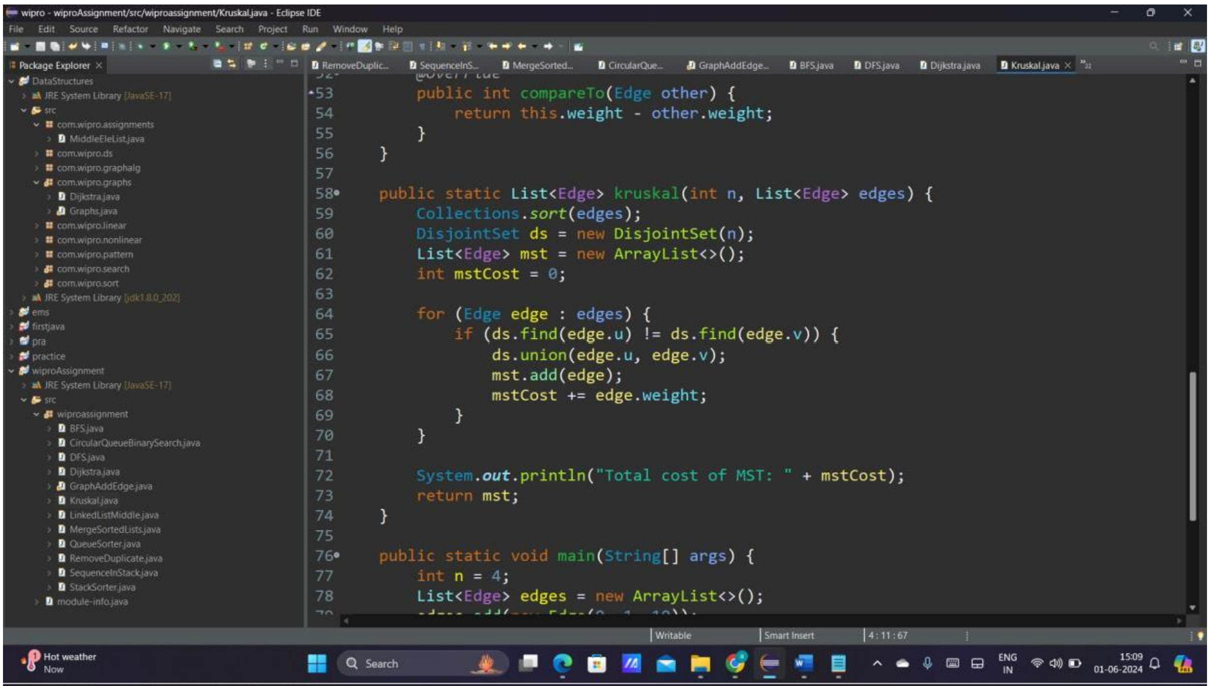
Implement Kruskal’s algorithm to find the minimum spanning tree of a given connected, undirectedgraph with non-negative edge weights.


wipro - wiproAssignment/src/wiproassignment/Kruskal.java - Eclipse IDE
Package Explorer
Kruskal.java
Writable
Smart Insert
61 : 39 [9]

