Symbiosis Institute of Technology, Pune

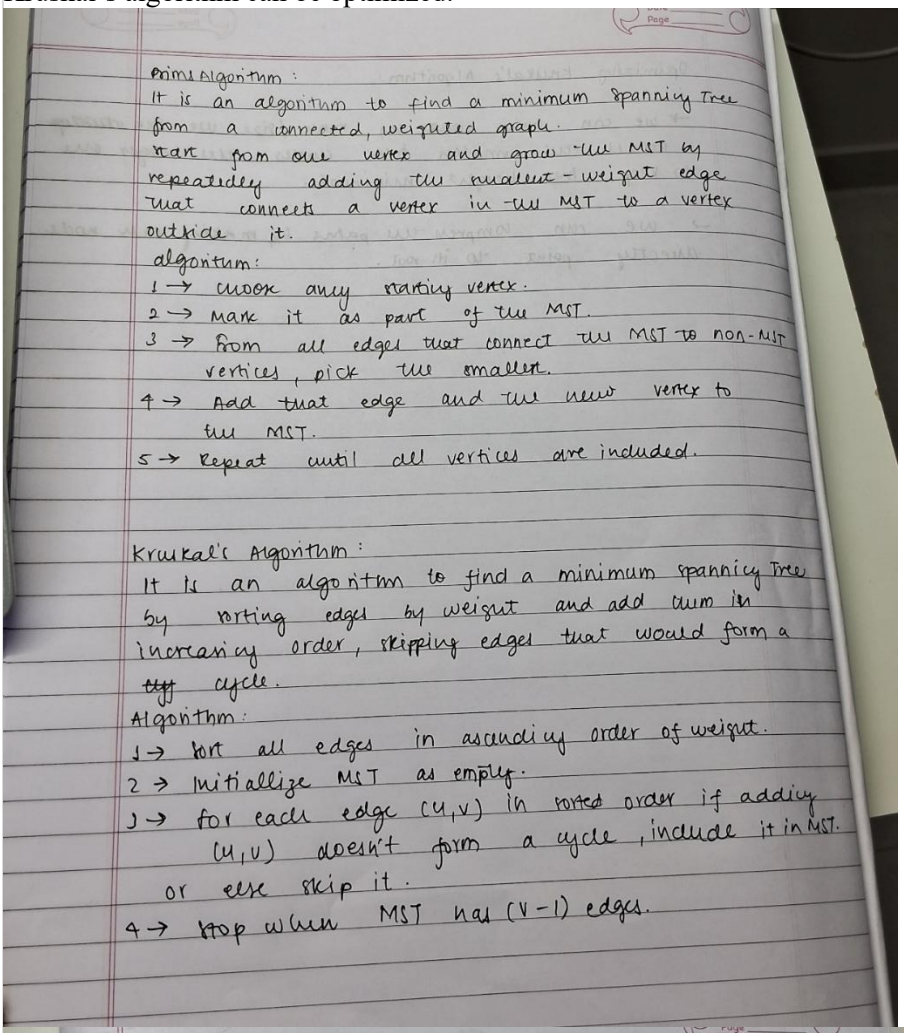
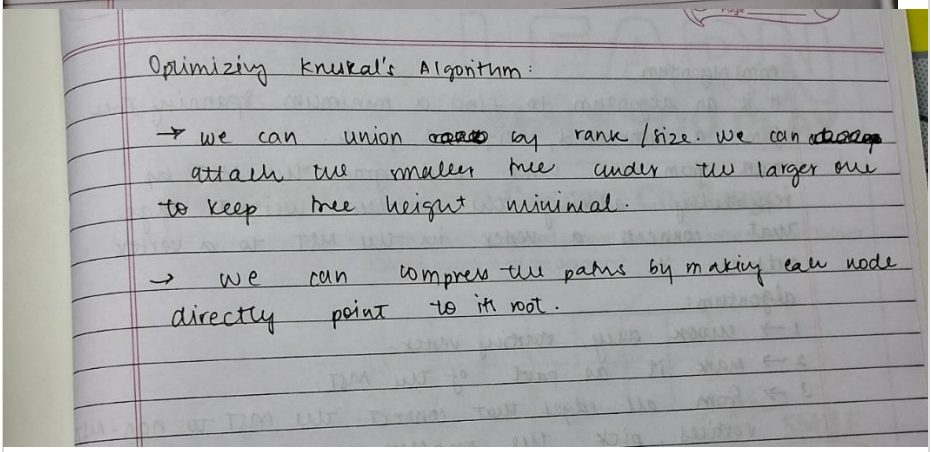
Department of Computer Science and Engineering

