discrete structures for computer science

**ID NO: 2018A7PS0173U**

**Name: ASHISH KUMAR JHA**

**Section:2**

I would like to express my deep gratitude and sincere thanks to my faculty, Dr. Shilpee Saxena and for her invaluable guidance, advice and constructive comments which has sustained my efforts at all stages of this assignment. This Assignment has helped me further my knowledge not only in the field of coding, but it has also helped me understand concepts regarding graphs and its applications in the real world.

I would also like to thank my parents and friends for their help and valuable assistance. Last but not the least I would like to thank God, the source of all my talents.

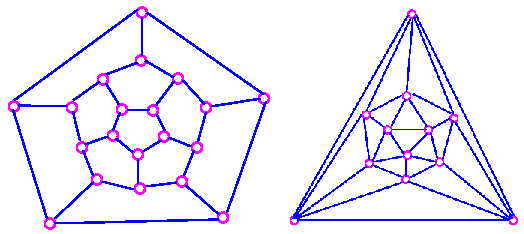
A picture containing whiteboard

Description automatically generated

**Acknowledgement**

Introduction

Discrete mathematics is the part of mathematics devoted to the study of discrete (i.e. distinct) objects. In general, it is used whenever objects are counted, when relationships between finite (or countable) sets are studied, and when processes involving a finite number of steps are analyzed. It is important for computer science because in computing machines, information is stored and manipulated in a discrete fashion. This assignment will focus on a part of the course i.e. Graphs. A graph is a collection of nodes connected by edges. A graph is also frequently called a network, a node may be called a vertex, and an edge may be called a link.  Commonly, a graph is denoted by an ordered pair, **G=(V, E):** where **V** is the set of vertices and **E** is the set of edges

.

Question

The question assigned according to the algorithm is Q54.

Q54: Write a program which computes the number of regions in a planar

graph, when the planar graph is represented using adjacency list

Solution

This problem has been solved using the software: Java SE11

Python is an interpreted, high-level, general-purpose programming language. MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

* **Java Code:**

This code will be used to count the number of regions in a given planar graph. Region of a plane graph are vertices connected in portions of the graph forming a closed component**.**

import java.util.\*;

public class Region

{

public static int calcRegion(int v,int e)

{

int r;

r = e - v + 2;

return r;

}

public static void main(String[] args)

{

Scanner s = new Scanner (System.in);

int i=0;

int n;

char arr[] = new char[26];

for(i=0;i<26;i++)

{

arr[i]=(char)(i+65);

}

System.out.println("Please Enter The Number Of Vertices In The Graph");

n=s.nextInt();

System.out.print("Let the vertices be:-");

for(i=0;i<n;i++)

{

if(i<n-1)

{

System.out.print(arr[i]+",");

}

else

{

System.out.println(arr[i]);

}

}

System.out.println("Enter the adjacency list one by one:-");

i=0;

String al;

String [] ale = new String[n];

int sum=0;

for (i=0; i<n;i++)

{

System.out.print("Enter the vertices connected to vertex "+arr[i]+" :");

if(i==0)

s.nextLine();

al=s.nextLine();

ale = al.split(" ");

sum = sum + ale.length;

}

int e;

e=sum/2;

int r;

r = calcRegion(n,e);

System.out.println("The number of regions in this graph is: "+r+" .");

}

} {\displaystyle E}

**Explanation of code:**

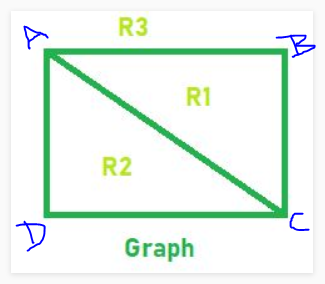
First, we are importing the utilities packages. Next, we create the function that takes in the vertices, edges as input(argument) and gives the number of regions in the graph as the output (return data). Then, in the main function, we have the code for taking the input from the user in the form of adjacency list, where the user tells in the information about the input data. As the user himself enters the number of vertices, we only must find the value of E or the edges, which we do using the relation 2|E|=∑deg(v). This gives us the value of E as well, which we use as the argument along with V to find the number of regions using Euler’s relation: - |R|=|E|-|V|+2.

This equation gives us the number of regions in the graph which we return to the main for displaying. That’s the overall working or the algorithm of the program, the complete program has already been provided above for reference, the symbols are very clear, and the program is easy to understand.

After Compiling and running the code we get the following output for sample input.

The output for the given sample graph follows below (on the next page):

A screenshot of a cell phone

Description automatically generated

Application

Firstly, planar graphs constitute quite simple class of graphs, much simpler than the class of all graphs. So, as the science frequently does, if some algorithmic problem cannot be solved efficiently for all interesting inputs, we can at least strive to solve it for *some* of the inputs. Indeed, many problems that are NP-hard for general graphs turn to possess polynomial-time algorithms when they are restricted to planar graphs (due to their sparsity and lots of interesting structural properties). An example of such classical problem is MAX-CUT. [On the other hand, some problems remain NP-hard even for planar graphs, e.g. variants of COLORING problem.] The *Planar Separator Theorem* due to Lipton and Tarjan helps devise divide-and-conquer-type algorithms for planar graphs.  
  
Secondly, planarity is one of the central notions of the whole graph theory, so just purely from the theoretical point of view it is interesting to consider planar graphs in algorithmic framework. For example, while there is a bunch of existing algorithms for testing graph planarity (with linear time complexity), the topic is still being researched and new optimizations and simplifications are being discovered.  
  
Although being almost too structurally simple, planar graphs should not be considered non-applicable to real-life. For example, the task of large electronic circuit layout employs planar graph layout, algorithms for splitting a general graph into planar components etc. Many algorithms for graph drawing, although targeting non-planar graphs, have a planar-oriented core, that is, try to make an input graph planar, then draw it, and then get back to the original graph.

Regions help in solving various real-life problems in various fields, viz, computer science, electrical science and even city planning, architecture, and even in corporate fields for area distribution properly.

Example, if a city planning department has to find out the different regions in a city for making proper policies, the important places in city can be marked as vertices of the graph and then the program can be used to find out the regions for proper plan implementation.

For companies like amazon, flip kart etc., they must make offices in almost every region.

They divide the country into various states or cities as the cities are marked as vertices on the planar graph and then the no of regions can be found out.

For metro stations, the govt wants metro stations in every region in the city, they divide the city into important landmarks which work as the vertices of the graph and then the regions are found which tell us where all to make metro stations.

Similarly, for simplifying electrical circuits and knowing the path of current in each branch etc.

Just like this there are many real-life uses of planar graph and regions of a graph.

A picture containing whiteboard

Description automatically generated