MMTE-001
GRAPH THEORY

Indira Gandhi National Open University
School of Sciences

Curriculum Design Committee

Dr. B.D. Acharya
Dept. of Science & Technology, New Delhi

Prof. Adimurthi
School of Mathematics
TIFR, Bangalore

Prof. Archana Aggarwal
CESP, School of Social Sciences
JNU, New Delhi

Prof. R.B. Bapat
Indian Statistical Institute
New Delhi

Prof. M.C. Bhandari
Dept. of Mathematics
IIT, Kanpur

Prof R. Bhatia
Indian Statistical Institute, New Delhi

Prof. A.D. Dharmadhikari
Dept. of Statistics
University of Pune

Prof. O.P. Gupta
Dept. of Financial Studies
University of Delhi

Prof. S.D. Joshi
Dept. of Electrical Engineering
IIT, Delhi

Dr. R.K. Khanna
Scientific Analysis Group, DRDO, Delhi

Prof. Susheel Kumar
Dept. of Management Studies
IIT, Delhi

Prof. Veni Madhavan
Scientific Analysis Group
DRDO, Delhi

Prof. J.C. Misra
Dept. of Mathematics
IIT, Kharagpur

Prof. C. Musili
Dept. of Mathematics and Statistics
University of Hyderabad

Prof. Sankar Pal
ISI, Kolkata

Prof. A.P. Singh
PG Dept. of Mathematics
University of Jammu

Faculty Members
School of Sciences, IGNOU
Prof. Poornima Mital
Dr. Atul Razdan
Prof. Parvin Sinclair
Prof. Sujatha Varma
Dr. S. Venkataraman

Course Design Committee

Prof. C. A. Murthy
ISI, Kolkatta

Prof. S. B. Pal
IIT, Kharagpur

Prof. C. E. Veni Madhavan
IISC, Bangalore

Dr. B. S. Panda
IIT, Delhi

Faculty Members
School of Sciences, IGNOU
Prof. Parvin Sinclair
Prof. Poornima Mital
Dr. S. Venkataraman
Dr. Deepika
Dr. Atul Razdan

Block Preparation Team

Prof. B.S. Panda (Editor)
Dept. of Mathematics
IIT Delhi

Dr. Rashmi Jain
Consultant
School of Sciences
IGNOU

Dr. Pawan Kumar
School of Sciences
IGNOU

Course Coordinator: Dr. Pawan Kumar

Acknowledgements: To Mr. Prashant Kumar for typesetting the manuscript, and Mr. Surender Singh Chauhan for preparing the CRC.

PRINT PRODUCTION TEAM

Mr. Rajiv Girdhar
A.R.(P) MPDD-IGNOU,
Maidan Garhi, Delhi-110068

Mr. Hemant Kumar Parida
S.O.(P) MPDD-IGNOU,
Maidan Garhi, Delhi-110068

December, 2021

© *Indira Gandhi National Open University, 2021*

ISBN : 978-93-5568-035-8

All right reserved. No part of this work may be reproduced in any form, by mimeograph or any other means, without permission in writing from the Indira Gandhi National Open University.

Further information on the Indira Gandhi National Open University courses, may be obtained from the University's office at Maidan Garhi, New Delhi-110 068 and IGNOU website www.ignou.ac.in.

Printed and Published on behalf of the Indira Gandhi National Open University, New Delhi by the Registrar, MPDD. IGNOU, New Delhi

Printed at : Akashdeep Printers, 20-Ansari Road, Daryaganj, New Delhi-110002

COURSE INTRODUCTION

Graph theory is the study of mathematical structures which are used to model the situations involving pairwise relations between objects from a certain collection. A "graph" in this context refers to a collection of *vertices* along with a collection of *edges* that connect certain pairs of vertices. The birth of graph theory took place long back while dealing with "The Königsberg Bridge Problem". In the 18^{th} century in the town of Königsberg, Germany, a favorite pastime was walking along the Pregel River and strolling over the town's seven bridges. During this period a natural question arose: Is it possible to take a walk and cross each bridge only once? This question was solved by the Swiss mathematician Leonard Euler and his solution was the beginning of network theory.

A graph may be *undirected*, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be *directed* from one vertex to another. In real life, a graph can be modelled, for example, to find the minimum cost to make every telephone reachable from every other, to find the fastest route from the national capital to each state capital, or to schedule the IPL 20-20 series into the minimum number of weeks. Graph theoretical concepts can also be used to colour the regions of a map. The discovery of Four-Colour Theorem is a major development in this direction.

Graphs are extremely important in Computer Science. They can be used to represent many problems of computer science that are otherwise abstract. Finding a way to represent the solution to a problem as a graph can present new approaches to solve the problem or even lead directly to a solution derived from graph theory.

Graph algorithms are also of much interest to emerging disciplines such as computational molecular biology and computational chemistry.

In the first block of this course we will discuss certain basic concepts such as paths, cycles, trails, different useful models, bipartite graphs, connected graphs, minimal spanning graphs, trees and distance, finding the shortest paths from one vertex to another.

In Block 2, we shall talk about Eulerian and Hamiltonian graphs, and some of their applications. Next we shall define the concept of graph colouring, and see how it can be used to solve some real life problems. Finally, we shall discuss the concept of planar graphs and some of their characterisations.

Block 3 begins with the concept of matchings and covers. You will see here Hall's Theorem for the existence of a special matching in a bipartite graph. We shall also present some algorithms for finding matchings of maximum size (in unweighted graphs) and maximum weight (in weighted graphs). The next concept you will study in this block is the connectivity of graphs. You will see the definitions and results on k-connected and k-edge-connected graphs. At last, we talk about flows in networks, and discuss how to find a maximum flow in a network. We shall also present necessary theoretical background for the existence of a maximum flow in a network.

This course also has a practical component given in the **Appendix** at the end. It consists of a set of practicals which are to be carried out at your Programme Centre. You may refer to the programme guide for more details regarding the evaluation and other aspects of the practical component.

Some useful books and websites are the following:

1) "Introduction to Graph Theory", Douglas West, Pearson
2) "Graph Theory with Applications to Engineering and Computer Science", Narsingh Deo, PHI
3) "Graph Theory: Modeling, Applications and Algorithms", Agnarsson, Pearson.
4) Youtube Graph Series by Sarada Herke (https://www.youtube.com/c/Saradaherke/Featured)

We hope you enjoy the course!

ignou
THE PEOPLE'S
UNIVERSITY

MMTE-001
GRAPH THEORY

Indira Gandhi National Open University
School of Sciences

Block

1
FUNDAMENTALS OF GRAPH THEORY

Curriculum Design Committee

Dr. B.D. Acharya
Dept. of Science & Technology, New Delhi

Prof. Adimurthi
School of Mathematics
TIFR, Bangalore

Prof. Archana Aggarwal
CESP, School of Social Sciences
JNU, New Delhi

Prof. R.B. Bapat
Indian Statistical Institute
New Delhi

Prof. M.C. Bhandari
Dept. of Mathematics
IIT, Kanpur

Prof R. Bhatia
Indian Statistical Institute, New Delhi

Prof. A.D. Dharmadhikari
Dept. of Statistics
University of Pune

Prof. O.P. Gupta
Dept. of Financial Studies
University of Delhi

Prof. S.D. Joshi
Dept. of Electrical Engineering
IIT, Delhi

Dr. R.K. Khanna
Scientific Analysis Group, DRDO, Delhi

Prof. Susheel Kumar
Dept. of Management Studies
IIT, Delhi

Prof. Veni Madhavan
Scientific Analysis Group
DRDO, Delhi

Prof. J.C. Misra
Dept. of Mathematics
IIT, Kharagpur

Prof. C. Musili
Dept. of Mathematics and Statistics
University of Hyderabad

Prof. Sankar Pal
ISI, Kolkata

Prof. A.P. Singh
PG Dept. of Mathematics
University of Jammu

Faculty Members
School of Sciences, IGNOU
Prof. Poornima Mital
Dr. Atul Razdan
Prof. Parvin Sinclair
Prof. Sujatha Varma
Dr. S. Venkataraman

Course Design Committee

Prof. C. A. Murthy
ISI, Kolkatta

Prof. S. B. Pal
IIT, Kharagpur

Prof. C. E. Veni Madhavan
IISC, Bangalore

Dr. B. S. Panda
IIT, Delhi

Faculty Members
School of Sciences, IGNOU
Prof. Parvin Sinclair
Prof. Poornima Mital
Dr. S. Venkataraman
Dr. Deepika
Dr. Atul Razdan

Block Preparation Team

Prof. B.S. Panda (Editor)
Dept. of Mathematics
IIT Delhi

Dr. Rashmi Jain
Consultant
School of Sciences
IGNOU

Dr. Pawan Kumar
School of Sciences
IGNOU

Course Coordinator: Dr. Pawan Kumar

Acknowledgements: To Mr. Prashant Kumar for typesetting the manuscript, and Mr. Surender Singh Chauhan for preparing the CRC.

BLOCK INTRODUCTION

This is the first block of the course Graph Theory. In this block, which is divided into 4 units we discuss certain basic concepts of graph theory.

In Unit 1 we start with the definition and explain the meaning of "graph" and discuss some fundamental concepts such as types of graphs, subgraph and complement of a graph, vertex degree, etc. You will see how the sum of the vertex degrees relates to the number of edges in a graph, in the form of Degree-sum formula. Then you will see the concepts of walks, paths, trails, circuits, cycles, cut-vertices, cut-edges which are essential for many other concepts such as connectedness.

Unit 2 discusses the concept of degree sequence and graph representation. We will discuss Havel-Hakimi Theorem about degree sequences. You will see how matrices can be used to represent and study properties of graphs.

In Unit 3, the attention is given on one particularly important and useful type of graph– that is known as a tree. Although trees are relatively simple structures, they form the basis of many of the practical techniques used to model and to design large scale systems. We discuss some characterisation of trees along with some additional concepts such as spanning trees, k -ary trees.

Unit 4 lists some algorithms on trees. For instance, the BFS and DFS algorithms construct spanning trees in unweighted graphs, whereas Kruskal's and Prim's algorithms construct spanning trees with minimum weight in weighted graphs. There is one more algorithm, namely the Dijkstra's Algorithm, which finds the distance from a given vertex to all the other vertices in the graph.

Lastly, we advise you once again to carefully go through the solved examples, and to attempt all the exercises in each unit. This will help you grasp the theory better.

ignou
THE PEOPLE'S
UNIVERSITY

NOTATIONS AND SYMBOLS (Used in Block 1)

$\mathbb{N}$	the set of natural numbers
$\mathbb{R}$	the set of real numbers
$A \setminus B$	the complement of the set B in the set A
$\emptyset$	empty set
$V(G)$	the vertex set of a graph (or multigraph) G
$E(G)$	the edge set of a graph (or multigraph) G
$N_G(v)$	the neighbourhood of a vertex v in G
$u \leftrightarrow v$	u is adjacent to v
$u \not\leftrightarrow v$	u is not adjacent to v
$n(G)$	the order of G
$m(G)$	the size of G
K_n	a complete graph on n vertices
$K_{n,n}$	a complete bipartite graph with m and n vertices in its partite sets
$\overline{G}$	the complement of a graph G
(G,W)	a weighted graph G with weight function W
$d_G(v)$	the degree of a vertex v in G
$\delta(G)$	the minimum degree of G
$\Delta(G)$	the maximum degree of G
C_n	a cycle on n vertices
P_n	a path on n vertices
S_n	an n-vertex star
W_n	an n-vertex wheel graph
$\binom{n}{k}$	the number of ways to select k objects from n distinct objects
$c(G)$	the number of components of G
$G - v$	the subgraph of G obtained by removing the vertex v and all the edges incident on v
$G - e$	the subgraph of G obtained by removing the edge e
$G + uv$	the graph (or multigraph) obtained by joining the vertices u and v of G
$d_G(u,v)$	the distance between the vertices u and v of G
$\mathrm{diam}(G)$	the diameter of G
$\varepsilon_G(v)$	the eccentricity of v in G
$\mathrm{rad}(G)$	the radius of G
Q_k	a k-dimensional cube
$A(G)$	the adjacency matrix of G
$M(G)$	the incidence matrix of G
$G_1 \cong G_2$	G_1 is isomorphic to G_2
$\tau(G)$	the number of spanning trees of a labelled graph G
(T,r)	a rooted tree T with root r

UNIT 1

FUNDAMENTAL CONCEPTS

Structure **Page Nos.**

1.1 INTRODUCTION

Basic ideas of graph theory were introduced in the eighteenth century by the great Swiss mathematician Leonhard Euler. He used graphs to solve the famous Königsberg Bridge Problem, which we will discuss in this unit.

Graphs are used to solve problems in many fields. For instance, graphs can be used to determine whether a circuit can be implemented on a planar circuit board. Using graphs, we can distinguish between two chemical compounds with the same molecular formula, but with different structures. Graphs can be used to study the structure of the World Wide Web. We can determine whether two computers are connected by a communication link using graph models of computer networks. Graphs with weights assigned to their edges can be used to solve problems such as finding the shortest path between two cities in a transportation network. We can also use graphs to schedule exams and assign channels to television stations.

In this unit, we shall begin with Section 1.2 some graph models, along with basic definitions of graph theory. In Section 1.3 we shall study different types of graphs. Section 1.4 deals with the notion of subgraph, and the complement of a graph. Section 1.5 is devoted to the study of vertex degree and the resulting concepts such as even graphs and regular graphs. In Section 1.6, we shall study walk, trail, path etc. and connected graphs and Section 1.7 is about cut-vertices and cut-edges.

