```
/*
Student name: Joaquin De Losada
Student ID: 6778757
Professor: Yifang Li
 */

public class Main {
    public static void main(String[] args) {
        DrugGraph dg = new DrugGraph();
        dg.ReadData();
        dg.FindModules();
        dg.KeepAModule(0);
        dg.FindShortestPath("DB01050", "DB00316", "unweighted");
        //dg.FindShortestPath('DB01050', 'DB00316', 'weighted');
        //dg.MSTPrim();
    }
}


public class Vertex {
    public String genericName, sMILES, drugBankID, url, drugGroups, score,
path;

    public int moduleGroup, posInArray;

    public float dist;

    public boolean wasVisited;

    public Vertex(){

    }

    public Vertex(String GenericName, String SMILES, String DrugBankID,
String URL, String DrugGroup, String Score, boolean WasVisited, float Dist,
String Path){
        SetGenericName(GenericName);
        SetSmiles(SMILES);
        SetDrugBankID(DrugBankID);
        SetURL(URL);
        SetDrugGroup(DrugGroup);
        SetScore(Score);
        SetVisitied(WasVisited);
        SetDist(Dist);
        SetPath(Path);
    }

    public void SetGenericName(String name){this.genericName = name;}

    public void SetSmiles(String sMILES){this.sMILES = sMILES;}

    public void SetDrugBankID(String drugID){this.drugBankID = drugID;}

    public void SetURL(String url){this.url = url;}

    public void SetDrugGroup(String group){this.drugGroups = group;}
```

```
    public void SetScore(String score){this.score = score;}

    public void SetVisitied(boolean wasVisited){this.wasVisited =
wasVisited;}

    public void SetDist(float dist){this.dist = dist;}

    public void SetModule(int moduleGroup){this.moduleGroup = moduleGroup;}

    public void SetPosInArray(int posInArray){this.posInArray = posInArray;}

    public void SetPath(String path){this.path = path;}

    public void DisplayDrug(){
        System.out.println("Drug name: " + genericName);
        System.out.println("Drug SMILES?: " + sMILES);
        System.out.println("Drug ID: " + drugBankID);
        System.out.println("Drug URL: " + url);
        System.out.println("Drug Group: " + drugGroups);
        System.out.println("Drug Score: " + score);
        System.out.println("Drug Visited: " + wasVisited);
        System.out.println("Drug Distance: " + dist);
        System.out.println("Drug Path: " + path);
        System.out.println(" ");
    }

    public String ReturnName(){return genericName;}

    public String ReturnSMILES(){return sMILES;}

    public String ReturnDrugBankID(){return drugBankID;}

    public String ReturnURL(){return url;}

    public String ReturnGroup(){return drugGroups;}

    public String ReturnScore(){return score;}

    public boolean ReturnVisited(){return wasVisited;}

    public float ReturnDist(){return dist;}

    public int ReturnModule(){return moduleGroup;}

    public String ReturnPath(){return path;}
}


import java.io.*;
import java.lang.Math;
import java.nio.BufferUnderflowException;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Objects;
import java.util.Scanner;
```

```
public class DrugGraph {
    public ArrayList<String> vertexList = new ArrayList<>();
    public ArrayList<String> simmatList = new ArrayList<>();
    public ArrayList<Vertex> sameModuleList = new ArrayList<>();
    public Vertex[] vertices, keepModule;

    public Queue<Vertex> q = new LinkedList<>();

    public LinkedList<Vertex> linkedList;

    public float[][] w, w2;
    public int[][] a, a2;

    public int toTheLeft;
    public int upTheMatrix;

    File writeToInOrderTraverse = new File("recourses//MSTPrimResult.txt");
    public BufferedWriter writeFileInOrder;

    public void ReadData() {
        LoadMainData();

        //Runs the method for creating a method and allowing java to later
write to said method
        CreateFile();
    }

    public void LoadMainData(){
        File mainTxtFile = new File("recourses//dockedApproved.tab"); //Gets
the file
        File simmatTxtFile = new File("recourses//sim_mat.tab"); //Gets the
file

        BufferedReader mainReadFile, sim_mat;

        int totalArrayLength = 0; //Size of array for vertexes and 2d matrix
        int row = 0;

        try{ //Try this code
            mainReadFile = new BufferedReader(new FileReader(mainTxtFile));
//Read file
            sim_mat = new BufferedReader(new FileReader(simmatTxtFile));
//Read sim mat file