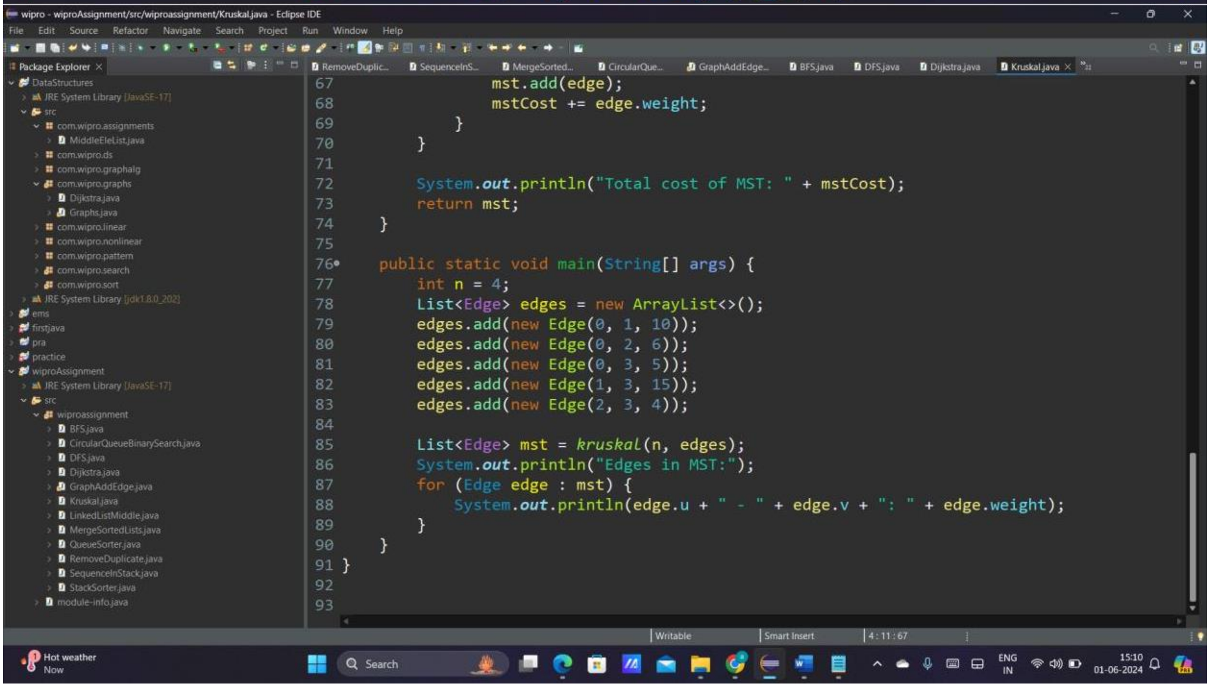
```
package wiproassignment;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

class DisjointSet {
    int[] parent, rank;

    public DisjointSet(int n) {
        parent = new int[n];
        rank = new int[n];
        for (int i = 0; i < n; i++) {
            parent[i] = i;
            rank[i] = 0;
        }
    }

    public int find(int u) {
        if (parent[u] != u) {
            parent[u] = find(parent[u]);
        }
        return parent[u];
    }

    public void union(int u, int v) {
        int rootU = find(u);
```


wipro - wiproAssignment/src/wiproassignment/Kruskal.java - Eclipse IDE
Package Explorer
Kruskal.java
Writable
Smart Insert
4 : 11 : 67

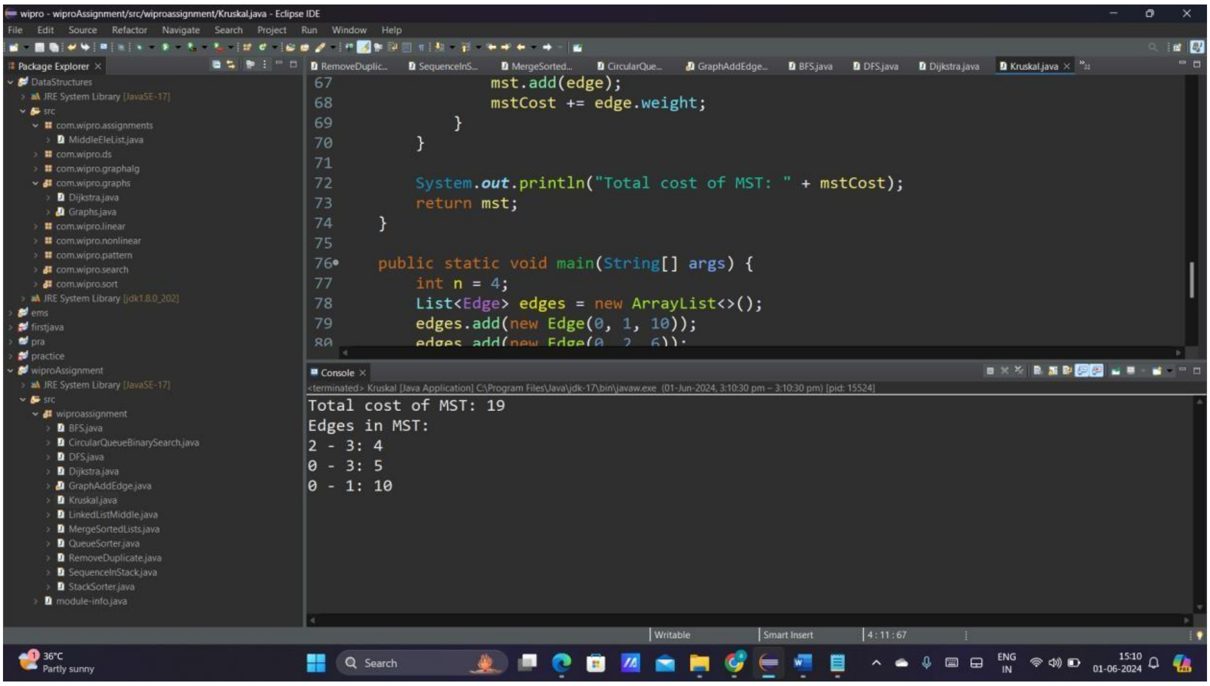
```
    public void union(int u, int v) {
        int rootU = find(u);
        int rootV = find(v);
        if (rootU != rootV) {
            if (rank[rootU] > rank[rootV]) {
                parent[rootV] = rootU;
            } else if (rank[rootU] < rank[rootV]) {
                parent[rootU] = rootV;
            } else {
                parent[rootV] = rootU;
                rank[rootU]++;
            }
        }
    }
}

public class Kruskal {
    static class Edge implements Comparable<Edge> {
        int u, v, weight;

        public Edge(int u, int v, int weight) {
            this.u = u;
            this.v = v;
            this.weight = weight;
        }

        @Override
```