Objectives

After studying this unit, you should be able to:

- explain 'graph models';
- describe what a graph is, and different types of graphs such as directed graphs, and weighted graphs;
- determine subgraphs, spanning subgraphs, and induced subgraphs of a graph;
- explain what are complete graphs, bipartite graphs, and complete bipartite graphs;
- find the neighbourhood and degree of a vertex;
- describe odd and even vertices, and their properties;
- find the complement of a graph;
- apply Handshaking Lemma;
- explain the terms walks, trails, paths, cycles, and circuits;
- describe connected graphs, and find the components of a disconnected graph;
- explain what are cut-vertices, and cut-edges.

1.2 GRAPHS AS MODELS

In this section, we shall discuss some real-life problems, which can be modelled as a graph. These problems can be solved, or be proved to be unsolvable, using the techniques of graph theory, as you will see in later units of this course. Along with, we shall present the basic definitions of graph theory.

Let us start with the 'Königsberg Bridge Problem'.

Königsberg Bridge Problem

The **Königsberg Bridge Problem** is a recreational mathematical puzzle, set in the old Prussian city of Königsberg(now Kaliningard, Russia), that **led to the development of the branches of mathematics known as topology and graph theory.** In the early 18th century, the citizens of Königsberg spent their days walking on the intricate arrangement of bridges across the river Pregel, which surrounded two central landmasses connected by a bridge (3). (See Fig. 1.) Additionally, the first landmass (an island) was connected by two bridges (5 and 6) to the lower bank of the Pregel and also by two bridges (1 and 2) to the upper bank, while the other landmass (which split the Pregel into two branches) was connected to the lower bank by one bridge (7) and to the upper bank by one bridge (4), for a total of seven bridges. According to folklore, the question arose of whether a citizen could take a walk through the town in such a way that each bridge would be crossed exactly once.

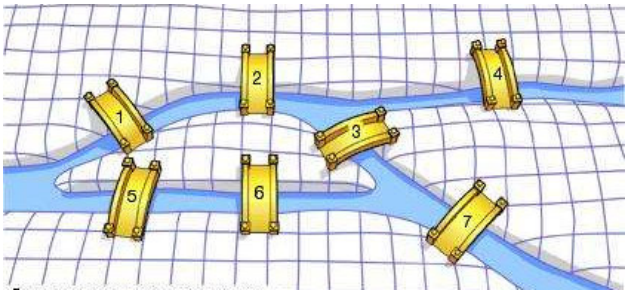


Fig. 1: Bridges of Königsberg

In 1735 the Swiss mathematician Leonhard Euler presented a solution to this problem, concluding that such a walk was impossible.

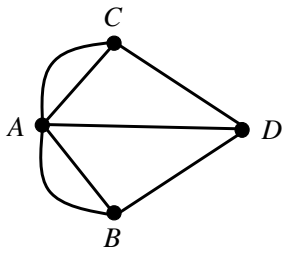


Fig. 2

In proving that the problem had no solution, Euler replaced each land area by a point (A, B, C and D corresponding to four landmasses) and each bridge by a line joining the corresponding points as shown in Fig. 2. For example, the island A can be reached by five bridges, and so in Fig. 2 there are five lines incident to A. Likewise, there are three lines incident to D corresponding to the three bridges that connect it to the other landmasses. This kind of diagram is called a *multigraph*. Notice that Euler was concerned not with the size or the shape of the bridges, or land regions – but rather with how these bridges were connected. This was a different kind of geometrical problem, which could not be solved by the contemporary tools of geometry.

So, what exactly did Euler do to show that the problem had no solution ? In fact, for Euler this was not a problem at all. What concerned him, and took his long time, was a generalised case consisting of n landmasses connected via m bridges.

For the generalised case, he proved that *such a walk is impossible if the graph contains any point on which an odd number of lines are incident*.

Before discussing other real life problems, let us first look at the basic definitions.

Basic Definitions

Definition: A **multigraph** G is a triple consisting of a **vertex set** $V(G)$ (which is nonempty), an **edge set** $E(G)$, and a relation that associates each edge with two vertices (not necessarily distinct) called its **endpoints**. The relation between the edges and vertices of G is defined as follows.

A graph is **finite** if $V(G)$ is finite. In this course, we shall discuss only finite graphs.

For each $e \in E(G)$, $f(e) = uv$ for some $u, v \in V(G)$ where u can be the same as v.

Thus an edge can have two endpoints or one. When an edge has only one endpoint it is called a **loop**. Note also that two edges may have the same endpoints. That is, it is possible that $e_1, e_2 \in E(G), e_1 \neq e_2$ but $f(e_1) = f(e_2)$. Such edges are said to be **parallel**. When a multigraph has no loops and no parallel edges it is called a **graph.** In this course we shall mostly be concerned with graphs. However, occasionally we shall discuss multigraphs.

Alternatively, a multigraph can also be defined as an ordered pair $G = (V, E)$, where $V \neq \emptyset$ is the vertex set of G, and E is a 2-element multiset of V, called the edge set of G.

We **represent** a graph or a multigraph on paper by taking each vertex as a point and representing each edge by a curve joining its endpoints. We denote the vertices usually by small letters such as a, b, c etc., u, v, w etc., or v_1, v_2, v_3 etc, or just the digits $1, 2, 3$ etc. An edge in a graph with endpoints u and v is denoted by uv (or by vu). An edge uv is said to be **incident** on its endpoints u and v, and in this case u and v are called **adjacent vertices**.

Consider the following example.

Example 1: Look at the multigraph G given below.

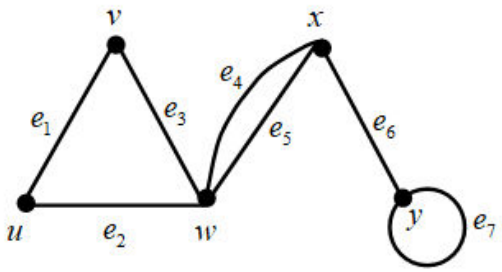


Fig. 3: A multigraph G

Here $V(G)=\{u,v,w,x,y\}$, and $E(G)=\{e_1,e_2,e_3,e_4,e_5,e_6,e_7\}$. The relation between the vertices and the edges of G is as follows:

$$e_1=uv,\ e_2=uw,\ e_3=vw,\ e_4=wx,\ e_5=wx,\ e_6=xy,\ e_7=yy$$

We can see that both e_4 and e_5 have the same endpoints. The edge e_7 has just one endpoint. Thus e_4 and e_5 are parallel edges, and e_7 is a loop. Hence G is a multigraph.

***

Example 2: Now consider the multigraph G given below.

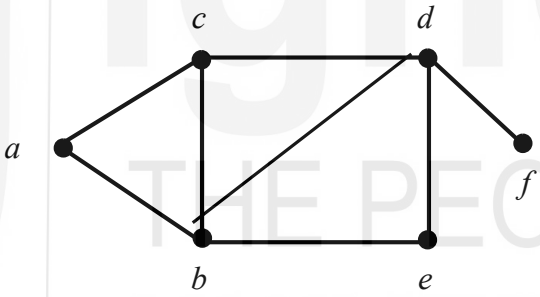


Fig. 4: A graph G

G has no parallel edges and no loops. Hence G is a graph.

***

Definition: The number of vertices in any multigraph is known as the **order** of the multigraph and is denoted by $n(G)$, i.e., $n(G)=|V(G)|$. The number of edges in a multigraph is called the **size** of the multigraph and is denoted by $m(G)$, i.e., $m(G)=|E(G)|$.

For the graph given in Fig. 4, $n(G)=6$ and $m(G)=8$.

Definition: Let G be a graph and $v\in V(G)$. A vertex $u\in V(G)$ is called a **neighbour** of v if $uv\in E(G)$. The set $N_G(v)$ of all neighbours of v is called the **neighbourhood** of v in G. Symbolically,

$$N_G(v)=\{u\in V(G):uv\in E(G)\}.$$

For example, in the graph in Fig. 4, the vertices b and c are the neighbours of a, hence $N_G(a)=\{b,c\}$. Likewise, $N_G(b)=\{a,c,d,e\}$.

If u is a neighbour of v, then v is also a neighbour of u. We shall denote it by $u\leftrightarrow v$.

When G is clear from the context, we shall write $N(v)$ for $N_G(v)$.

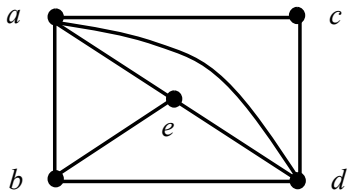
Definition: A set of pairwise adjacent vertices in a graph is called a **clique** and a set of pairwise nonadjacent vertices is called an **independent set.**

nonadjacent = not adjacent

Look at the graph in Fig. 4, again. Here $\{a,b,c\}$ is a clique because a is adjacent to b and c, b is adjacent to c and a, and c is adjacent to a and b. Likewise, you can see that $\{b,c,d\}$ is a clique. However, $\{a,b,c,d\}$ is not a clique, because a is not adjacent to d. The set $\{a,d\}$ is an independent set.

Let us see an example.

Example 3: Consider the graph G given below.



Find the following:

i) the order and size of G

ii) all cliques in G of size at least 3

iii) all independent sets in G

Solution: i) We first note that $V(G) = \{a,b,c,d,e\}$ and $E(G) = \{ab, ac, ad, ae, bd, be, cd, de\}$. Since $|V(G)| = 5$ and $|E(G)| = 8$, order of G is 5 and size is 8.

ii) You can observe that vertices a, b and e are pairwise adjacent, hence $\{a,b,e\}$ is a clique of size 3. Similarly, $\{a,b,d\}, \{a,c,d\}, \{b,e,d\}, \{a,e,d\}$ are also cliques of size 3. There are no other cliques of size 3. The set $\{a, b, d, e\}$ is a clique of size 4. Since any other clique of size 4 would contain c, which does not have 3 neighbours, there are no other cliques of size 4. Also there is no clique of size 5.

iii) You can observe that the vertices b and c are nonadjacent. Hence $\{b,c\}$ is an independent set. Similarly, $\{c,e\}$ is also an independent set. There are no other independent sets.

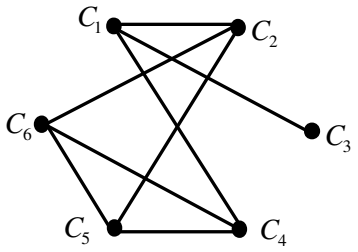
***

Since we have seen the basic definitions concerning graphs, let us see some more applications which can be modelled as graphs.

Scheduling Problem

Suppose six committees of an organisation have to meet on specific time periods in a week. However, certain members are the part of more than one committee. So, any two committees cannot meet at the same time if they have a common member. Let the committees be C_1, C_2, C_3, C_4 and C_5. Further, let C_1 have a member common with C_2, C_3 and C_4; C_2 and C_4 in addition, have a member common with C_5 and C_6; and C_5 has, in addition, a member common with C_6. We represent this information through a graph by treating

each of the C_i s as a vertex. We join two vertices if the corresponding committees have a common member. The graph is given below:



The problem is to allot the minimum number of time periods so that all the committees can meet without any member skipping any meeting.

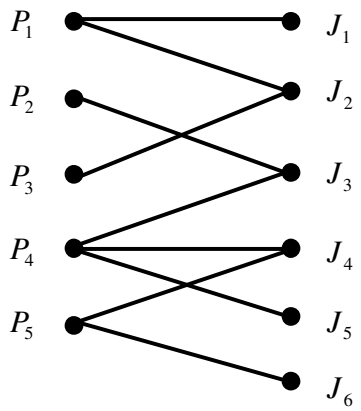
Let us see one more problem.

Assignment Problem

The Assignment Problem is one of the most famous problems in graph theory. Assume we have 5 applicants and 6 jobs. Each applicant is qualified for a certain number of jobs, however every applicant can get at most one job. Let us denote the applicants by P_1, P_2, P_3, P_4 and P_5, and the jobs by J_1, J_2, J_3, J_4, J_5. Who is qualified for what job is given below:

Applicant	Jobs qualified for
P_1	J_1, J_2
P_2	J_3
P_3	J_2
P_4	J_3, J_4, J_5
P_5	J_4, J_6

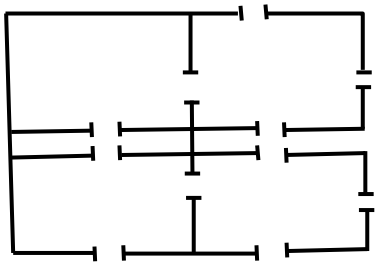
We can represent this information through graph as follows: Each of P_i s and J_j s is regarded as a vertex. If any P_i is qualified for any J_j we draw an edge between that P_i and J_j. The graph is drawn below.



The problem is: can every applicant get a job to which he/she is qualified for? In the terminology of graphs, it is equivalent to: does there exist a set M of edges such that no two edges in M have a common endpoint, and every P_i is an endpoint of some edge in M ?

You may try these exercises, now.

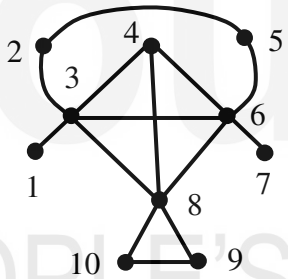
E1) In a museum an exhibition is arranged in five rooms shown as below.



The problem is to find whether it is possible to complete a visit starting from any place passing through all the doors exactly once. Model this problem by a graph.

E2) Four children A, B, C, and D are asked to give their choices about five gifts P, Q, R, S, and T. The choices of the children are as follows: A likes P, Q, and R; B likes only Q; C likes Q and S; and D likes R and T. The problem is to determine whether each child would get a gift of his/her choice or not. Draw a graph to represent this problem. Do you think a solution exist?

E3) For the graph given alongside, find



i) a clique of maximum size;

ii) an independent set of size 5;

iii) two vertices u and v such that $|N(u) \cap N(v)| = 2$;

iv) three vertices u, v and w with mutually disjoint neighbourhoods.