            String line = ""; //String that will contain each line
            String lineSimMat = "";
            //ArrayList<String> drugList = new ArrayList<>();
            //System.out.println(readFile);
            while(line != null){
                line = mainReadFile.readLine(); //Read the line
                if (line == null) {
                    break;
                } //If the current line is null it will stop reading
                if (line.contains("Generic Name")) {} //If the line ever
contains name then it will do nothing. This is used for the first line of the
txt which dosent contain a patient
                else { //Will add to the current patients list and increase
```

```
size of patients array for later
                    totalArrayLength++;
                    vertexList.add(line);
                }
            }

            //Will read the sim mat file and add each line to an array list
            while (lineSimMat != null) { //Loop through the txt file
                lineSimMat = sim_mat.readLine(); //Read the line
                if (lineSimMat == null) {
                    break;
                }
                simmatList.add(lineSimMat); //Add to the simmat list
            }

            w = new float[totalArrayLength][totalArrayLength]; //Create a 2d
array for the weighted matrix
            a = new int[totalArrayLength][totalArrayLength]; //Create a 2d
array for the unweighted matrix
            vertices = new Vertex[totalArrayLength]; //Create the array of
ADT of drug of the size of patients
            //System.out.println(vertexList.get(0));
            for(String i : vertexList){ //Go through each arraylist of drug
string to split the string up and add it to a drug data type
                String[] currentPatient = i.split("\\t"); //Spliting the drug
into an array
                Vertex newVertex = new Vertex(); //Create a drug/vertex data
type
                newVertex.SetGenericName(currentPatient[0]); //Set the
drug/vertex name
                newVertex.SetSmiles(currentPatient[1]); //Set the drug/vertex
smiles
                newVertex.SetDrugBankID(currentPatient[2]); //Set the
drug/vertex ID
                newVertex.SetURL(currentPatient[3]); //Set the drug/vertex
URL
                newVertex.SetDrugGroup(currentPatient[4]); //Set the
drug/vertex group
                newVertex.SetScore(currentPatient[5]); //Set the drug/vertex
score
                newVertex.SetVisitied(false); //Sets the visited to false
                newVertex.SetModule(-1); //Sets the current module group to -
1
                newVertex.SetPosInArray(vertexList.indexOf(i));
                vertices[vertexList.indexOf(i)] = newVertex; //Add the drug
to the array of drugs
            }
            //System.out.println(vertices.length);
            System.out.println("Loaded Drugs/Vectors."); //Confirms that
vertex have been loaded
        } catch (IOException e) { //If the file isnt found then print this
            System.out.println("File not found. Did you try to move it? Not a
good idea return it or give me 100%.");
        }

        //Creates a weighted 2d matrix and confirms that it was made
        WeightedMatrix();
```

```
        System.out.println("Weighted matrix made");

        //Creates a unweighted 2d matrix and confirms it was made
        UnWeightedMatrix();
        System.out.println("Unweighted matrix made.");
    }

    //Create a weighted matrix
    public void WeightedMatrix(){
        for(String x : simmatList){
            //String tokanizer
            String[] simmatValues = x.split("\\t"); //Spliting the distance
string into an array
            //Will l
            for(int y = 0; y < simmatValues.length; y++){
                float currentVal = Float.parseFloat(simmatValues[y]); //Will
grab the current string value and turn it into a float
                float WeightedValv = (1 - currentVal); //Grab the current
float and turn it into the weighted value which will be used for the weighted
matrix

                //If less than 0.7 then it means both drugs/vertexs are
connected and will add the value to the weighted 2d matrix
                if(WeightedValv <= 0.7){
                    //System.out.println("Connected Graph");
                    w[simmatList.indexOf(x)][y] = WeightedValv;
                }else{ //Else if both drugs are not connected I make the
value infinte