wipro - wiproAssignment/src/wiproassignment/Kruskal.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
RemoveDuplic... SequenceInS... MergeSorted... CircularQue... GraphAddEdge... BFS.java DFS.java Dijkstra.java Kruskal.java
Writable
Smart Insert
4 : 11 : 67
Hot weather
Now
Search
ENG IN
15:09
01-06-2024

```
        @Override
        public int compareTo(Edge other) {
            return this.weight - other.weight;
        }
    }

    public static List<Edge> kruskal(int n, List<Edge> edges) {
        Collections.sort(edges);
        DisjointSet ds = new DisjointSet(n);
        List<Edge> mst = new ArrayList<>();
        int mstCost = 0;

        for (Edge edge : edges) {
            if (ds.find(edge.u) != ds.find(edge.v)) {
                ds.union(edge.u, edge.v);
                mst.add(edge);
                mstCost += edge.weight;
            }
        }

        System.out.println("Total cost of MST: " + mstCost);
        return mst;
    }

    public static void main(String[] args) {
        int n = 4;
        List<Edge> edges = new ArrayList<>();
        edges.add(new Edge(0, 1, 10));
```


wipro - wiproAssignment/src/wiproassignment/Kruskal.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
RemoveDuplic... SequenceInS... MergeSorted... CircularQue... GraphAddEdge... BFS.java DFS.java Dijkstra.java Kruskal.java
Writable
Smart Insert
4 : 11 : 67
Hot weather
Now
Search
ENG IN
15:10
01-06-2024

```
                mst.add(edge);
                mstCost += edge.weight;
            }
        }

        System.out.println("Total cost of MST: " + mstCost);
        return mst;
    }

    public static void main(String[] args) {
        int n = 4;
        List<Edge> edges = new ArrayList<>();
        edges.add(new Edge(0, 1, 10));
        edges.add(new Edge(0, 2, 6));
        edges.add(new Edge(0, 3, 5));
        edges.add(new Edge(1, 3, 15));
        edges.add(new Edge(2, 3, 4));

        List<Edge> mst = kruskal(n, edges);
        System.out.println("Edges in MST:");
        for (Edge edge : mst) {
            System.out.println(edge.u + " - " + edge.v + ": " + edge.weight);
        }
    }
}