If you have understood what a graph is, and how it can be used to model real world problems, we can proceed to learn more about graphs. Next, we shall discuss different types of graphs.

1.3 TYPES OF GRAPHS

There are various types of graphs depending upon the number of vertices, number of edges, interconnectivity, multiplicity and the directionality of edges and their overall structure. We will discuss a few important types of graphs in this section.

Definition: A graph on n vertices is called an n-**vertex graph**. A graph on n vertices and m edges is called an (n, m)-**graph**.

A graph with only one vertex and no edge is called a **trivial graph** and a graph having no edges is called a **null graph**.

Definition: An n-vertex graph is said to be **complete** if there is an edge between every pair of its vertices. So, an n-vertex graph is complete graph if it has a clique of size n.

Thus in a null graph no vertex is adjacent to any vertex. On the other hand, in a complete graph every vertex is adjacent to all the other vertices. We shall

A graph is said to be labelled if its vertex set is specified, otherwise **unlabelled.** Sometimes, the labels of the vertices of a graph are irrelevant. In such cases, we shall omit the labels, and consider the unlabelled graph.

use the notation N_n to denote a null graph on n vertices, and K_n to denote a complete graph on n vertices. For example, N_4 and K_4 are drawn below.

N_4 K_4

Let us consider an example.

Note that $\binom{n}{r} = \frac{n!}{(n-r)!}$

Example 4: Let G be a graph such that for every distinct $u, v \in V(G)$, $N_G(u) \cup N_G(v) = V(G)$. Show that G is a complete graph.

Solution: Let us assume, if possible, that G is not complete. Then there exist two nonadjacent vertices u and v in G. This means $u \notin N_G(v)$ and $v \notin N_G(u)$. Since G has no loops, $u \notin N_G(u)$ and $v \notin N_G(v)$. Thus $\{u, v\} \notin N_G(u) \cup N_G(v)$, which implies $N_G(u) \cup N_G(v) \neq V(G)$. This is a contradiction. Hence, G is complete.

***

Next, we discuss two important statements related to simple graphs.

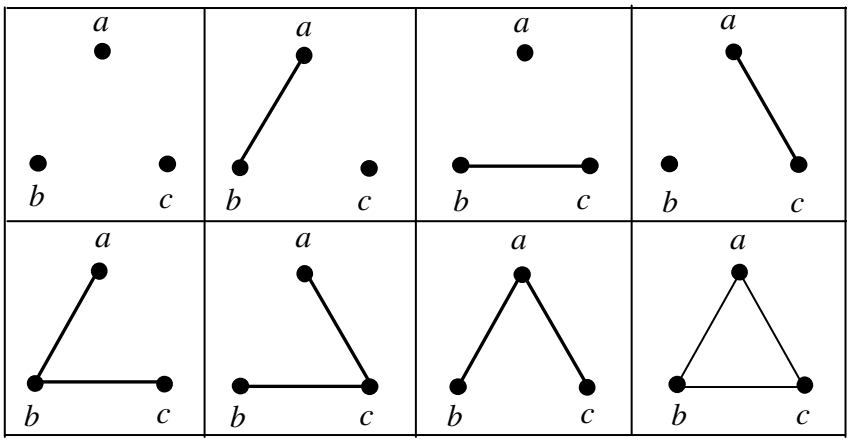
Proposition 1: An n-vertex graph has at most $\binom{n}{2}$ edges.

Proof: Note that an edge joins a pair of vertices. So, there are at most $\binom{n}{2}$ pairs of vertices, that is, the edges in a graph. ■

Proposition 2: The number of n-vertex labelled graphs is $2^{\binom{n}{2}}$.

Proof: There are a maximum of $\binom{n}{2}$ edges in a graph on n vertices. Each edge's presence or absence gives a new graph. Hence the total number of labelled graphs is $2^{\binom{n}{2}}$. ■

For example, all the $2^{\binom{3}{2}} = 8$ graphs with three vertices are drawn below.



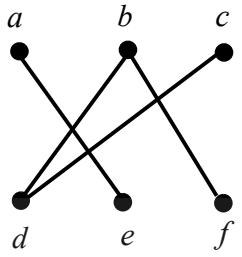
Many times graphs arise from situations where the vertices belong to two groups, for example, men-women, candidates–jobs, etc. Such situations can be modelled using bipartite graphs defined below.

Definition: A graph G is said to be **bipartite** if there exist two sets $X, Y \subseteq V(G)$ such that $X \cup Y = V(G), X \cap Y = \emptyset$ and every edge in G has one endpoint in X and the other in Y.

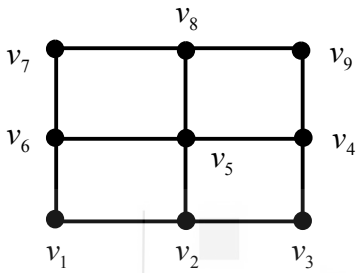
The sets X and Y are called **partite sets** of G and (X, Y) is called a **bipartition** of G.

Sometimes, a bipartite graph with bipartition (X, Y) is also called an (X, Y)-bigraph.

For example, look at the following graphs G_1, and G_2.



G_1



G_2

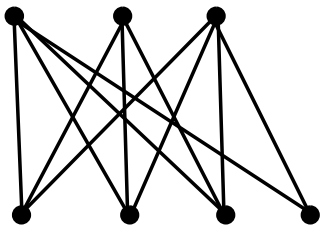
It is easy to see that $X = \{a, b, c\}$ and $Y = \{d, e, f\}$ are the partite sets of G_1, and so G_1 is a bipartite graph.

In case of G_2, let $X = \{v_1, v_3, v_5, v_7, v_9\}$ and $Y = \{v_2, v_4, v_6, v_8\}$. Then observe that every edge of G_2 has one endpoint in X and the other endpoint in Y. Also $X \cup Y = V(G_2)$ and $X \cap Y = \emptyset$. Thus (X, Y) is a bipartition of G_2. Hence, G_2 is a bipartite graph.

Finding a bipartition of a bipartite graph may not be easy always. We shall see an alternate characterisation later, in Unit 2. Now we define a special class of bipartite graphs.

Definition: A bipartite graph G is said to be **complete bipartite** if every vertex of one partite set is adjacent to every vertex of the other partite set.

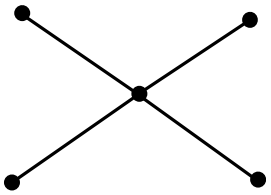
A complete bipartite graph with m and n vertices in the partite sets, respectively, is denoted by $K_{m,n}$. For example, $K_{3,4}$ is drawn below.



$K_{3,4}$

Definition: The graph $K_{1, n-1}$ is called a **star** on n vertices, or an n-**star**.

For example, the 5-star is drawn below.



The 5-star

Example 5: What is the total number of edges in $K_{m,n}$?

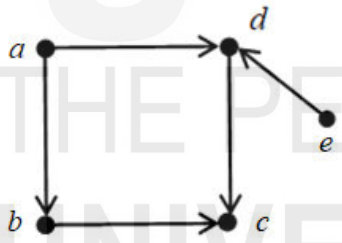
Solution: Let X and Y be the two partite sets of $K_{m,n}$ with $|X| = m,\ |Y| = n$. Since every vertex in X is adjacent to every vertex in Y, there are mn edges in $K_{m,n}$.

***

In many networks, such as in information networks, citation networks, and food web networks edges have directions that indicate the flow of information. For example, in a citation network if an article x cites article y, then y cannot cite x. This leads to the following definition.

Definition: A **directed graph** (or, in short, a **digraph)** is an ordered pair $G = (V, E),$ where V is a finite nonempty set, called the vertex set of G, and $E \subseteq V \times V$ is the edge set of G.
In a drawing of a digraph G, we shall draw an edge $e = uv$ as an arrow heading from u to v, and call u the **tail** of e, and v the **head** of e.
For example, the graph given below is a directed graph

If $e = uv$ is an edge in a directed graph, we may sometimes write $u = \text{tail}(e)$, $v = \text{head}(e)$.

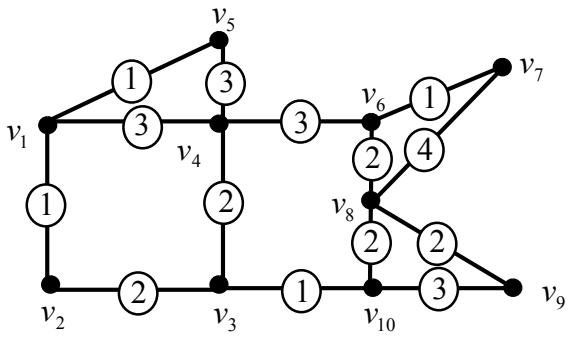


In another kind of real world networks, 'weights' are usually associated with edges.

Definition: A **weighted** graph is an ordered pair $(G, W),$ where G is a graph, and W is a weight function defined as $W : E(G) \to \mathbb{R}$ that associates each edge e of G a real number $W(e)$.

By the weights in a weighted graph, we mean the weights on the edges. However, sometimes weights can be assigned to vertices also.

For example, the following graph is weighted.

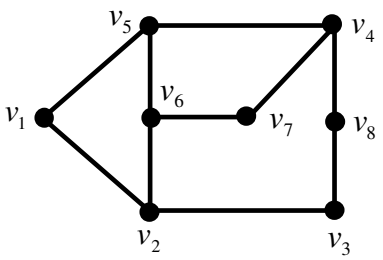


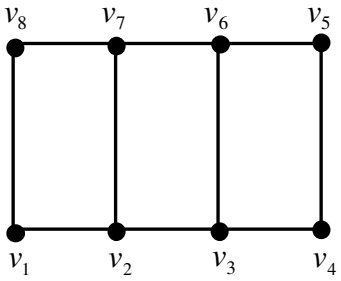
The weight of edge v_1v_2 is 1, of v_2v_3 is 2. The maximum weight is 4, corresponding to v_7v_8.

Now, you may try these exercises.

E4) If G is a bipartite graph with bipartition (X, Y), then X and Y are independent sets. True or false? Why?

E5) Find a bipartition of each of the following bipartite graphs.

i) 

ii) 

E6) There are four basic blood types: A, B, AB and O. Type O can donate to any of the four types. Type A and B can donate to AB as well as to their own types, but type AB can only donate to AB. Draw a digraph (directed multigraph) that presents this information.

E7) The distances (in kilometers) between six towns A, B, C, D, E and F of a city are as follows:

Distance between A and $B = 8$

Distance between A to $C = 5$

Distance between A to $F = 18$

Distance between B to $C = 10$

Distance between B to $D = 20$

Distance between B to $E = 18$

Distance between D to $E = 20$

Distance between E to $F = 15$

Model this situation through a weighted graph. Which town is well connected?

The directed graphs and weighted graphs arise in special applications, and we shall discuss them as and when the situation arises. Next, our aim is to discuss some more basic concepts essential for building a theory.

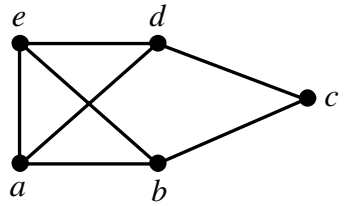
1.4 SUBGRAPH AND COMPLEMENT OF A GRAPH

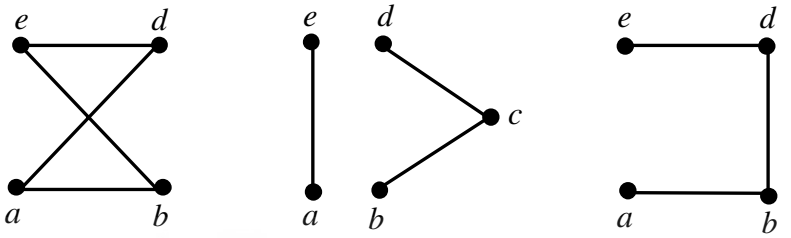
We shall, in this section, discuss what we mean by a subgraph, and what are the different types of subgraphs of a graph. We shall also discuss the notion of complement of a graph.

We call two graphs G and H equal if $V(G) = V(H)$ and $E(G) = E(H)$. Now look at the following definition.

Definition: A graph H is said to be a **subgraph** of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.

Example 6: Consider the graphs given below. Which of H_1, H_2 and H_3 are subgraphs of G?





H_1 H_2 H_3

Solution: Note that
$V(G) = \{a,b,c,d,e\}$, and $E(G) = \{ab, ad, ae, bc, be, cd, de\}$. We have $V(H_1) = \{a,b,d,e\}$ and $E(H_1) = \{ab, ad, be, de\}$. Clearly, $V(H_1) \subseteq V(G)$ and $E(H_1) \subseteq E(G)$. Hence H_1 is a subgraph of G.

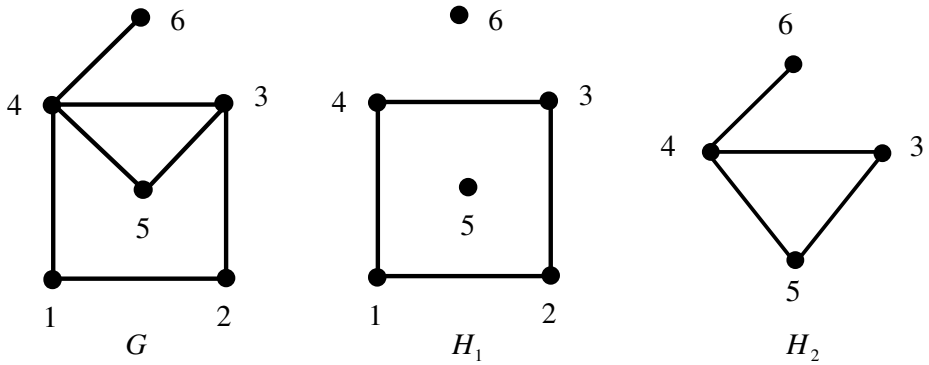
Also, note that $V(H_2) = V(G)$, and $E(H_2) = \{ae, bc, cd\} \subseteq E(G)$. Therefore, H_2 is also a subgraph of G. Now, consider the graph H_3. Edge bd of H_3 is not an edge of G. Hence, H_3 is not a subgraph of G.

