Lab 8 Design and Analysis of Algorithms

Name: Tanvi Penumudy

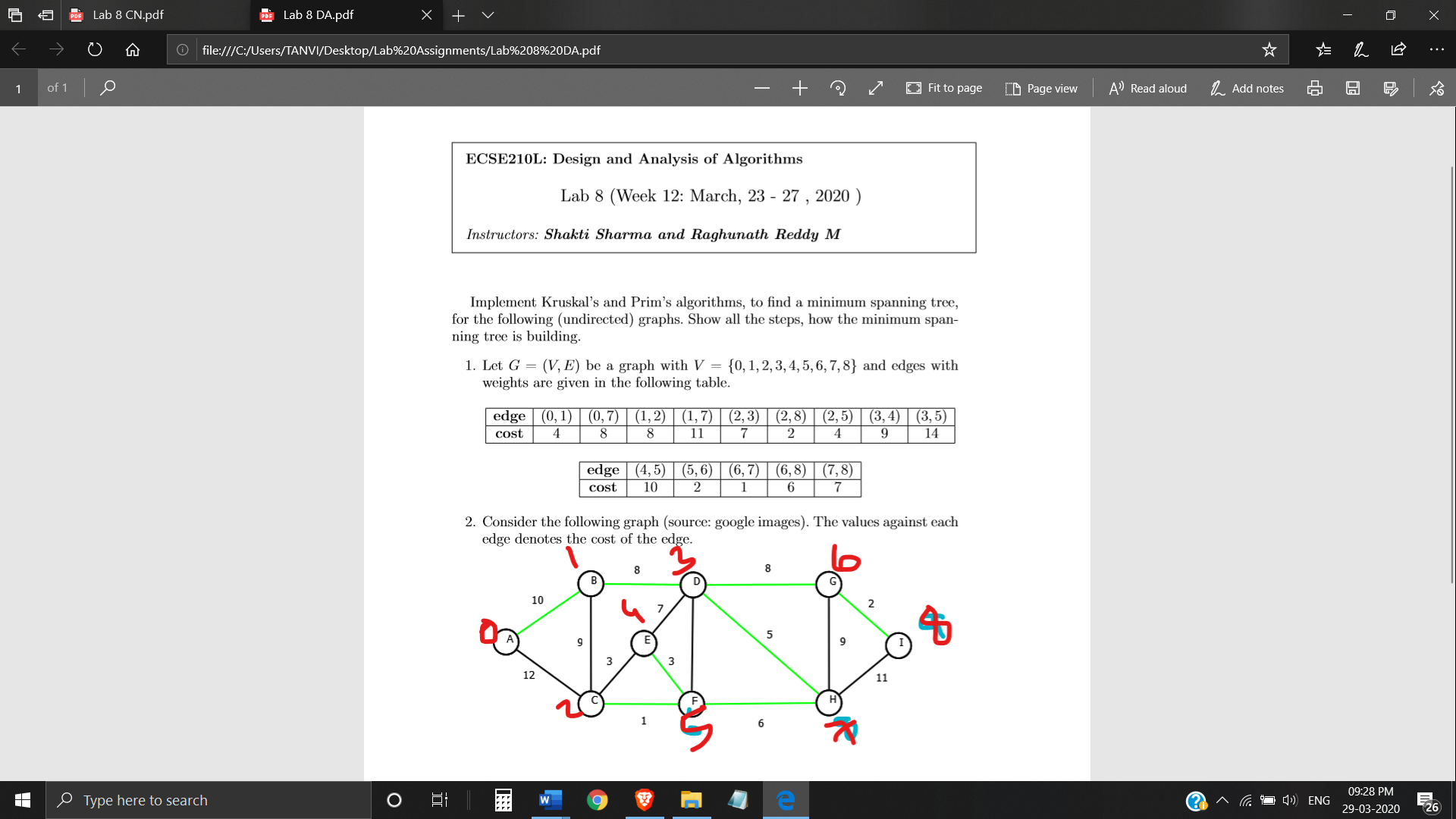
Enroll No: E18CSE187

Batch: EB06

**Contents:**

* Prim’s Ques
* Kruskal’s Ques

**Prim’s**

****

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class Main

{

private static final int V=5;

int minKey(int key[],Boolean mstSet[])