                    w[simmatList.indexOf(x)][y] = (float)(1.0/0.0);
                }
            }
        }
    }

    //Create an unweighted matrix
    public void UnWeightedMatrix(){
        //Loops through the weighted graph
        for(int x = 0; x < w.length; x++){
            for(int y = 0; y < w.length; y++){
                if(w[x][y] <= 0.7){ //If the current value is less than 0.7
then it will add a 1 to the unweighted matrix
                    a[x][y] = 1;
                }else{ //Else if its any other number than it will place a 0
                    a[x][y] = 0;
                }
            }
        }
    }

    //Will loop through the verticis array and find all vertexes that are
part of certain group IDs
    public void FindModules(){
        //System.out.println("Finding modules");
        int moduleGroup = 0; //How do I know when to increase

        for(int i = 0; i < vertices.length; i++){
            //System.out.println(vertices[i].wasVisited);
```

```
            if(!vertices[i].wasVisited){ //If the vertex hasent been visited
then run BFS
                //System.out.println(moduleGroup);
                BFS(vertices[i], moduleGroup);
                moduleGroup++;
                //System.out.println("Module group found");
            }
            //System.out.println(moduleGroup);
        }
        System.out.println(moduleGroup);
        System.out.println("All modules found.");
    }

    public void BFS(Vertex s, int moduleGroup){
        LinkedList<Vertex> bfsLL = new LinkedList<>();
        for(Vertex vertex : vertices){
            vertex.SetDist((float)(1.0/0.0));
        }
        s.dist = 0;
        bfsLL.add(s); //Add initial vertex to linked list

        //While the linked list is not empty it will see if it is related to
any other vertex
        while(!bfsLL.isEmpty()) {
            Vertex v = bfsLL.remove(); //Remove vertex from linked list
            for (int i = 0; i < vertices.length; i++){
                //If the current vertex hasnt been visited and its related to
the current i drug then it will set the module group for the drug
                if (!vertices[i].wasVisited && w[i][v.posInArray] !=
(float)(1.0 / 0.0)) {
                    vertices[i].wasVisited = true;
                    vertices[i].SetModule(moduleGroup);
                    bfsLL.add(vertices[i]); //Add new vertex to linked list
                }
            }
            //System.out.println(bfsLL.size());
        }
    }

    //Will create a new arraylist and a pair of 2d matirxs that contain only
vertexes that are part of a specific module
    public void KeepAModule(int moduleID){
        int size = 0;
        for(Vertex vertex : vertices){ //Will loop through every vertex in
the vertices array and if its part
            vertex.SetVisitied(false);
            if(vertex.ReturnModule() == moduleID){
                sameModuleList.add(vertex);
                size++;
            }
        }
        //System.out.println(sameModuleList.size());
        w2 = new float[size][size]; //Create a new 2d matrtix that is the
size of the module group for the weighted values
        a2 = new int[size][size]; //Create a new 2d matrtix that is the size
of the module group for the unweighted values
```

```
        upTheMatrix = 0; //Counter for the amount of rows I need to place the
new rows
        //Loop through each row of the 2d matrix (Since both matrixes are the
same size)
        for(int x = 0; x < w.length; x++){
            //If the current row value is part of a diffrent module then it
will just increse the row counter.
            toTheLeft = 0; //Counter for the amount of columns I need to
place the new column
            if(vertices[x].ReturnModule() != moduleID){
                upTheMatrix++;
            }else{
                //Loop through each column which in combination with the
first for loop gives me the position for both drugs distances.
                for (int y = 0; y < w.length; y++) {
                    //If the current column drug is not part of the module
group that is being searched then it will just increase the column counter
                    if (vertices[y].ReturnModule() != moduleID) {
                        toTheLeft++;
                    }else{
                        //Will transfer the values from both the weighted and
unweighted matrixes to the new and updated matrixes.