Academic Year 2025-26

Design and Analysis of Algorithms– Lab

Batch 2023-27 - Sem V

Lab Assignment No:- 1	
Name of Student	Deepti Pal
PRN No.	23070122081
Batch	CSE
Class	A3
Academic Year & Semester	2023-27 , TY, 5th Sem
Date of Submission	11 Aug 2025
Title of Assignment:	Create implementations for the Kruskal's & Prim's Algorithm for finding the Minimum Cost Spanning Tree for a given graph. You may input the graph from the file and use an adjacency matrix. **Scenario:** A university campus **is planning to connect all its departments with** network cables **to enable high-speed internet access. The goal is to** minimize the total cost **of laying down the cables while ensuring every department is connected.** **Your Task:** Design a program that: • Reads a graph (departments as vertices and cost of laying cables as edge weights) from a file • Implements **Prim's Algorithm** to find the **Minimum Cost Spanning Tree (MCST)**. • Tracks the number of **edge comparisons**. • Displays each stage of node inclusion and cumulative cost.

Theory: (Handwritten)	Explain Prim's and Kruskal's algorithms in detail. Discuss in detail how Kruskal's algorithm can be optimized.  

Source code (Implementation Screenshot)	import java.io.*;
	import java.util.*;
	
	public class Main {
	Scanner scan = new Scanner(System.in);
	public static void main(String[] args) {
	