***

In the example above you have seen that the subgraph H_2 contains all the vertices of G. By adding the missing edges in H_2, we can get the original graph G. Thus, in a sense H_2 "spans" G. This leads to the following definition.

Definition: If H is a subgraph of a graph G such that $V(H) = V(G)$ then H is called a **spanning subgraph** of G.

Consider the following graphs G, H_1 and H_2.



Let us check whether graphs H_1 and H_2 are spanning subgraphs of graph G, or not.

Note that $V(H_1) = V(G)$ and $E(H_1) \subseteq E(G)$. Hence H_1 is a spanning subgraph of G. However H_2 is not a spanning subgraph of G as $V(H_2) \neq V(G)$.

Example 7: Let G be the graph with $V(G) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $E(G) = \{v_1v_2, v_1v_5, v_2v_3, v_2v_4, v_2v_5, v_3v_4, v_4v_5, v_4v_6, v_5v_6\}$. Let H_1, H_2 and H_3 be graphs defined as below:

i) $V(H_1) = \{v_1, v_2, v_3\}$, $E(H_1) = \{v_1v_2, v_1v_3, v_2v_3\}$

ii) $V(H_2) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, $E(H_2) = \{v_1v_2, v_2v_3, v_4v_5\}$

iii) $V(H_3) = \{v_2, v_4, v_5, v_6\}$, $E(H_3) = \{v_2v_4, v_2v_5, v_4v_5, v_4v_6, v_5v_6\}$

Which of H_1, H_2 and H_3 are spanning subgraphs of G ?

Solution: i) Here we find that v_1v_3 is an edge in H_1, but not in G. This means H is not a subgraph of G. Therefore, H is not a spanning subgraph of G.

ii) Here $V(H_2) = V(G)$, and $E(H_2) \subseteq E(G)$. Hence, H_2 is a spanning subgraph of G.

iii) Since $V(H_3) \neq V(G)$, H_3 is not a spanning subgraph of G.

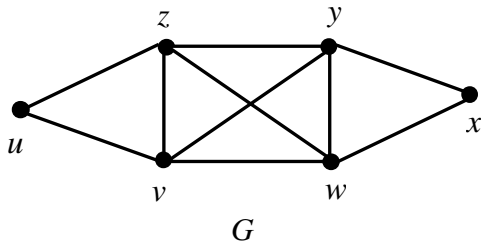
***

Although H_3, in the example above, is not a spanning subgraph of G, there is something interesting about it. It contains every edge that is present in G between any pair of its vertices. Such subgraphs are "induced" by their vertex sets. A formal definition is given below.

Definition: Let G be a graph. A subgraph H of G is said to be an **induced subgraph** if $E(H) = \{uv \in E(G) : u, v \in V(H)\}$.

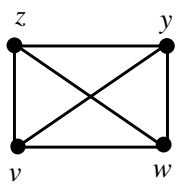
Let us consider an example.

Example 8: Look at the graph G given below:

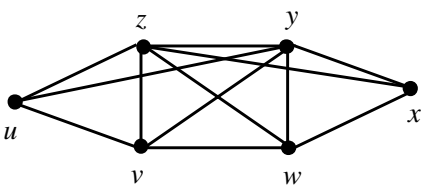


G

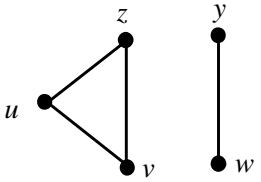
Which of the following graphs are induced subgraphs of G?



H_1



H_2



H_3

Solution: We have $V(H_1) \subseteq V(G)$. Here $E(H_1) = \{vw, wy, yz, vz, vy, wz\} \subseteq E(G)$. Thus H_1 is a subgraph of G. Further, H_1 contains all the edges present in G between every pair of the vertices of H_1. Hence, H_1 is an induced subgraph of G.

In case of H_2, note that $uy \in E(H_2)$, but $uy \notin E(G)$. So H_2 is not a subgraph of G. Hence H_2 is not an induced subgraph of G. For H_3, we have $V(H_3) \subseteq V(G)$, and $E(H_3) \subseteq E(G)$. So H_3 is a subgraph of G. Now, $y, z \in V(H_3)$ and $yz \in E(G)$, but $yz \notin E(H_3)$. Hence H_3 is not an induced subgraph of G.

***

Any subgraph of a graph is essentially obtained by 'removing' certain vertices or edges, or a collection of them. However, we must have a precise definition of what a vertex or edge removal means. If e is an edge in a graph G, then $G - e$ is the graph with $V(G-e) = V(G)$ and $E(G-e) = E(G) \setminus \{e\}$. On the other hand, when v is a vertex in a graph G, then $G - v$ is the graph with $V(G-v) = V(G) \setminus \{v\}$, and

$$E(G-v) = \{e \in E(G) : v \text{ is not an endpoint of } e\}.$$

The definition of $G - e$ and $G - v$ can be generalised as follows:

Definition: Let G be a graph and $M \subseteq E(G)$. Then $G - M$ is the graph with $V(G-M) = V(G)$, and $E(G-M) = E(G) \setminus M$.

Definition: Let G be a graph and $S \subseteq V(G)$. Then $G - S$ is the graph with $V(G-S) = V(G) \setminus S$, and

$$E(G-S) = \{e \in E(G) : e \text{ has no endpoints in } S\}.$$

Immediately, you must see that $G - M$ is a spanning subgraph of G, whereas $G - S$ is an induced subgraph.

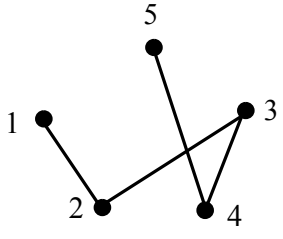
We often come across situations where two graphs have the same set of vertices but their edge sets are complementary. That is, the union of their edge sets contains all the possible edges a complete graph on the same vertex set could have. We call such graphs "complements" of each other.

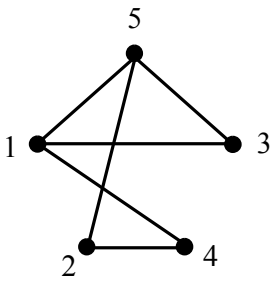
Definition: The **complement of a graph** G, denoted by $\overline{G}$, is the graph with the vertex set $V(\overline{G}) = V(G)$ and the edge set

$$E(\overline{G}) = \{uv : u, v \in V(G), uv \notin E(G)\}.$$

The definition of $E(\overline{G})$ immediately implies that $e \in E(\overline{G})$ if and only if $e \notin E(G)$.

Example 9: Consider the graph G given below:





Here $V(G)=\{1,2,3,4,5\}$. So $V(\overline{G})=\{1,2,3,4,5\}$. Now $13\notin E(G)$, so $13\in E(\overline{G})$. Likewise, we can see that the edges $14,15,24,25,35$ do not belong to G, and hence they are in $\overline{G}$. Thus $E(\overline{G})=\{13,14,15,24,25,35\}$.

***

What is the complement of the complement of a graph? To investigate, let us consider a graph G. Then we know that $V(\overline{\overline{G}})=V(\overline{G})=V(G)$. Now

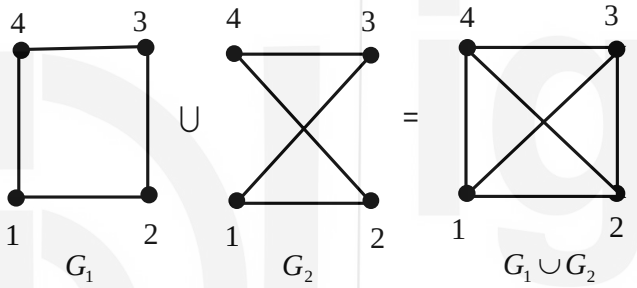
$$E(\overline{\overline{G}})=\{uv:u,v\in V(G),uv\notin E(\overline{G})\}=\{uv:u,v\in V(G),uv\in E(G)\}=E(G)$$

Thus $\overline{\overline{G}}=G$ holds for every graph G.

Let us consider the following definition

Definition: The **union** of two graphs G_1 and G_2 is the graph G with $V(G)=V(G_1)\cup V(G_2)$ and $E(G)=E(G_1)\cup E(G_2)$.
For example, look at the following graphs.

Note that we shall use the notation $G+H$ for the union of G and H, when $V(G)\cap V(H)=\emptyset$.



You may try the following exercises now.

E8) If H is a spanning induced subgraph of a graph G, then $H=G$. True or false? Justify your answer.

E9) Let G be a graph with $V(G)=\{v_1,v_2,v_3,v_4,v_5,v_6\}$ and $E(G)=\{v_1v_2,v_1v_5,v_1v_6,v_2v_3,v_2v_5,v_3v_4,v_4v_5\}$. Find $\overline{G}$. Also, draw both G and $\overline{G}$.

E10) What is the complement of K_n?

E11) Show that $\overline{K}_{m,n}=K_m+K_n$.

E12) Draw a graph G, with $n(G)=5$ and $m(G)=m(\overline{G})$.

E13) What is the relation between $m(G)$ and $m(\overline{G})$ for any graph G?

E14) Let G be a graph and $v\in V(G)$. Show that $\overline{G}-v=\overline{G-v}$.

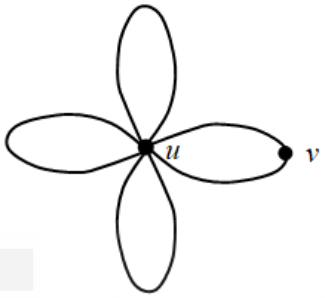
You know that the vertices in a graph have neighbourhoods. The size of the neighbourhoods of vertices of a graph can reveal a lot about the structure of the graph, as you will see in the next section.

1.5 VERTEX DEGREE

In this section, we shall define the degree of a vertex, and discuss a few related results. We shall also discuss the concepts such as even graphs and regular graphs.

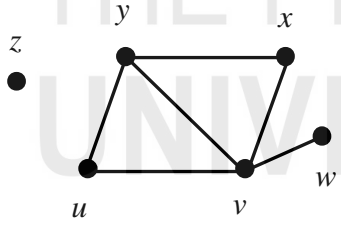
Before giving the definition of degree of a vertex, first note that if e is a loop from a vertex v to v, then e is said to be **incident on** v **twice**.

Definition: Let G be a multigraph and v a vertex in G. The **degree** of v written $d_G(v)$ or $d(v)$, is the number of edges incident on v.
For example, in the following multigraph $d(u)=8$ and $d(v)=2$.



A vertex of degree 1 is called a **leaf** (plural leaves) and a vertex of degree 0 is called an **isolated vertex**. The **maximum degree** of a graph (or multigraph) G, written as $\Delta(G)$ is the maximum of the degrees of all the vertices of G. Similarly, the **minimum degree** of a graph G, denoted by $\delta(G)$, is the minimum of all the degrees of the vertices of G.

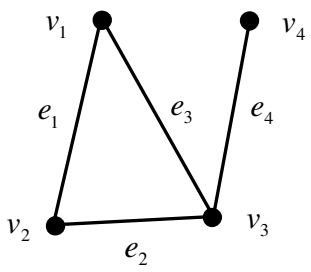
Example 10: For the following graph G, find the degree of all the vertices of G. Hence find $\delta(G)$ and $\Delta(G)$.



Solution: Since there are two edges incident on u, the degree of u is $d_G(u)=2$. Note that there are four edges incident on v. So, $d_G(v)=4$. Similarly, you can see that $d_G(w)=1$, $d_G(x)=2$ and $d_G(y)=3$. Since no edge is incident on z, $d_G(z)=0$. Thus we have $\Delta(G)=4$ and $\delta(G)=0$.

Let us discuss some more examples.

Example 11: Find the sum of all the degrees of the vertices of the following graph.



Solution: To find $d(v_1)$, we count the total number of edges, incident on v_1. Since two edges e_1 and e_3 are incident on v_1, $d(v_1)=2$. Likewise $d(v_2)=2, d(v_3)=3$ and $d(v_4)=1$. Hence the sum of the degrees is $2+2+3+1=8$.

***

From the example above, you must have observed that while counting all the vertex degrees we have counted each edge twice. For example, the edge $e_1=v_1v_2$ is counted twice, once to obtain $d(v_1)$ and secondly to obtain $d(v_2)$. Thus, while computing the sum $d(v_1)+d(v_2)+d(v_3)+d(v_4)$, we have counted each of the edges e_1, e_2, e_3 and e_4 twice. Hence $d(v_1)+d(v_2)+d(v_3)+d(v_4) = 2\times$(number of edges) $=2\times 4=8$.

This is not true only for this graph. It holds for all the graphs as stated in the following lemma.