```

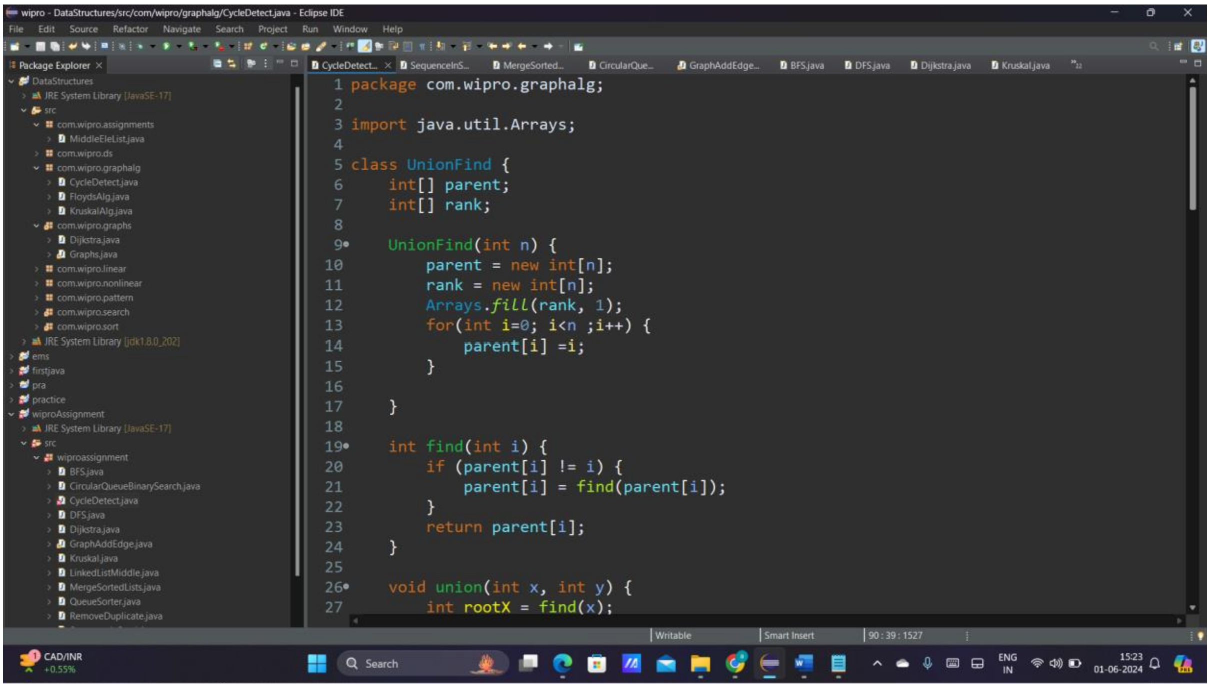

wipro - wiproAssignment/src/wiproassignment/Kruskal.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Kruskal.java
Console
Writable
Smart Insert
4 : 11 : 67

```java
                mst.add(edge);
                mstCost += edge.weight;
            }
        }

        System.out.println("Total cost of MST: " + mstCost);
        return mst;
    }

    public static void main(String[] args) {
        int n = 4;
        List<Edge> edges = new ArrayList<>();
        edges.add(new Edge(0, 1, 10));
        edges.add(new Edge(0, 2, 6));
```

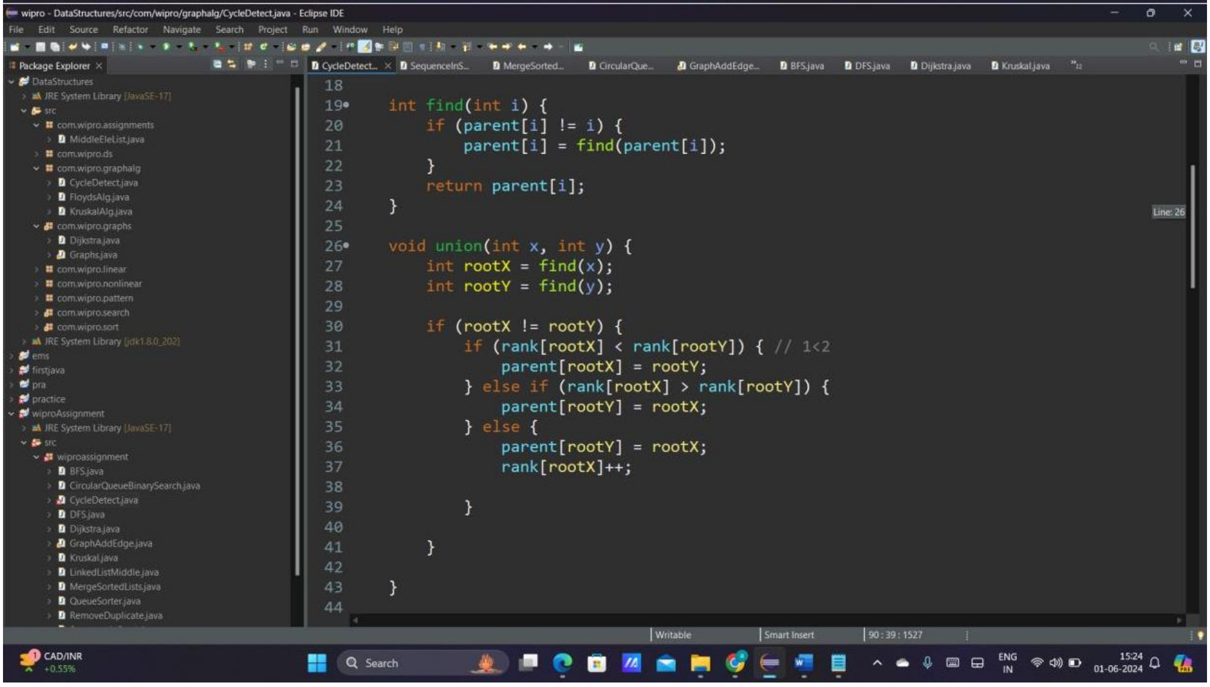
```
<terminated> Kruskal [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (01-Jun-2024, 3:10:30 pm – 3:10:30 pm) [pid: 15524]
Total cost of MST: 19
Edges in MST:
2 - 3: 4
0 - 3: 5
0 - 1: 10
```