{

int min=Integer.MAX\_VALUE,min\_index=-1;

for(int v=0;v<V;v++)

if(mstSet[v]==false && key[v]<min)

{

min=key[v]; min\_index=v;

}

return min\_index;

}

void printMST(int parent[],int n,int graph[][])

{

System.out.println("Edge\tWeight");

for(int i=1;i<V;i++) System.out.println(parent[i]+"-"+i+"\t"+graph[i][parent[i]]);

}

void primMST(int graph[][])

{

int parent[]=new int[V];

int key[]=new int[V];

Boolean mstSet[]=new Boolean[V];

for(int i=0;i<V;i++)

{

key[i]=Integer.MAX\_VALUE; mstSet[i]=false;

}

key[0]=0; parent[0]=-1;

for(int count=0;count<V-1;count++)

{

int u=minKey(key,mstSet);

mstSet[u]=true;

for(int v=0;v<V;v++)

if(graph[u][v]!=0 && mstSet[v]==false && graph[u][v]<key[v])

{

parent[v]=u;

key[v]=graph[u][v];

}

}

printMST(parent,V,graph);

}

public static void main(String[]args)

{

System.out.println("0->A");

System.out.println("1->B");

System.out.println("2->C");

System.out.println("3->D");

System.out.println("4->E");

System.out.println("5->F");

System.out.println("6->G");

System.out.println("7->H");

System.out.println("8->I");

System.out.println("9->J");

Main t=new Main();

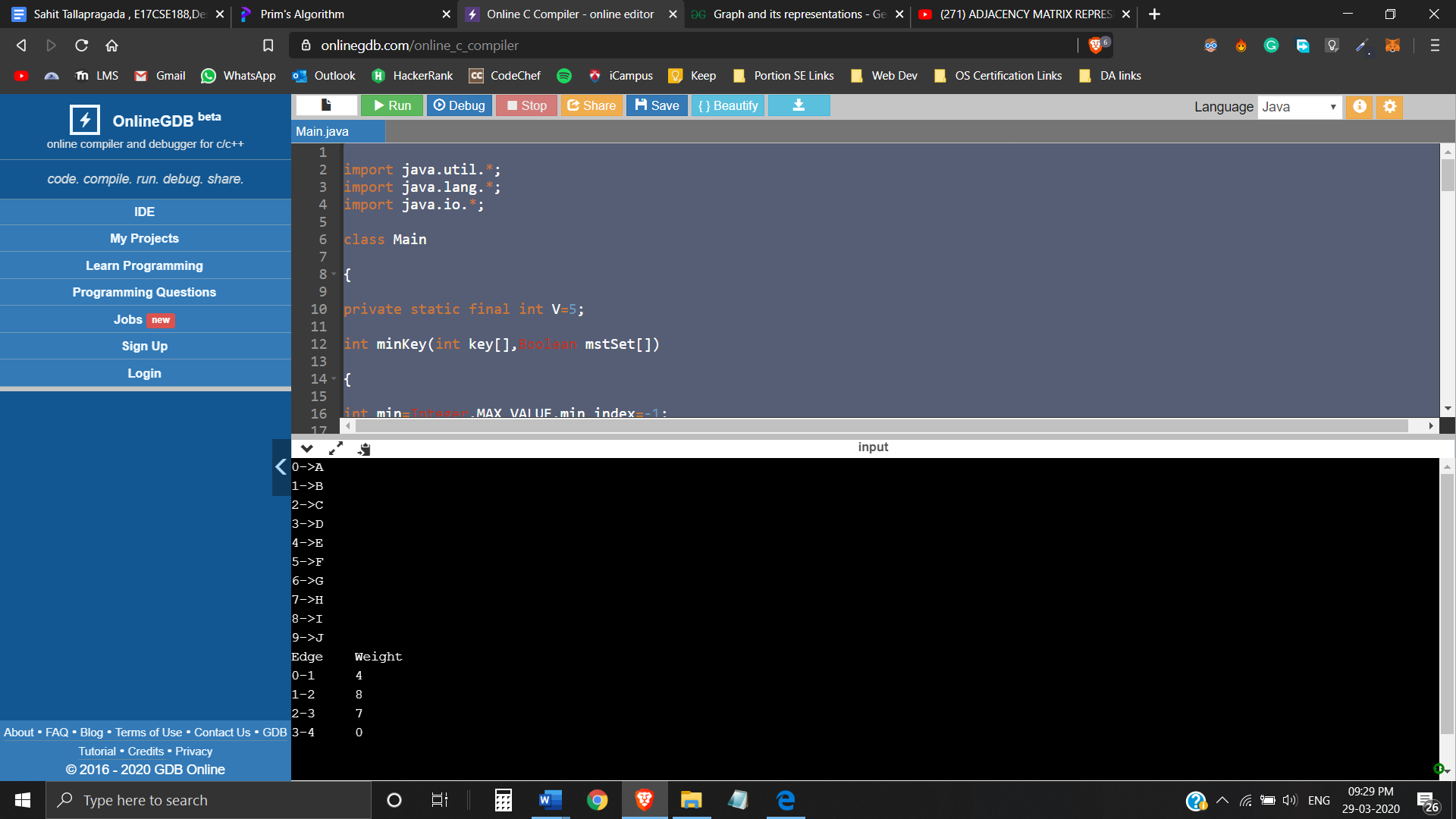
int graph[][]=new int[][]{{0,4,0,0,0,0,0,8,0},{4,0,8,0,0,0,0,11,0},{0,8,0,7,0,4,0,0,2},{0,0,7,0,9,14,0,0,2},{0,0,7,0,9,14,0,0,0},{0,0,0,9,0,10,0,0,0},{0,0,4,14,10,0,2,0,0},{0,0,0,0,0,2,0,1,6},{8,11,0,0,0,0,1,0,7},{0,0,2,0,0,0,6,7,0}};