	//read the matrix:
	try {

```
                        File file = new File("clg_network_cost.txt");
                        Scanner scan = new Scanner(file);
                        int v = scan.nextInt();
                        int[][] graph = new int[v][v];

                        for(int i=0; i<v; i++) {
                                for(int j=0; j<v; j++) {
                                        graph[i][j] = scan.nextInt();
                                }
                        }
                        scan.close();


                        System.out.println("Prim's Algorithm Output : ");
                        prims(graph, v);

                        System.out.println("\nKruskal's Algorithm Output :
");
                        kruskal(graph, v);
                } catch (FileNotFoundException e) {
                        System.out.println("Graph file not found!");
                }

        }

        //prims algorithm
        public static void prims(int[][] graph, int v) {
                int[] key = new int[v];
                boolean[] mstSet = new boolean[v];
                int[] parent = new int[v];

                //initailize
                Arrays.fill(key, Integer.MAX_VALUE);
                key[0] = 0;
                parent[0] = -1;
                int cumCost = 0;
                int edgeCompares = 0;

                for(int count = 0; count<v-1; count ++) {
                        int u = minkey(key, mstSet, v);
                        mstSet[u] = true;

                        if (parent[u] != -1) {
                                cumCost += graph[u][parent[u]];
                                System.out.println("Edge: (" + parent[u] + " -
" + u +
                                                ") | Cost: " + graph[u][parent[u]]
+
                                                " | Cumulative Cost: " +
cumCost);
                        }

                        for(int i=0; i<v; i++) {
                                edgeCompares++;
                                if(graph[u][i] != 0 && !mstSet[i] &&
```

```
graph[u][i]<key[i]) {
                                        parent[i] = u;
                                        key[i] = graph[u][i];
                                }
                        }
                }

                System.out.println("Total Cost of MST: " + cumCost);
                System.out.println("Total Edge Comparisons: " +
edgeCompares);

        }

        public static int minkey(int[] key, boolean[] mstSet, int n) {
                int min = Integer.MAX_VALUE;
                int minIndex = -1;
                for(int v=0; v<n; v++) {
                        if(!mstSet[v] && key[v]<min) {
                                min = key[v];
                                minIndex = v;
                        }
                }
                return minIndex;
        }

        //kruskal-----------
        static class Edge implements Comparable<Edge> {
                int src, dest, weight;
                public int compareTo(Edge other) {
                        return this.weight - other.weight;
                }
        }

        static class Subset {
                int parent, rank;
        }

        static void kruskal(int[][] graph, int v) {
                List<Edge> edges = new ArrayList<>();
                int edgeCompares = 0;
                int cumCost = 0;

                for (int i = 0; i < v; i++) {
                        for (int j = i + 1; j < v; j++) {
                                if (graph[i][j] != 0) {
                                        Edge e = new Edge();
                                        e.src = i;
                                        e.dest = j;
                                        e.weight = graph[i][j];
                                        edges.add(e);
                                }
                        }
                }

                Collections.sort(edges);
```

```
		Subset[] subsets = new Subset[v];

		for (int i = 0; i < v; i++) {
			subsets[i] = new Subset();
			subsets[i].parent = i;
			subsets[i].rank = 0;
		}

		int e = 0;
		int i = 0;
		while (e < v-1 && i < edges.size()) {
			Edge nextEdge = edges.get(i++);
			edgeCompares++;

			int x = find(subsets, nextEdge.src);
			int y = find(subsets, nextEdge.dest);

			if (x != y) {
				e++;
				cumCost += nextEdge.weight;
				System.out.println("Included Edge: (" +
nextEdge.src + " - " + nextEdge.dest +
					") | Cost: " + nextEdge.weight +
					" | Cumulative Cost: " +
cumCost);
				union(subsets, x, y);
			}
		}

		System.out.println("Total Cost of MST: " + cumCost);
		System.out.println("Total Edge Comparisons: " +
edgeCompares);
	}

	static int find(Subset[] subsets, int i) {
		if (subsets[i].parent != i)
			subsets[i].parent = find(subsets, subsets[i].parent);
		return subsets[i].parent;
	}

	static void union(Subset[] subsets, int x, int y) {
		int xroot = find(subsets, x);
		int yroot = find(subsets, y);

		if (subsets[xroot].rank < subsets[yroot].rank)
			subsets[xroot].parent = yroot;
		else if (subsets[xroot].rank > subsets[yroot].rank)
			subsets[yroot].parent = xroot;
		else {
			subsets[yroot].parent = xroot;
			subsets[xroot].rank++;
		}
	}
}
```

Output Screenshots

Submit Before: 8/11/2025, 1:59:00 PM

al's & Prim's Algorithm for finding the Minimum

and use an adjacency matrix.

nning to connect all its departments with

internet access. The goal is to minimize the

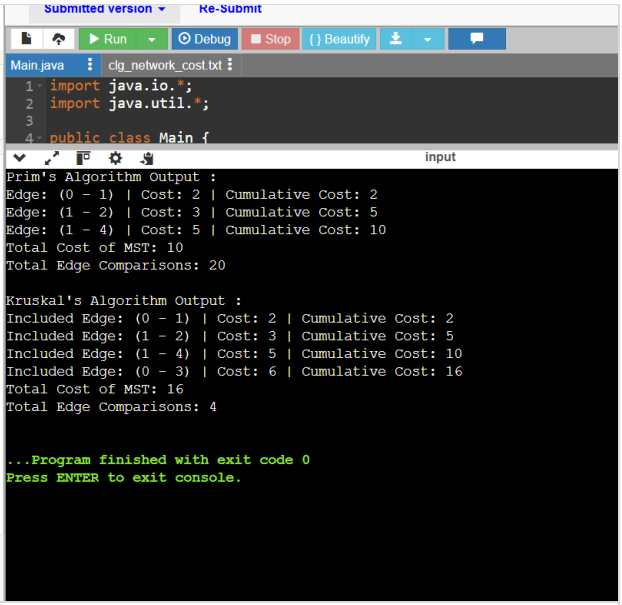
hile ensuring every department is connected.