Theorem 1 (Handshaking Lemma): If G is a graph, then $\sum_{v\in V(G)} d_G(v) = 2m(G).$

Proof: Let $v_1, v_2, ..., v_n$ be the vertices of G. Consider the degree sum

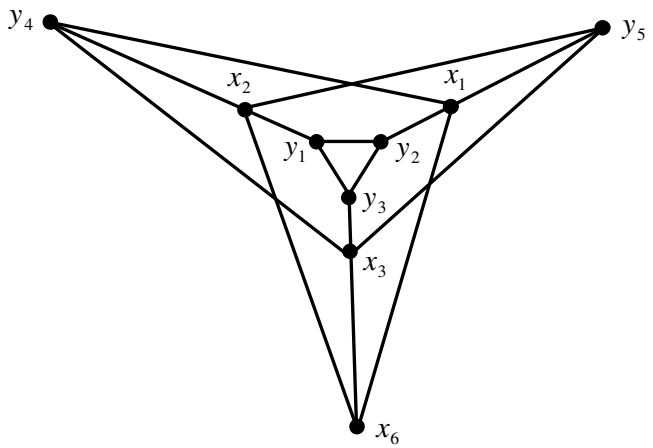
$$d_G(v_1)+d_G(v_2)+\cdots+d_G(v_n)$$

In the degree sum, every edge v_iv_j contributes 2, one for $d(v_i)$ and one for $d(v_j)$. Hence in the degree sum, all the edges are counted twice. Therefore,

$$d_G(v_1)+d_G(v_2)+\cdots+d_G(v_n)=2m(G).$$ ■

The above lemma is named so because of the following case. A party consists of n people, between whom some handshakes take place. If $d(p_i)$ denotes the total number of handshakes done by the person p_i, then the sum $\sum_{i=1}^{n} d(p_i)$ is twice of the total number of handshakes."

Thus the sum of the degrees of all the vertices of any graph is even. To verify, consider the following graph.



Clearly each $x_i, 1 \le i \le 3,$ has degree 4. All the remaining vertices $y_i, 1 \le i \le 6$ have degree 3, each. The sum of the degrees of all the vertices is 30, which is even.

In a graph G, observe that for any $v \in V(G)$, $d_G(v) = |N_G(v)|$.

Example 12: For an (n,m)- graph G, show that $\delta(G) \le \frac{2m}{n} \le \Delta(G)$.

Solution: We know that for all $v \in V(G)$, $\delta(G) \le d(v) \le \Delta(G)$.
This implies

$$\sum_{v \in V(G)} \delta(G) \le \sum_{v \in V(G)} d(v) \le \sum_{v \in V(G)} \Delta(G) \Rightarrow n\delta(G) \le 2m \le n\Delta(G)$$

$$\Rightarrow \quad \delta(G) \le \frac{2m}{n} \le \Delta(G)$$

***

Let us consider the following definition.

Definition: A vertex in a graph is said to be **even** if its degree is even, otherwise **odd**. A graph is said to be **even** if all its vertices are even.

Have you observed that the number of odd vertices in the graph above is even. This is because of the following corollary.

Corollary 1: Every graph has even number of odd vertices.

Recall that the sum of n odd numbers is even, only when n is even.

Proof: Let G be a graph. The sum of the vertex degrees of G can be calculated by taking the sum of the degrees of all even vertices and degrees of all odd vertices. That is,

$$\sum_{v \text{ is even}} d(v) + \sum_{v \text{ is odd}} d(v) = m(G)$$

Since $\sum_{v \text{ is even}} d(v)$ is even as it is the sum of even numbers, $\sum_{v \text{ is odd}} d(v)$ must also be even. Now, for each odd $v, d(v)$ is odd, and the sum $\sum_{v \text{ is odd}} d(v)$ is even. That means, the total number of odd vertices must be even. ■

Example 13: A graph has 21 edges. It has 3 vertices of degree 4 and all the other vertices of degree 3. Find the order of the graph.

Solution: Let there be n vertices in the graph. Out of these n vertices 3 are of degree 4 and $(n-3)$ vertices are of degree 3. By Handshaking Lemma

$$4 \times 3 + (n-3) \times 3 = 2 \times 21 \Rightarrow n = 13$$

Thus, G has 13 vertices.

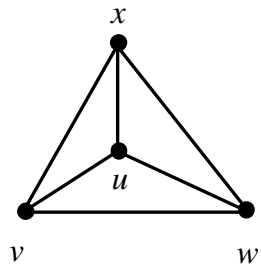
***

Clearly, if G is a regular graph, then $\Delta(G) = \delta(G)$.

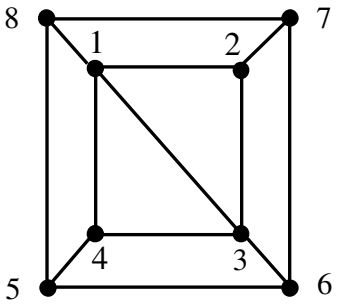
What will you call a graph whose all vertices have the same degree? Look at the following definition.

Definition: A graph is called **regular** if the degrees of all its vertices are equal. If degree of all the vertices of a graph G is $k \ge 0$ then G is called a k-**regular** graph, and k is called the **degree of regularity** of G.

Example 14: Which of the following graphs are regular?



G_1



G_2

Can you find a regular induced subgraph of G_2?, regular spanning subgraph of G_2?

Solution: Note that the degree of each vertex of G_1 is 3, so G_1 is a 3-regular graph. In G_2, all the vertices do not have the same degree. For example, the vertex 1 has degree 4, but the vertex 5 has degree 3. Hence, G_2 is not regular.

***

You can see that a complete graph K_n is $(n-1)$-regular, as

$$\Delta(K_n) = \delta(K_n) = n-1.$$

Example 15: Find the size of an n-vertex k-regular graph.

Solution: Let graph G be k-regular, i.e., $d(v) = k$ for all $v \in V(G)$.
By Handshaking lemma,

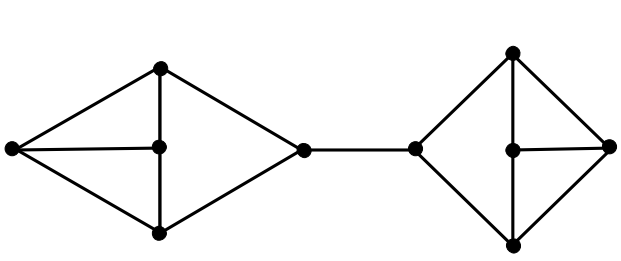
$$\sum_{v \in G} d(v) = 2m(G) \Rightarrow kn = 2m(G) \Rightarrow m(G) = \frac{kn}{2}$$

***

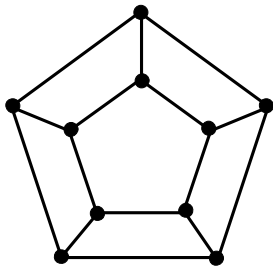
The example given above tells us what kinds of regular graphs are possible.
For a k-regular graph on n vertices, **at least one of k and n must be even.**

Example 16: Construct two 3-regular graphs on 10 vertices.

Solution: Two 3-regular graphs with 10 vertices are drawn below.



(a)



(b)

***

Now, to check your understanding try the following exercises.

E15) Show that the Handshaking Lemma holds when G is a multigraph.

E16) Let G be a graph with $n(G) \geq 2$. Show that there are at least two vertices in G with equal degrees.

E17) Consider a graph G with 12 edges. If G has 6 vertices of degree 3 and the other vertices have degree less than 3, what is the minimum order of G ?

E18) Consider a graph G of order 13 such that each vertex has degree 7 or 8. Prove that there are at least 8 vertices of degree 7, or 7 vertices of degree 8.

E19) Prove or disprove: There exists a graph G such that $n(G)=5,\ \delta(G)=1$ and G contains two vertices of degree 4.

E20) Does there exist a 3-regular graph on 5 vertices? Justify.

E21) Consider the following statement:
"If G is a regular graph, then $\overline{G}$ is regular."
If this statement is true, prove it. Otherwise, give a counterexample.

In any network, reachability from one vertex to another is of utmost importance. For example, in a housing colony it is desirable that every house is connected by a road to address the issue of reachability. In the next section, we shall discuss the notions such as walks, trails, paths, and connected graphs.

1.6 CONNECTED GRAPHS

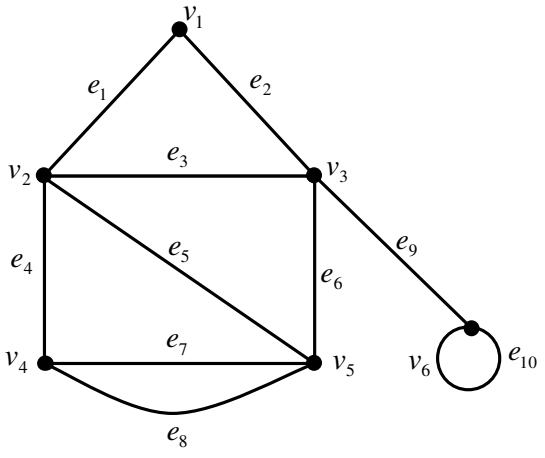
In this section, we shall discuss about connected graphs. Let us first define some important terms.

Definition: Let G be a multigraph, and $u,v\in V(G)$. A (u,v) - **walk** is a finite alternating sequence of vertices and edges $(w_0, e_1, w_1, e_2, w_2, e_3, ..., w_{k-1}, e_k, w_k)$ such that $w_0=u$, $w_k=v$, and for each $1\le i\le k$, the edge e_i has endpoints w_{i-1} and w_i.

Note that a walk always begins and ends at a vertex.

The **length** of a walk W, denoted as $\text{length}(W)$, is the number of edges in W, with repeated edges counted as many times as they occur. A (u,v) - walk is said to be **closed** if $u=v$, otherwise **open**.

For instance, the sequence $(v_1,e_1,v_2,e_3,v_3,e_6,v_5,e_8,v_4,e_7,v_5,e_5,v_2)$, in the multigraph given below, is a (v_1,v_2) - walk. Of course, (v_1,e_1,v_2) is also a (v_1,v_2) - walk, with smaller length.



Can you find a (v_1, v_5) - walk of length 5, with no edge repeated, in the graph above? Just try.

Definition: A walk with no repeated edges is called a **trail.** A closed trail is called a **circuit**. A walk with no repeated vertex is called a **path.** A closed path is called a **cycle**.

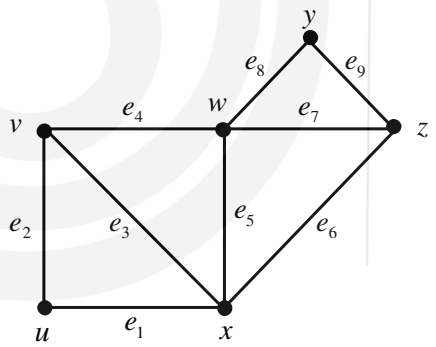
For example, in the multigraph given above, $(v_2, e_3, v_3, e_6, v_5, e_5, v_2, e_4, v_4)$ is a (v_2, v_4)-trail, and $(v_2, e_5, v_5, e_7, v_4, e_8, v_5, e_6, v_3, e_3, v_2)$ is a closed trail, i.e., a circuit. Likewise, you can see that $(v_1, e_2, v_3, e_9, v_6)$ is a (v_1, v_6)-path, and $(v_1, e_2, v_3, e_6, v_5, e_7, v_4, e_4, v_2, e_1, v_1)$ is a cycle.

Walks, trails and paths provide different ways of traversing a graph. Suppose from our home we walk to reach our friend's home. All the three types of traversing are possible. It could be (as if a drunkard or a senseless person does) we keep on travelling through the same road and repeating the same spots or junctions. Usually we avoid repetition and use an optimal path (may be unknowingly).

Now we shall discuss walks, trails and paths in graphs.

We denote a cycle on n vertices by C_n and a path on n vertices by P_n. A cycle with 3 vertices, C_3, is often called a **triangle**.

Note that in a graph there is at most one edge between any two vertices. So in a graph trails and paths can be represented by vertex sequences. For example, consider the following graph.



Look at the vertices u and y. There are many ways to go from u to y. One can take the walk (u, x, v, w, x, z, w, y) of length 7. One can also go via a shorter walk (u, x, v, w, x, z, y) of length 6. There is an even shorter walk namely (u, x, w, z, y). What do you think about the walk (u, x, w, y)? It is one of the shortest paths from u to y. The point we emphasise is that whenever a (u, v)-walk contains repeated edges or vertices, we can always cut short the repeated edges and vertices to get a (u, v)-path.

Let us look at the following lemma, in this regard.

Lemma 1: In a graph G for distinct vertices u and v, every (u, v) - walk contains a (u, v)-path

Proof: Let W be a (u, v)-walk in G. Let P be a shortest (u, v) - walk in W. We have to show that P is a (u, v) - path. Assume, if possible, that P is not a

path. This means there is a vertex w which is repeated in P. Then P contain a subsequence $(w, e_1, w_1, \ldots e_k, w)$. Removing $(e_1, w_1, \ldots, e_k, w)$ from P leaves a (u,v) - walk shorter than P. This is a contradiction. Hence P must be a (u,v) - path.

Does Lemma 1 hold when $u = v$?

■

It is natural to *concatenate* two walks ending at a common vertex. Let $W = (u = x_0, x_1, ..., x_k = v)$ be a (u,v) -walk, and $W' = (v = y_0, y_1, ..., y_l = w)$ a (v,w) -walk. Then we define $W + W'$ to be the walk

$$W + W' = (u = x_0, x_1, ..., x_k = v = y_0, y_1, ..., y_l = w)$$

The length of $W + W'$ is

$$\text{length}(W + W') = \text{length}(W) + \text{length}(W')$$

Consider the following example which shows the concatenation of two paths.

Example 17: Let G be a graph, and $u, v, w \in V(G)$. If P is a (u,v) -path, and P' is a (v,w) -path such that P and P' have no common vertex other than v, then $P + P'$ is also a path, and

$$\text{length}(P + P') = \text{length}(P) + \text{length}(P').$$

If P and P' have exactly two common vertices, that is, their endpoints, then $P + P'$ is a cycle.

***

Presence or absence of cycles of a particular length is often useful in studying the structure of graphs. In this connection, we have the following definition.