Task 3: Union-Find for Cycle Detection

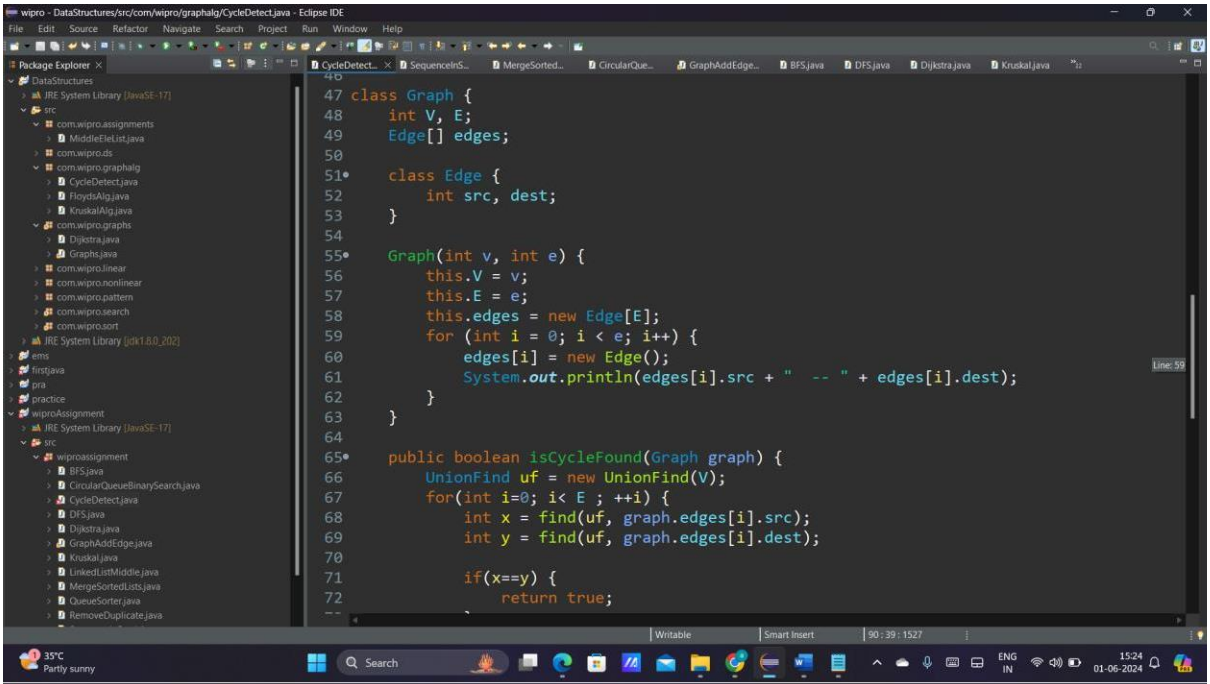
Write a Union-Find data structure with path compression. Use this data structure to detect a cycle inan undirected graph.