as vertices and cost of laying cables as edge

to find the **Minimum Cost Spanning Tree**

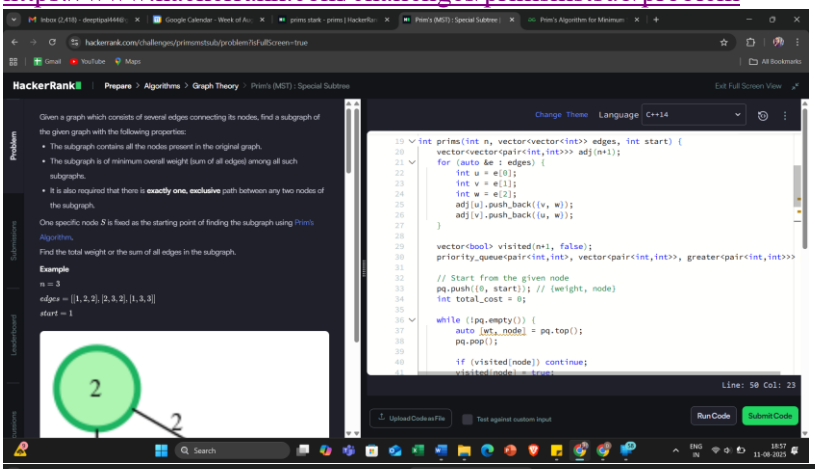
omparisons.

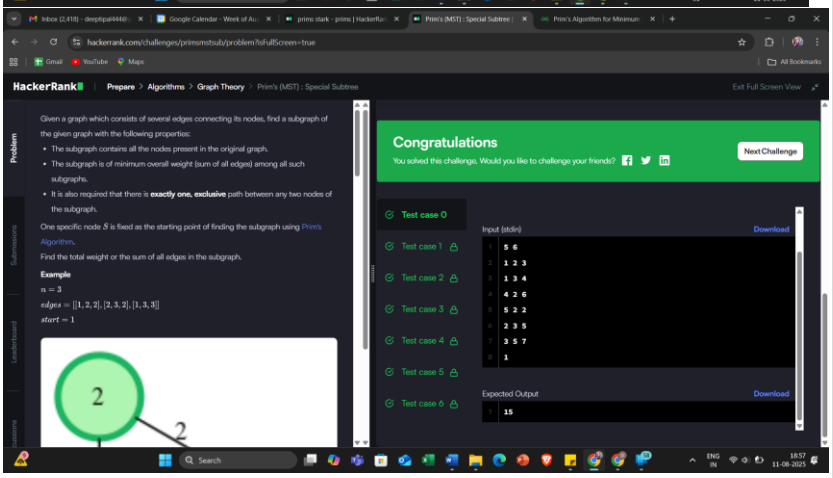
and cumulative cost.



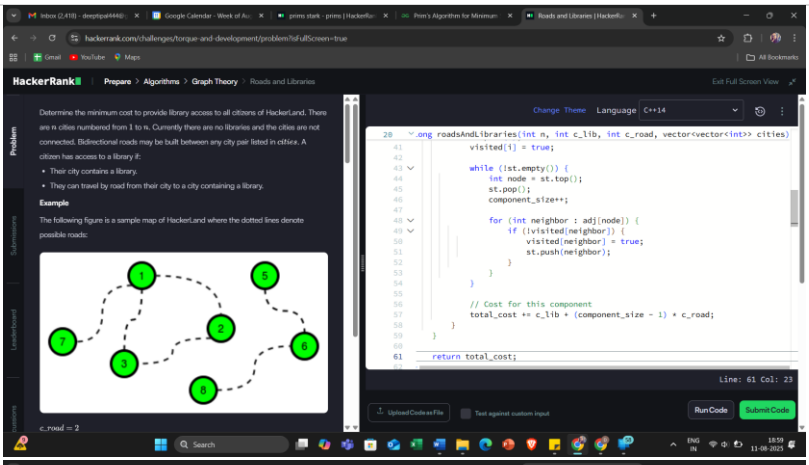
Problems Solved from Hacker Rank (Minimum 4)