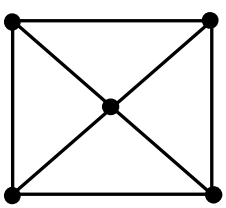
Definition: Let G be a graph with at least one cycle. The **girth** of G is the length of a shortest cycle in G. If G has no cycles, then we say G has infinite girth.

For instance, the girth of C_n is n, whereas the girth of P_n is infinite. Similarly, you can see that K_1 and K_2 have infinite girths, whereas for $n \geq 3$, K_n has girth 3. A triangle-free graph has girth at least 4. Can you think of a graph with girth 4? What about $K_{2,2}$? Just try to draw a graph with girth 5, apart from C_5.

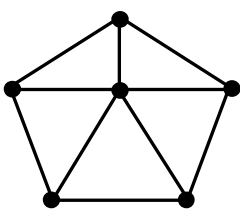
Now consider the following definition.

Definition: An n-vertex **wheel**, denoted as W_n, is a graph which contains a cycle C_{n-1} and a vertex adjacent to each of the vertices of C_{n-1}.

For example, the 5-vertex wheel W_5 and W_6 are drawn below. Note that W_5 has girth 3. What is the girth of W_6?



W_5



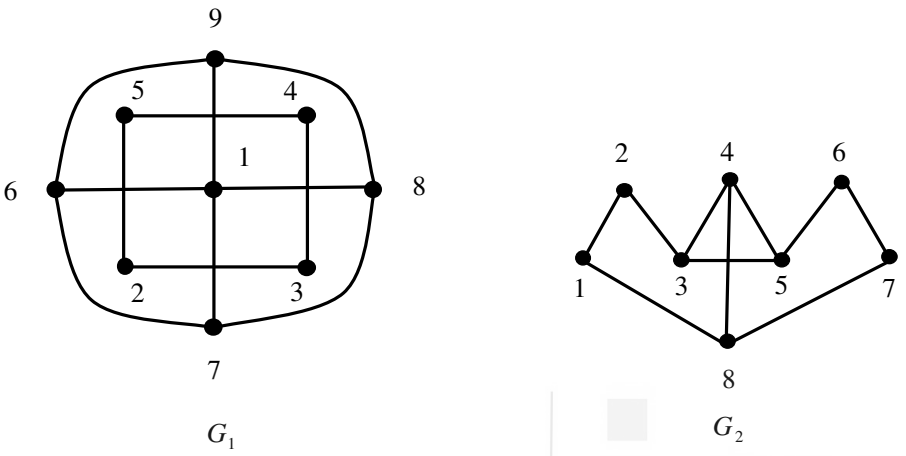
W_6

In a graph, to reach from one vertex to another, all we need to know is a path between them. This leads to the following definition.

Definition: A graph G is said to be **connected** if for any $u, v \in V(G)$, there is a (u,v)-path in G. If a graph is not connected then it is called a **disconnected** graph.

Consider, for example, the graphs G_1 and G_2 given below:

What is the girth of W_n?

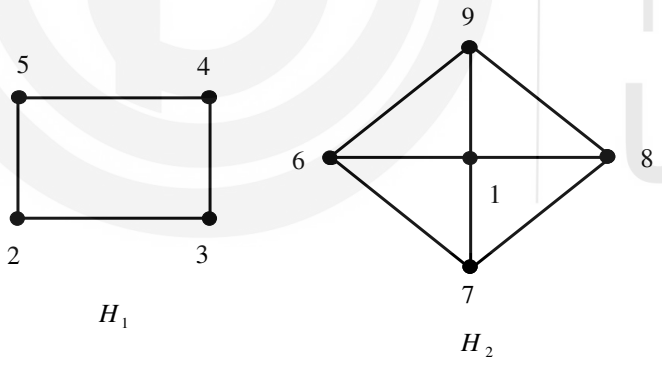


G_1 G_2

Note that in G_1, there is no path from 1 to 4. Hence G_1 is disconnected. On the other hand, in G_2 we can find a path between every pair of vertices. Hence G_2 is connected.

$c(G)$ = no. of components in G

Definition: A **component** of graph G is a subgraph of G that is connected and is not contained in any other connected subgraph of G.
For example, the graph G_1 given above can be redrawn as follows:

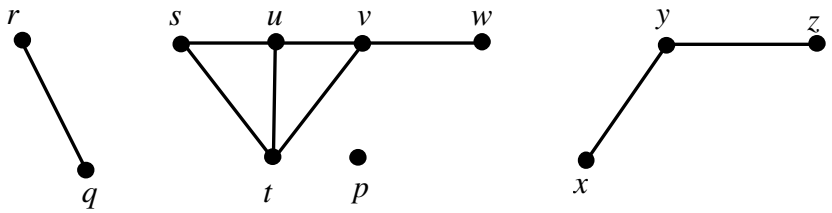


H_1 H_2

Note than N_n has n components, whereas K_n has just one component.

Note that both H_1 and H_2 are connected. Also, none of them is contained in another connected subgraph. Thus, H_1 and H_2 are the components of G_1.

Example 18: In the following graph G, there are four components whose vertex sets are $\{p\},\{q,r\},\{s,t,u,v,w\}$ and $\{x,y,z\}$.



***

We know that N_n contains n components. If you add one edge to N_n one component is reduced. That is, the resulting graph has $n-1$ components. Further, add one edge. How many components does the resulting graph have, now? At least $n-2$, right? Now think of a graph G with n vertices and m edges. How many components can G have? The answer is **at least** $n-m$**,** as stated below.

Proposition 3: An (n,m) - graph has at least $n-m$ components.
Now, we will discuss a very important theorem.

Theorem 2: A disconnected graph G with n vertices and k components can have at most $\binom{n-k+1}{2}$ edges.

Proof: Let n_i be the number of vertices in the i^{th} component of G . Then

$$n_1 + n_2 + .. + n_k = \sum_{i=1}^{k} n_i = n.$$

The maximum number of edges in the i^{th} component is $\binom{n_i}{2}$. So, the maximum number of edges in G is

$$\sum_{i=1}^{k}\binom{n_i}{2} = \frac{1}{2}\sum_{i=1}^{k} n_i(n_i-1) = \frac{1}{2}\left(\sum_{i=1}^{k} n_i^2 - \sum_{i=1}^{k} n_i\right) = \frac{1}{2}\left(\sum_{i=1}^{k} n_i^2 - n\right)$$

$$= \frac{1}{2}\left[\sum_{i=1}^{k}(n_i - 1 + 1)^2 - n\right]$$

$$= \frac{1}{2}\left[\left(\sum_{i=1}^{k}(n_i-1)^2 + 2\sum_{i=1}^{k}(n_i-1) + \sum_{i=1}^{k}1\right) - n\right]$$

$$= \frac{1}{2}\left[\sum_{i=1}^{k}(n_i-1)^2 + n - k\right] \le \frac{1}{2}\left[\left(\sum_{i=1}^{k}(n_i-1)\right)^2 + n - k\right] \text{ (Why?)}$$

$$= \frac{1}{2}\left[(n-k)^2 + n - k\right]$$

$$= \frac{1}{2}(n-k)(n-k+1) = \binom{n-k+1}{2}.$$

■

Let us consider some examples.

Example 19: If u,v are the only odd vertices in a graph G , then show that G contains a (u,v) - path.

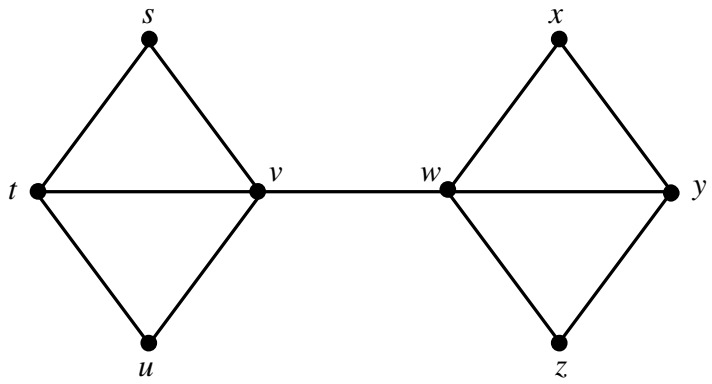
Solution: If u and v are the only vertices of odd degree in a graph G then u and v have to be in the same component of G . Because, if u and v belong to two distinct components say, H_1 and H_2 , respectively, then H_1 being a graph must contain an odd vertex w , different from u. This is a contradiction. Hence G contains a (u,v) -path.

***

Now, you should try the following exercises.

E22) Let uv be an edge in a graph G. Show that uv belongs to at least $d(u)+d(v)-n(G)$ triangles in G.

E23) In the following graph



i) find two paths from s to z;

ii) find a trail of length 7 which is not a path;

iii) are there any cycles containing both s and z?

E24) What is the size of W_n?

E25) Find the value of n for which the following graphs are regular.

i) P_n ii) C_n iii) W_n

E26) If G is a connected graph, then $\overline{G}$ is also connected. Prove or give a counterexample.

E27) Check whether the following statements are true or false. Give reasons for your answers.

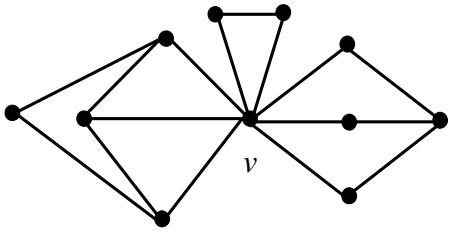
i) Every disconnected graph has an isolated vertex.

ii) An $(n, n-1)$-graph has exactly 1 component.

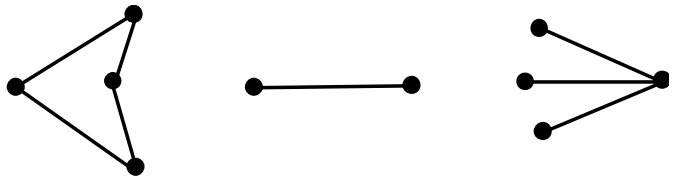
For many applications it is necessary to know more about a graph than just whether or not it is connected. For example, in telecommunications networks there are usually several different paths between a given pair of subscribers (vertices). In such a situation, it is important to know how many links (edges) can be blocked or broken without preventing a call being made between the two subscribers. In order to answer this and similar questions, in the next section we shall investigate connected graphs a little further.

1.7 CUT-VERTEX AND CUT-EDGE

In this section, we shall discuss the notions of cut-vertex and cut-edge. These notions are essential to study many important concepts of graph theory. Consider the graph G given below:



After deleting v from G, how many components are left in the remaining graph $G-v$? We have $c(G-v)=3$, as you can see below.



$G-v$

Thus, the removal of v cuts G into 3 components. Such a vertex is called a "cut- vertex". Let us look at the formal definition.

Note that to have cut-vertices a graph need not be connected.

Definition: Let G be a multigraph and $v \in V(G)$. Then v is called a **cut-vertex** of G if $c(G-v) > c(G)$.

In the case given above, $c(G-v)=3>1=c(G)$. Hence, v is a cut vertex of G. Let us consider one more example.

Example 20: Find the cut-vertices of the following graph.

G

Solution: Note that $c(G)=1$. The vertices u and y cannot be cut-vertices, because $c(G-u)=c(G-y)=1$. Likewise, w and x are not cut-vertices. For v note that $c(G-v)=2>c(G)$. Hence, v is a cut-vertex.

***

Example 21: Suppose v is a cut-vertex of a graph G. Prove that $\overline{G}-v$ is connected.

Solution: Let $u, w \in V(G)$. First assume that u and w belong to two different components of $G-v$. So, in $\overline{G}-v$ there is an edge joining u and w. (Why?) Now let u, and w be in the same component of $G-v$. Then in $\overline{G}-v$, u and w have a common neighbour lying in another component of $G-v$. Thus, in either case there is an (u,w)-path in $\overline{G}-v$. Hence $\overline{G}-v$ is connected.

***

Now, we shall discuss the characterisation of cut-vertices.

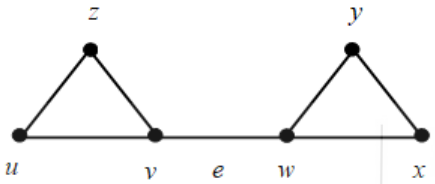
Theorem 4: Let G be a connected graph, and $v \in V(G)$. Then v is a cut-vertex of G if and only if there exist vertices $u, w \in V(G) \setminus \{v\}$ such that every (u,w)-path in G passes through v.

Proof: Let G be a connected graph and let v be a cut-vertex of G. Then $G-v$ is disconnected. Let $G_1, G_2, ..., G_k$ be the components of $G-v$. Let

$U = V(G_1)$ and $W = \bigcup_{i=2}^{k} V(G_i)$. Also, let $u \in U$ and $w \in W$. Obviously, $w \in V(G_i)$ for some $i \neq 1$. If there is a (u,w)-path P in G not passing through v, then P connects u and w in $G - v$ also. (Why?) Therefore $G_1 \cup G_i$ is a single component in $G - v$, contradicting our assumption. Thus every (u,w)-path in G passes through v.

Conversely, let there be vertices $u, w \in V((G) \setminus \{v\}$ such that every (u,w)-path in G passes through v. Then there is no (u,w)-path in $G - v$. Therefore u and w belong to different components of $G - v$. Thus $G - v$ is disconnected and v is a cut-vertex of G. ■

Now let us see how an edge removal affects the connectedness of a graph. Consider the graph G given below.