/\* adjacency matrix 9\*9 \*/

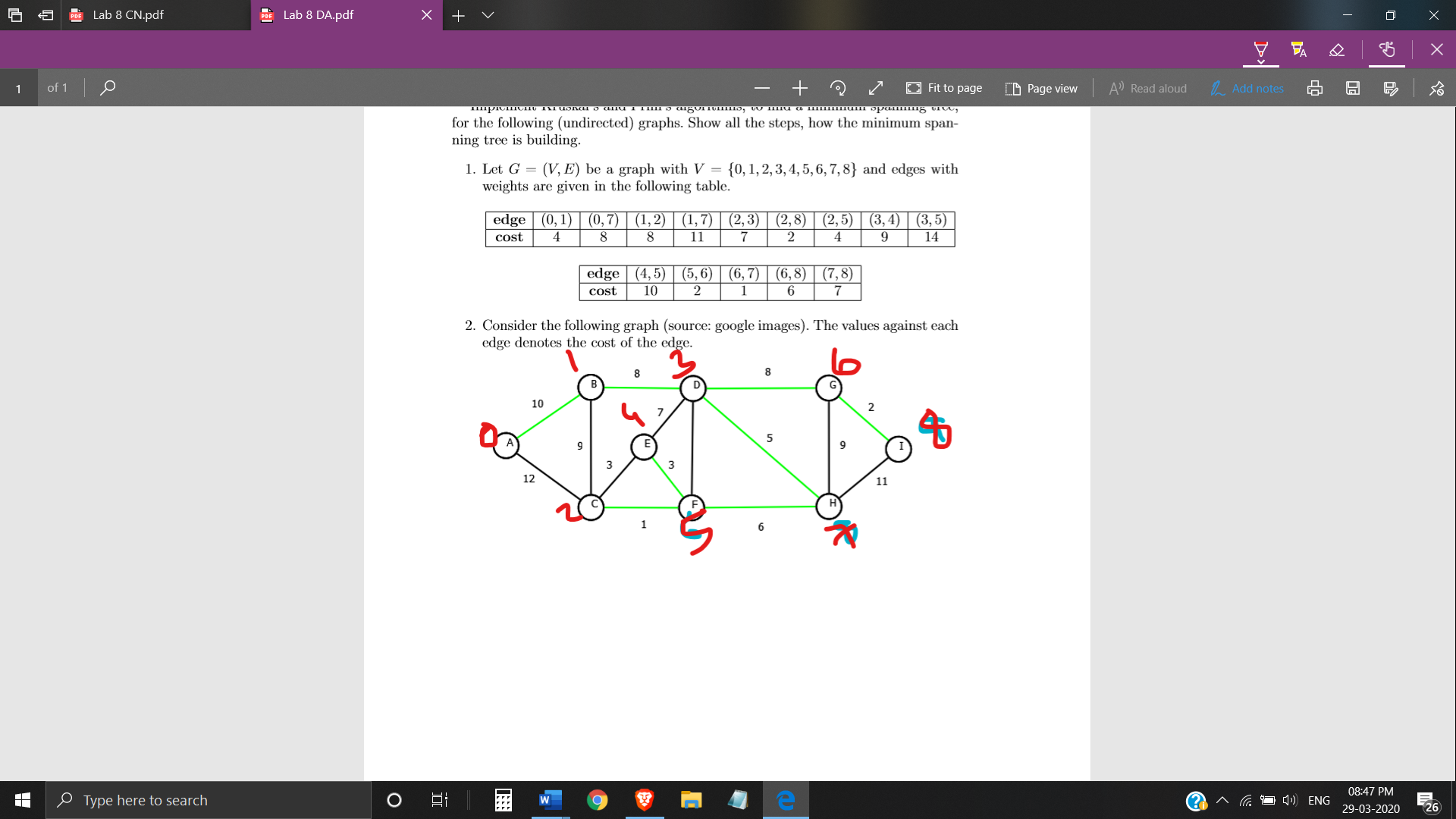
t.primMST(graph);