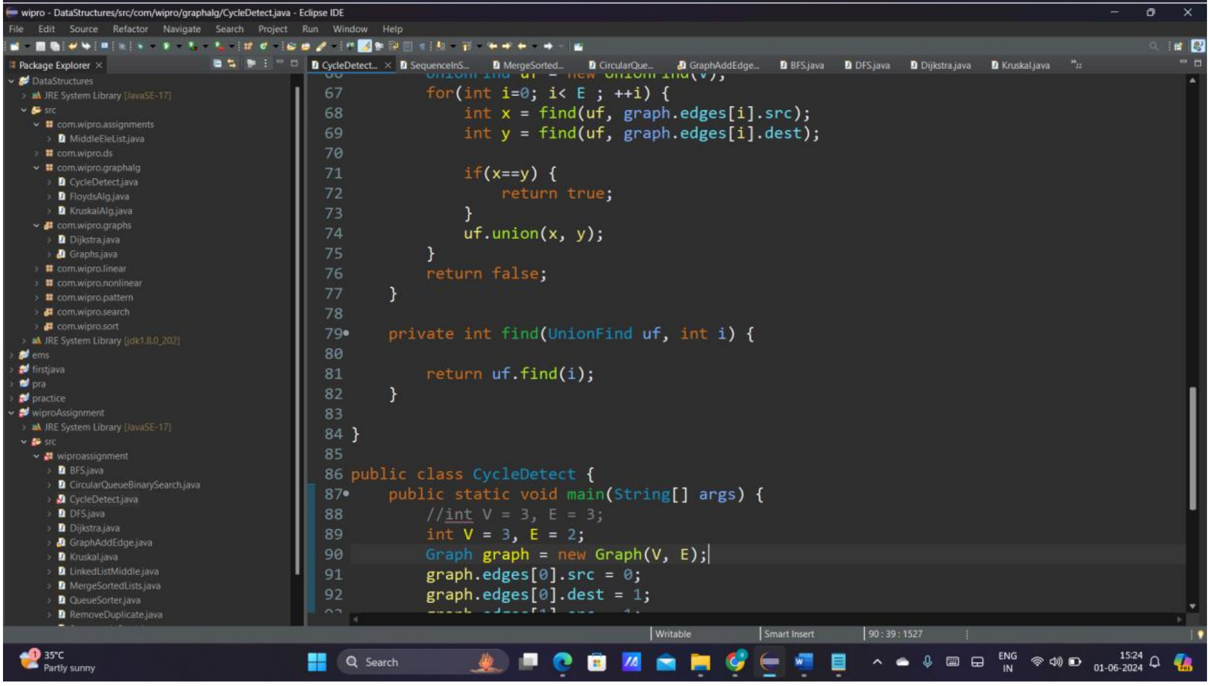
```java
package com.wipro.graphalg;

import java.util.Arrays;

class UnionFind {
    int[] parent;
    int[] rank;

    UnionFind(int n) {
        parent = new int[n];
        rank = new int[n];
        Arrays.fill(rank, 1);
        for(int i=0; i<n ;i++) {
            parent[i] =i;
        }

    }

    int find(int i) {
        if (parent[i] != i) {
            parent[i] = find(parent[i]);
        }
        return parent[i];
    }

    void union(int x, int y) {
        int rootX = find(x);
```