G

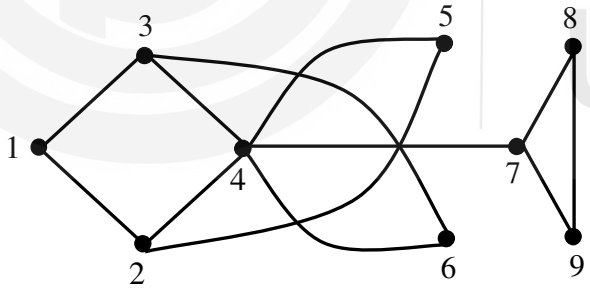
Look at the edge $e = vw$ in G. Removal of e from G leaves two components. That is, $G - e$ has more components than G. Such an edge is called a "cut-edge" as defined below.

How many components can an edge removal create?

Definition: Let G be a multigraph. An edge e of G is said to be a **cut-edge** if $c(G-e) > c(G)$.

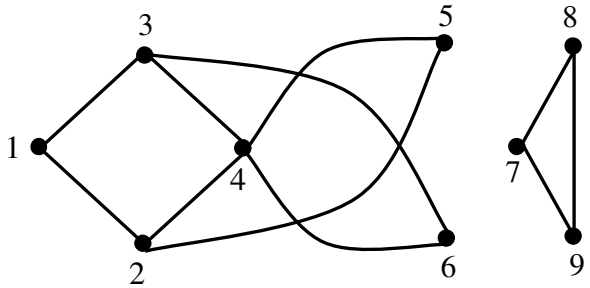
Let us consider the following example.

Example 22: Find a cut-edge in the following graph.



G

Solution: Look at the edge 47. The graph $G - 47$ is drawn below.



Clearly $G - 47$ has more components than G. Hence 47 is a cut-edge of G.

***

Next, we characterise cut-edges in terms of cycles.

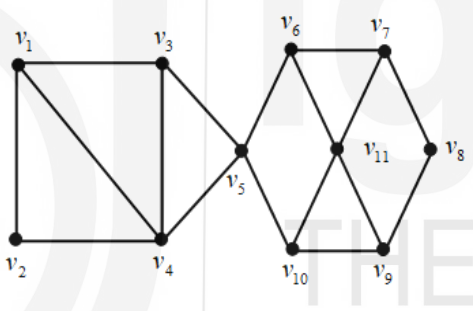
Theorem 5: Let G be a connected graph. Then an edge of G is a cut-edge of G iff it belongs to no cycle of G.

Proof: First we shall prove that if e is a cut-edge of G, then e belongs to no cycle of G. So, let us assume that $e = xy$ is a cut-edge of G. Then $G - e$ is disconnected. So, x and y must lie in different components of $G - e$. Now, if e lies on a cycle in G, then there must be an (x, y)-path in G not containing e, i.e., an (x, y)-path in $G - e$. But this not possible because x and y are in the different components of $G - e$. Hence e belongs to no cycle of G.

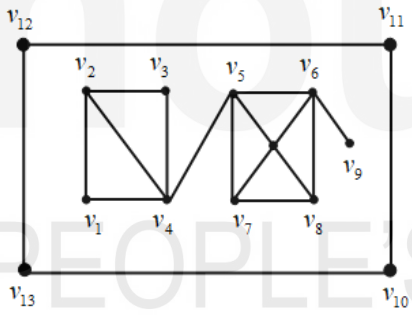
Now let us prove that if e belongs to no cycle of G, then e is a cut-edge of G. So, we assume that $e = xy$ lies in no cycle of G. This means, (x, e, y) is the only (x, y)-path in G. Hence $G - e$ is disconnected. Consequently, e is a cut-edge of G. ■

Now try the following exercises.

E28) Identify the cut-vertices and cut-edges in the following graphs.



i)



ii)

E29) Determine whether the statements below are true or false:

i) If a graph is connected then some vertex is adjacent to all other vertices.

ii) Every leaf is a cut-vertex.

iii) Every edge incident to a leaf is a cut-edge.

Let us now summarise what we have covered in this unit.

1.8 SUMMARY

In this unit, we have covered the following points:

1. Modelling of real world problems through graphs.
2. Basic terminology from graph theory, such as definitions of a graph, subgraph, and different types of graphs including directed, and weighted graphs.
3. The notions of spanning and induced subgraphs.

4. Complement of a graph.

5. Independent sets, and cliques.
6. Complete graphs, bipartite graphs and complete bipartite graphs.

7. Concepts of neighbourhood and degree of a vertex, and related concepts such as even, and odd vertices, regular graphs.

8. Some results related to vertex degree sums such as handshaking lemma.

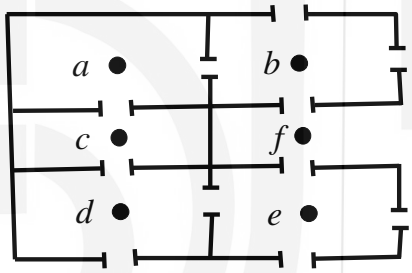
9. The concepts walk, trail, circuit, path, cycle in a given graph.

10. The concepts of connected and disconnected graphs.

11. The concepts of cut-vertex and cut-edge in a graph, and some characterisations of cut-vertices and cut-edges.

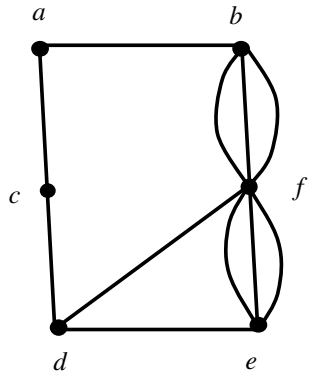
1.9 SOLUTIONS/ ANSWERS

E1) Let us denote the five rooms by the letters $a, b, c, d, e,$ and the open place by f as shown in the figure given below.

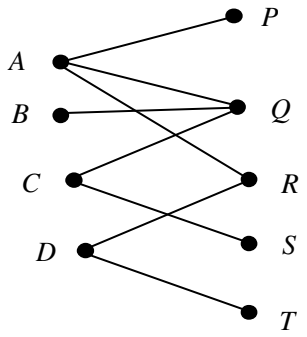


Corresponding to each room we draw a vertex as shown below.

Note that there is a door between a and b, and between a and c, to indicate them we have joined a to b, and a to c by lines. Between b and f there are 3 doors, so we have drawn 3 lines between b and f. There are also 3 doors between e and f, and hence 3 lines between them. The room d is connected to each of c and e by one door. So there is a line between c and d, and between d and e. One door at d connects it to f, and hence one line is drawn to join d and f. Thus the following graph models the given problem.



E2) We have drawn the children as the vertices A, B, C, D and the chocolates as the vertices P, Q, R, S, T. Each edge between a child and a chocolate indicates that the child likes the chocolate. The graph modelling this situation is drawn below.



Existence of a solution to this problem means that each child gets a chocolates of his/her choice. Let us see whether a solution exists or not. Note that B likes Q, so B must get Q. The child C likes Q and S. Since Q is given to B, C must get S. Now A still has two choices namely, P and R. Let us give P to A. Then we can give one of R and T to D. Thus this problem has a solution.

E3) i) We can see that $\{3,4,6,8\}$ is a clique of maximum size.

ii) $\{1,2,4,7,9\}$ is an independent set of size 5.

iii) We find that
$N(3) \cap N(6) = \{1,2,4,6,8\} \cap \{3,4,5,7,8\} = \{4,8\}$.
So $|N(3) \cap N(6)| = 2$.

iv) Look at the vertices $1, 7$ and 9. Here
$N(1) \cap N(7) = N(1) \cap N(9) = N(7) \cap N(9) = \emptyset$.

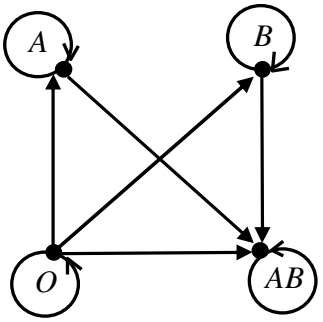
E4) True. Because no edge has its endpoints in the same partite set.

E5) i) Let $v_1 \in X$, Then $v_2 \in Y$ as $v_1 v_2$ is an edge of the graph. Similarly, $v_6 \in X$ and $v_5 \in Y$. This way, we get $X = \{v_1, v_3, v_4, v_5\}$ and $Y = \{v_2, v_5, v_7, v_8\}$.

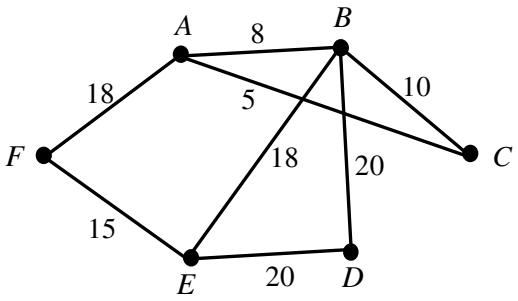
ii) Let $v_1 \in X$. Then $v_2, v_8 \in Y$, and $v_3, v_7 \in X$. Then $v_4, v_6 \in Y$ and $v_5 \in X$. So, we get

$X = \{v_1, v_3, v_5, v_7\}$ and $Y = \{v_2, v_4, v_6, v_8\}$

E6) Let us draw the vertices as A, B, AB and O as shown below. An edge uv indicates that the blood type u can donate to the blood type v.



E7) The required weighted graph is drawn below.

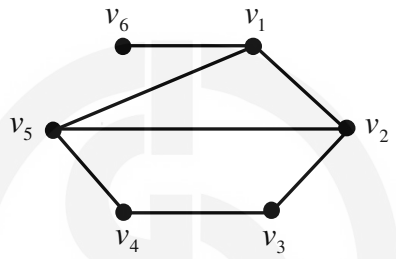


We can see that Town B is well connected.

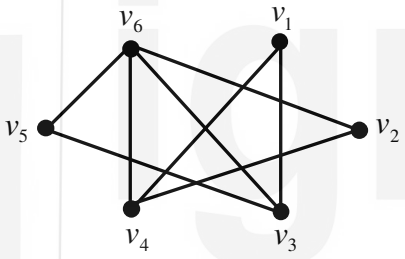
E8) True. Since H is spanning, $V(H)=V(G)$. Since H is induced $E(H)=E(G)$. Hence, $H=G$.

E9) By definition $V(\overline{G})=\{v_1,v_2,v_3,v_4,v_5,v_6\}$. Since $\overline{G}$ contains precisely those edges which are missing in G we have

$$E(\overline{G})=\{v_1v_3,v_1v_4,v_2v_4,v_2v_6,v_3v_5,v_3v_6,v_4v_6,v_5v_6\}.$$



G

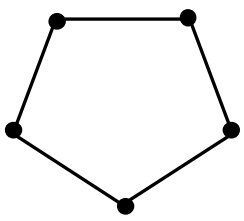


$\overline{G}$

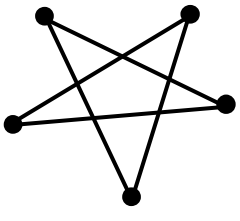
E10) N_n.

E11) $K_{m,n}$ has two independent sets X and Y of sizes m and n, respectively. Each of these independent sets becomes a clique in $\overline{K}_{m,n}$. In $K_{m,n}$, each vertex in X is adjacent to all the vertex in Y. Hence in $\overline{K}_{m,n}$ there is no edge between X and Y. Thus $\overline{K}_{m,n}=K_m\cup K_n=K_m+K_n$.

E12) One such graph is drawn below.



C_5



$\overline{C}_5$

E13) We know that $\binom{n(G)}{2}$ is the maximum number of edges in any graph G.

So $m(\overline{G})=\binom{n(G)}{2}-m(G)$.

E14) It is easy to see that $V(\overline{G}-v)=V(\overline{G-v})$. Now

$$\begin{aligned} xy\in E(\overline{G}-v) &\Leftrightarrow xy\in E(\overline{G}), \text{ and } x,y\in V(\overline{G}-v)\\ &\Leftrightarrow xy\notin E(G), \text{ and } x,y\in V(G-v)\\ &\Leftrightarrow xy\notin E(G-v) \qquad \text{(Why?)}\\ &\Leftrightarrow xy\in E(\overline{G-v}). \end{aligned}$$

Hence $E(\overline{G}-v)=E(\overline{G-v})$. Therefore, $\overline{G}-v=\overline{G-v}$ for any vertex v of G.

E15) Note that every loop or parallel edge contributes exactly 2 towards the degree sum. Hence, the Handshaking Lemma holds for multigraphs.

E16) Let $n(G)=n$. If G has two or more isolated vertices, the result holds. If G has exactly one isolated vertex, then the remaining $n-1$ vertices have degrees in $\{1,2,\ldots,n-2\}$. Hence by the Pigeonhole Principle at least two vertices have the same degree. Finally, if G has no isolated vertices, then there are n vertices and $n-1$ degrees $1,2,3,\ldots,n-1$. Again by the Pegionhole Principle, there are two vertices with equal degrees.

E17) We have $m(G)=12$. Since 6 vertices have degree 3, the remaining $n-6$ vertices have degree less than 3. Applying the Handshaking Lemma, we get

$$6\times 3+\sum_{i=1}^{n-6} d(v_i)=2\times 12 \Rightarrow 6=\sum_{i=1}^{n-6} d(v_i)$$

$$\Rightarrow 6\le 3(n-6) \Rightarrow n\ge 8$$

Thus, the graph has minimum order 8.

E18) Let n_1 and n_2 be the number of vertices of degree 7, and 8, respectively. Then $n_1+n_2=13$. Now, if possible, let $n_1\le 7$ and $n_2\le 6$. Then $n_1=7$ gives $n_2=6$, and the degree sum becomes $7\times 7+6\times 8$, which is odd. Hence, $n_1=7$ is not possible. So, we have $n_1\le 6$ and $n_2\le 6$. But then $n_1+n_2\le 12$, which is a contradiction. Therefore, either $n_1\ge 8$ or $n_2\ge 7$.