}

}

* 

**Kruskal’s**

****

#include<bits/stdc++.h>

using namespace std;

typedef pair<int, int> iPair;

struct Graph

{

  int V, E;

  vector< pair<int, iPair> > edges;

  Graph(int V, int E)

  {

    this->V = V;

    this->E = E;

  }

  void addEdge(int u, int v, int w)

  {

    edges.push\_back({w, {u, v}});

  }

  int kruskalMST();

};

struct DisjointSets

{

  int \*parent, \*rnk;

  int n;

  DisjointSets(int n)

  {

    this->n = n;

    parent = new int[n+1];

    rnk = new int[n+1];

    for (int i = 0; i <= n; i++)

    {

      rnk[i] = 0;

      parent[i] = i;

    }

  }

  int find(int u)

  {

    if (u != parent[u])

      parent[u] = find(parent[u]);

    return parent[u];

  }

  void merge(int x, int y)

  {

    x = find(x), y = find(y);

    if (rnk[x] > rnk[y])

      parent[y] = x;

    else

      parent[x] = y;

    if (rnk[x] == rnk[y])

      rnk[y]++;

  }

};

int Graph::kruskalMST()

{

  int mst\_wt = 0;

  sort(edges.begin(), edges.end());

  DisjointSets ds(V);

  vector< pair<int, iPair> >::iterator it;

  for (it=edges.begin(); it!=edges.end(); it++)

  {

    int u = it->second.first;

    int v = it->second.second;

    int set\_u = ds.find(u);

    int set\_v = ds.find(v);

    if (set\_u != set\_v)

    {

      cout << u << " - " << v << endl;

      mst\_wt += it->first;

      ds.merge(set\_u, set\_v);

    }

  }

  return mst\_wt;

}

int main()

{

  int V = 9, E = 14;

  Graph g(V, E);

  g.addEdge(0, 1, 10);

  g.addEdge(0, 2, 12);

  g.addEdge(1, 2, 9);

  g.addEdge(1, 3, 8);

  g.addEdge(2, 4, 3);

  g.addEdge(2, 5, 1);

  g.addEdge(3, 4, 7);

  g.addEdge(3, 7, 5);

  g.addEdge(3, 10, 8);

  g.addEdge(4, 5, 3);

  g.addEdge(5, 7, 6);

  g.addEdge(6, 7, 9);

  g.addEdge(6, 8, 2);

  g.addEdge(7, 8, 11);

  cout << "Edges of MST are \n";

  int mst\_wt = g.kruskalMST();

  cout << "\nWeight of MST is " << mst\_wt;

  return 0;

}