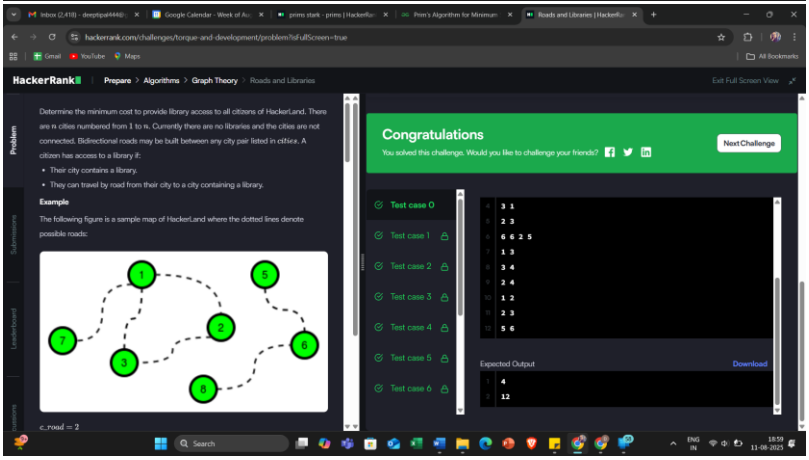
1. https://www.hackerrank.com/challenges/primsmstsub/problem



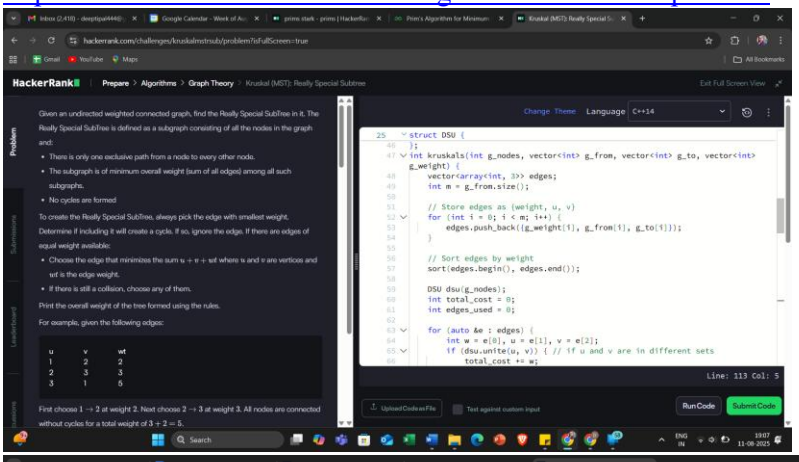


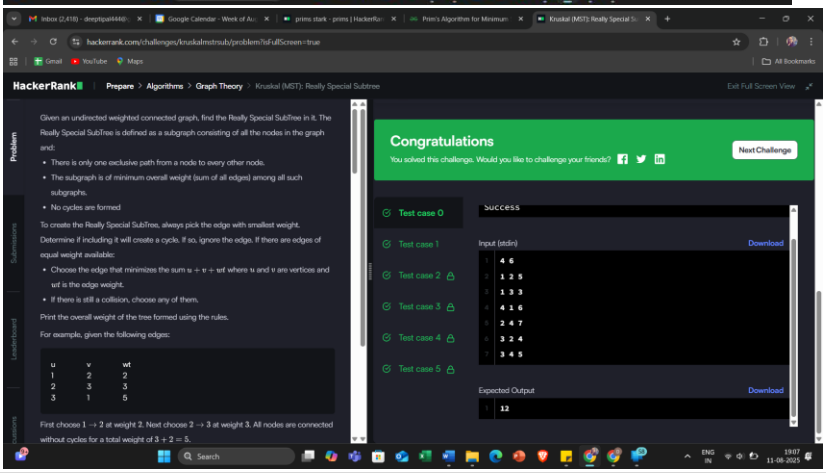
2. https://www.hackerrank.com/challenges/torque-and-development/problem





3. https://www.hackerrank.com/challenges/kruskalmstrsub/problem





	4. https://www.hackerrank.com/contests/vit-bhopal/challenges/prims-algorithm-2 - **locked**