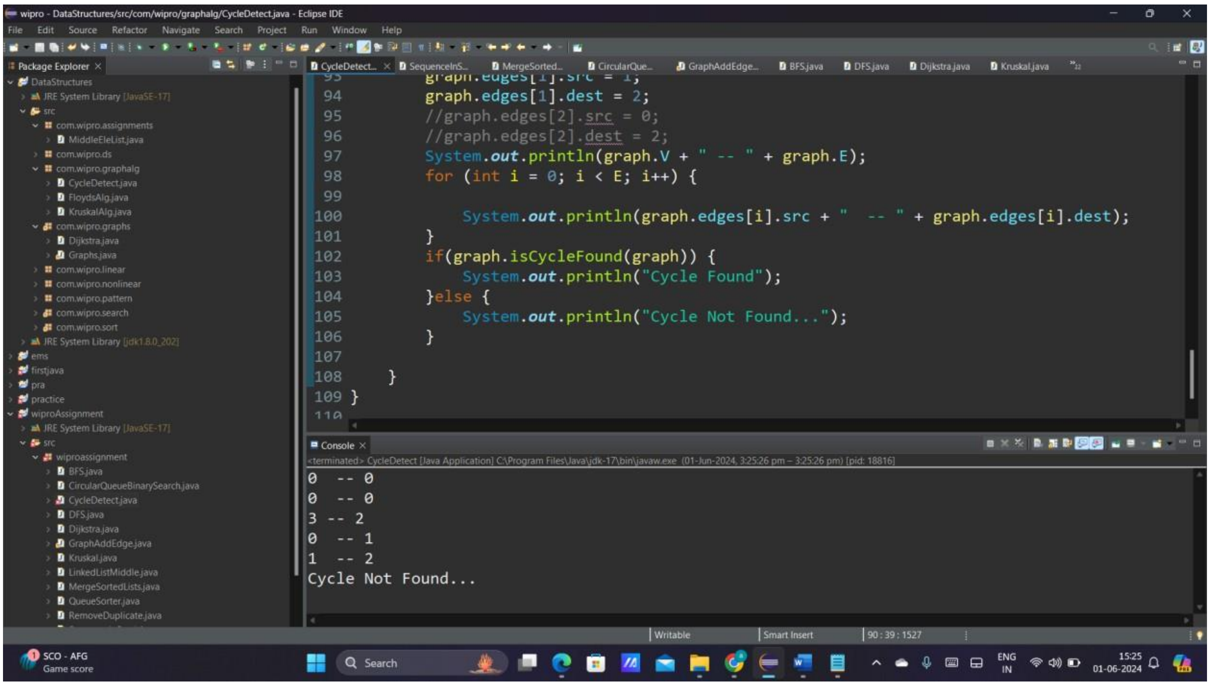
```java

    int find(int i) {
        if (parent[i] != i) {
            parent[i] = find(parent[i]);
        }
        return parent[i];
    }

    void union(int x, int y) {
        int rootX = find(x);
        int rootY = find(y);

        if (rootX != rootY) {
            if (rank[rootX] < rank[rootY]) { // 1<2
                parent[rootX] = rootY;
            } else if (rank[rootX] > rank[rootY]) {
                parent[rootY] = rootX;
            } else {
                parent[rootY] = rootX;
                rank[rootX]++;

            }

        }

    }

```



```java
class Graph {
    int V, E;
    Edge[] edges;

    class Edge {
        int src, dest;
    }

    Graph(int v, int e) {
        this.V = v;
        this.E = e;
        this.edges = new Edge[E];
        for (int i = 0; i < e; i++) {
            edges[i] = new Edge();
            System.out.println(edges[i].src + "  -- " + edges[i].dest);
        }
    }

    public boolean isCycleFound(Graph graph) {
        UnionFind uf = new UnionFind(V);
        for(int i=0; i< E ; ++i) {
            int x = find(uf, graph.edges[i].src);
            int y = find(uf, graph.edges[i].dest);

            if(x==y) {
                return true;
```


wipro - DataStructures/src/com/wipro/graphalg/CycleDetect.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
CycleDetect...
Writable
Smart Insert
90 : 39 : 1527
35°C
Partly sunny
Search
ENG
IN
15:24
01-06-2024

```java
        for(int i=0; i< E ; ++i) {
            int x = find(uf, graph.edges[i].src);
            int y = find(uf, graph.edges[i].dest);

            if(x==y) {
                return true;
            }
            uf.union(x, y);
        }
        return false;
    }

    private int find(UnionFind uf, int i) {

        return uf.find(i);
    }

}

public class CycleDetect {
    public static void main(String[] args) {
        //int V = 3, E = 3;
        int V = 3, E = 2;
        Graph graph = new Graph(V, E);
        graph.edges[0].src = 0;
        graph.edges[0].dest = 1;
```



```java
        graph.edges[1].src = 1;
        graph.edges[1].dest = 2;
        //graph.edges[2].src = 0;
        //graph.edges[2].dest = 2;
        System.out.println(graph.V + " -- " + graph.E);
        for (int i = 0; i < E; i++) {

            System.out.println(graph.edges[i].src + "  -- " + graph.edges[i].dest);
        }
        if(graph.isCycleFound(graph)) {
            System.out.println("Cycle Found");
        }else {
            System.out.println("Cycle Not Found...");
        }

    }
}
```

<terminated> CycleDetect [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (01-Jun-2024, 3:25:26 pm – 3:25:26 pm) [pid: 18816]

```
0  -- 0
0  -- 0
3 -- 2
0  -- 1
1  -- 2
Cycle Not Found...
```