                        w2[x - upTheMatrix][y - toTheLeft] = w[x][y]; //Place
weighted value into new matrix
                        a2[x - upTheMatrix][y - toTheLeft] = a[x][y]; //Place
unweighted value into new matrix
                    }
                }
            }

        }

        System.out.println("Module " + moduleID + " kept.");
    }

    //Will determine which matrix (Unweighted or weighted matrixes) was
chosen and find the shortest path between the two chossen drugs.
    public void FindShortestPath(String fromVertex, String toVertex, String
method){
        Vertex fromVertexObj = null,toVertexObj = null;
        for(int i = 0; i < vertices.length; i++){
            //Will turn the string that the method recives into the vertex
data type
            if(Objects.equals(vertices[i].ReturnDrugBankID(), fromVertex)){
                fromVertexObj = vertices[i];
            }
            if(Objects.equals(vertices[i].ReturnDrugBankID(), toVertex)){
                toVertexObj = vertices[i];
            }
        }

        /* Couldnt get the methods working in time so it will only be partial
marks */
        //Will run the correct method for either unweighted or weighted
matrixes
        if(Objects.equals(method, "unweighted")){
            //unweightedShortestPath(fromVertexObj, toVertexObj);
```

```
        } else if (Objects.equals(method, "weighted")) {
            //WeightedShortestPath(fromVertexObj,toVertexObj);
        }
    }

    public void unweightedShortestPath(Vertex start, Vertex finish) {
        for(int i = 0; i < keepModule.length; i++){
            keepModule[i].dist = (float)(1.0/0.0);
            keepModule[i].wasVisited = false;
        }
        finish.dist = 0;

        /*for (int currDist = 0; currDist < NUM_VERTICES; currDist++) {
            for (int i = 0; i < vertices.length; i++) {
                if (!vertices[i].wasVisited && vertices[i].dist == currDist)
{
                    vertices[i].wasVisited = true;
                    for each Vertex w adjacent to v {
                        if (w.dist == INFINITY) {
                            w.dist = currDist + 1;
                            w.path = v;
                        }
                    }
                }
            }
        }*/
    }

    public void WeightedShortestPath(Vertex start, Vertex finish){

        for(int i = 0; i < sameModuleList.size(); i++){
            sameModuleList.get(i).dist = (float)(1.0/0.0);
            sameModuleList.get(i).wasVisited = false;
        }
        finish.dist = 0;

        /*while( there is an unknown distance vertex ){
            Vertex v = smallest unknown distance vertex;
            v.known = true;
            for each Vertex w adjacent to v
            if( !w.known )
            {
                DistType cvw = cost of edge from v to w;
                if( v.dist + cvw < w.dist )
                {
                    // Update w
                    decrease( w.dist to v.dist + cvw );
                    w.path = v;
                }
            }
        }*/
    }

    //Turn a string into an int ID
    public int DrugIDToInt(String drugID){
        String[] drugIDString = drugID.split("");
```

```
        return Integer.parseInt(drugIDString[1]);
    }

    //Grab the ID from the drug
    public int DrugIDToInt(Vertex vertex){
        String[] drugIDString = vertex.ReturnDrugBankID().split("B");

        return Integer.parseInt(drugIDString[1]);
    }

    //Will create a new file if it dosent exist or will state that said file
already exists.
    public void CreateFile(){
        try {
            if(writeToInOrderTraverse.createNewFile()) {
                System.out.println("File Created: " +
writeToInOrderTraverse.getName());
            }else{
                System.out.println("File exists.");
            }

            writeFileInOrder = new BufferedWriter(new
PrintWriter(writeToInOrderTraverse));
        }catch(IOException e){
            System.out.println("Error 404");
        }
    }
}
```