E19) Assume, if possible, that G is such a graph. Let u and v be the two vertices of degree 4 in G. Then u is adjacent to all the other vertices of G, and so is the case with v. This means, every vertex has degree at least 2, which contradicts the fact that $\delta(G)=1$. Hence, no such graph exists.

E20) No. A 3-regular graph on 5 vertices would have degree sum 15, which is odd. Hence no such graph exists.

E21) The statement is true, the proof of which is as follows. Since G is regular, there is some nonnegative integer k such that $d_G(v)=k$ for all $v\in V(G)$. Then $d_{\overline{G}}(v)=n(G)-k$ for all $v\in V(\overline{G})$. Since $n(G)-k\ge 0$, $\overline{G}$ is regular.

E22) Observe that uv belongs to precisely as many triangles as there are common neighbours of u and v. Now

$$|N(u)\cup N(v)| = |N(u)| + |N(v)| - |N(u)\cap N(v)|,$$

which implies

$$|N(u)\cap N(v)| = |N(u)| + |N(v)| - |N(u)\cup N(v)|$$
$$= d(u) + d(v) - |N(u)\cup N(v)|$$

Note that $|N(u)\cup N(v)| \le n(G)$. So

$|N(u)\cap N(v)| \ge d(u) + d(v) - n(G)$, as desired.

E23) i) Two (s,z)-paths are (s,v,w,z) and (s,t,u,v,w,z).

ii) One such trail is (s,v,t,u,v,w,y,z). Since the vertex v is repeated, it is not a path.

iii) Note that in any cycle containing s and z, the edge vw must occur twice, which is impossible. Hence, there is no cycle containing both s and z.

E24) W_n contains a cycle C_{n-1} which has $n-1$ edges. Also, W_n has a vertex adjacent to each of the vertices of C_{n-1} which gives additionally $n-1$ edges. There are no other edges. Thus the size of W_n is $2n-2$.

E25) i) Note that P_n has two vertices of degree 1 for all n. Then P_n can be regular only when all its vertices have degree 1. This is possible only when $n = 2$.

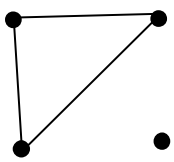
ii) Note that whatever n is, every vertex of C_n has degree 2. Thus C_n is regular for all n.

iii) W_n has a cycle C_{n-1}. Every vertex of W that lies on C_{n-1} has degree 3. (Why?) So, the remaining vertex which is adjacent to every vertex of C_{n-1} must have degree 3. This gives $n-1=3$, i.e., $n=4$.

E26) If G is connected, then it is not necessary that $\overline{G}$ is connected. For example, K_n is connected but $\overline{K}_n = N_n$ which is disconnected.

E27) i) False. The graph given below is disconnected but has no isolated vertex.

ii) False. For a counterexample look at the graph given below.

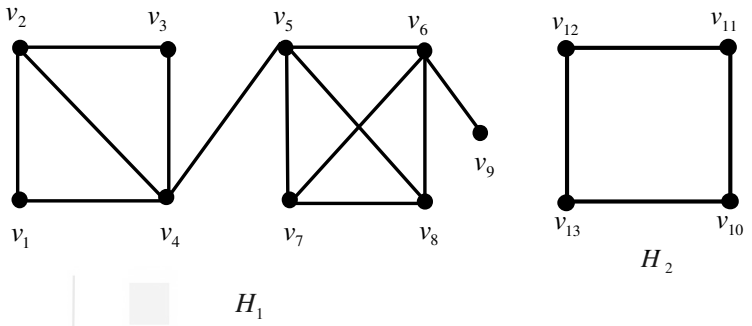


E28) i) The given graph has only one component. Note that the graph obtained after removal of any of the vertices v_1, v_2, v_3 or v_4 is

connected. Hence, v_1, v_2, v_3 and v_4 are not cut-vertices. Similarly, the vertices $v_6, v_7, v_8, v_9, v_{10}$, and v_{11} are not cut-vertices. However, the graph obtained by deleting v_5 has two components induced by $\{v_1, v_2, v_3, v_4\}$ and $\{v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$. Hence, v_5 is a cut-vertex.

Now observe that each edge belongs to some cycle. Hence, by Theorem 5, the graph has no cut-edges.

ii) Let G be the given graph. It is easy to see that G is disconnected. Its two components H_1 and H_2 are shown below



H_1

We can see that removal of any of the vertices v_1, v_2 or v_3 leaves a graph with 2 components. That is, $c(G - v_1) = c(G - v_2) = c(G - v_3) = c(G)$. So, none of v_1, v_2 and v_3 is a cut-vertex of G. Likewise, we can see that none of the vertices v_7, v_8 and v_9 is a cut-vertex. Now consider the removal of v_4, v_5 or v_6. For instance, for v_4, we find that $c(G - v_4) = 3 > c(G)$. Thus v_4 is a cut-vertex. Similarly, v_5 and v_6 are cut-vertices.

Finally let us note that the removal of any of the vertices v_{10}, v_{11}, v_{12} or v_{13} does not increase the components. Hence v_{10}, v_{11}, v_{12} and v_{13} are not cut-vertices.

Let us now find the cut-edges.

We can see that every edge, except v_4v_5 and v_6v_9, of the component H_1 lies in a cycle. Hence v_4v_5 and v_6v_9 are the only cut-edges of H_1. Also every edge of the component H_2 lies in a cycle. So, H_2 has no cut-edges. Therefore, the only cut-edges of G are v_4v_5 and v_6v_9.

E29) i) False. For example, C_4 is connected, but it has no vertex adjacent to all the other vertices.

ii) False. Without loss of generality, assume that G is a connected graph. Let v be a leaf vertex in G. Take $u, w \in V(G) \setminus \{v\}$. Then no (u, w)-path passes through v. Hence, by Theorem 4, v is not a cut-vertex. Since v is arbitrary, no leaf is a cut-vertex of G.

iii) True. Since the edge which is incident on a leaf does not lie in any cycle, it must be a cut-edge.

UNIT 2

DEGREE SEQUENCES AND GRAPH REPRESENTATION

Structure **Page Nos.**

2.1 INTRODUCTION

In Unit 1, you saw graphs as models, and the basic concepts of graphs theory including neighbourhood and degree; walks, paths and cycles; connectedness and components. In this unit, we shall introduce some more essential concepts to understand the structure of graphs.

In Section 2.2, you will see how to define the 'distance' between two vertices of a graph. One particular application of distance, you will see is in characterising bipartite graphs. In this section, you will also see, the related concepts, such as radius and diameter of a graph. In Section 2.3, we shall introduce two important graphs namely, Petersen graph and Hypercube graph, and look at their properties. Section 2.4 deals with the representation of graphs through matrices.

In Section 2.5, we shall discuss the concept of isomorphism, which will help you to decide when two graphs are 'isomorphic'. Finally, Section 2.6 deals with the notion of degree sequence, and graphic sequence.

Objectives

After studying this unit, you should be able to:

- define the distance between two vertices of a graph;
- find the radius and diameter of a graph;
- describe the Petersen and Hypercube graphs;
- represent a graph by adjacency matrix or incidence matrix;
- explain what isomorphism is, and when two graphs are isomorphic;
- differentiate between a degree sequence and a graphic sequence;
- describe and use the Havel-Hakimi Theorem to determine whether a given sequence of non negative integers is a graphic sequence or not.

2.2 DISTANCE

In this section we shall study the concept of distance between two vertices of a graph. We shall also see the related concepts such as the diameter and the radius of a graph.

Definition: Let G be a graph, and $u, v \in V(G)$. The **distance** between u and v, denoted by $d_G(u,v)$, is the length of a shortest (u,v)-path. If there is no (u,v)-path in G then $d_G(u,v) = \infty$.

Note that $d_G(u,v) = \infty$ iff u and v lie in different components. When there is no confusion, we shall write $d(u,v)$ in place of $d_G(u,v)$.

For example consider the graph given below.

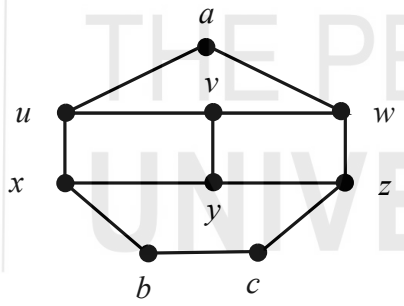


Fig. 1

The distance between some pairs of vertices of this graph is as follows.

$$d(a,u) = 1, \quad d(a,c) = 3, \quad d(u,z) = 3, \quad d(v,z) = 2$$

You can compute the distances between other pairs of vertices similarly. You will find that the maximum distance between any two vertices of this graph is 3.

From other mathematics courses, such as real analysis or topology, you can recall that a distance function d on a set S is a nonnegative function that satisfies for all $x, y, z \in S$:

1) $d(x,y) = 0 \Leftrightarrow x = y$

2) $d(x,y) = d(y,x)$

3) $d(x,z) \leq d(x,y) + d(y,z)$

The first two properties hold trivially for the distance defined in graphs. Let us verify the third property, which is called the **triangle inequality,** in the example below.

Example 1: In a graph G, show that for all $x, y, z \in V(G)$
$d(x,z) \le d(x,y) + d(y,z)$.

Solution: Let P, P' and P'' be three shortest (x,y)-path, (y,z)-path, and (x,z)-path, respectively. Then

$$\text{length}(P) = d(x,y), \quad \text{length}(P') = d(y,z), \quad \text{length}(P'') = d(x,z)$$

Note that $P + P'$ is an (x,z)-walk of length $d(x,y) + d(y,z)$. So, there exists an (x,z)-path in $P + P'$ of length at most $d(x,y) + d(y,z)$. (Why?) But P'' is an (x,z)-path of shortest length. Hence, $d(x,z) \le d(x,y) + d(y,z)$.

***

Recall, from Unit 1, that a bipartite graph can be written as a union of two nonempty disjoint independent sets. The concept of distance helps us get another characterisation of bipartite graphs in terms of even cycles.

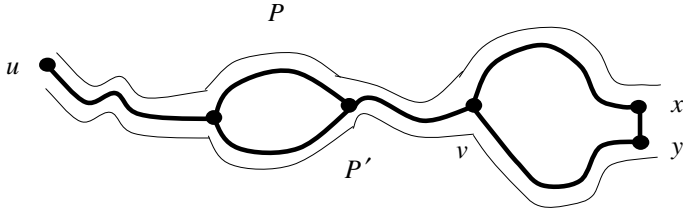
Theorem 1: A graph is bipartite if and only if all its cycles are even.

Proof: Suppose that G is a bipartite graph with a bipartition (X,Y). Let $C = (v_1, v_2, v_3, ..., v_k, v_1)$ be a cycle in G. Without loss of generality, we can assume that $v_1 \in X$. Since v_2 is adjacent to v_1, we have $v_2 \in Y$. Similarly, v_3 belongs to X, v_4 to Y, and so on. Thus, $v_i \in X$ or Y according as i is odd or even, for $1 \le i \le k$. Since $v_k v_1$ is an edge of G and $v_1 \in X$, we have $v_k \in Y$. Accordingly, k is even. Thus, C is an even cycle.

Conversely, let us suppose that G contains only even cycles. We first assume that G is connected. Let u be a vertex of G. Define

$X = \{v \in V(G) : d(u,v)$ is even. $\}$ and $Y = \{v \in V(G) : d(u,v)$ is odd. $\}$.

We will prove that (X,Y) is a bipartition of G. To prove this, we have to show only that X and Y are independent sets. So, let $x, y \in X$ be such that $x \leftrightarrow y$. Then $d(u,x)$ and $d(u,y)$ are even. Let P be a shortest (u,x)-path, and P' a shortest (u,y)-path. Then the lengths of both P and P' are even. If u is the only common vertex of P and P', then $(P \cup P') + xy$ is a cycle of odd length, a contradiction. Otherwise consider a vertex v on both P and P' such that $d(u,v)$ is maximum. (See the following picture.)



Now look at the cycle C formed by the (v,x)-path along P, the edge xy, and the (y,v)-path along P'. We have

$$\text{length}(C) = [\text{length}(P) - d(u,v)] + [\text{length}(P') - d(u,v)] + 1$$
$$= \text{length}(P) + \text{length}(P') - 2d(u,v) + 1$$

This implies C is an odd cycle, which contradicts the assumption. Hence, X is an independent set. Similarly, Y is an independent set. Consequently, G is a bipartite graph. ■

Recall that the diameter of a set in a metric space is the maximum distance between any two points of the set. In the same spirit, we can define the diameter of a graph.

Definition: The **diameter** of a graph G, denoted by $\text{diam}(G)$, is the maximum of the distances between all the pairs of the vertices of G, that is,

$$\text{diam}(G) = \max_{u,v \in V(G)} \{d_G(u,v)\}.$$

Since the maximum distance between any two vertices of the graph in Fig. 1 above is 3, the diameter of the graph is 3. Now let us consider the following theorem that relates the diameter of a graph with its complement.

Theorem 2: If G is a graph and $\text{diam}(G) \geq 3$, then $\text{diam}(\overline{G}) \leq 3$.

Proof: Suppose we have $\text{diam}(G) \geq 3$. Choose $u, v \in V(G)$ such that $d(u,v) \geq 3$. Then u and v are not adjacent, and neither they have any common neighbour in G. (Why?). Let $x \in V(\overline{G}) \setminus \{u,v\}$. Then x must be adjacent to at least one of u and v in $\overline{G}$. Since $u \leftrightarrow v$ in $\overline{G}$, we have $d_{\overline{G}}(x,u) \leq 2$, and $d_{\overline{G}}(x,v) \leq 2$.

Now let, $x, y \in V(\overline{G}) \setminus \{u,v\}$. Then there are four cases.

Case i): $x, y \in N_{\overline{G}}(u)$

In this case $d_{\overline{G}}(x,y) \leq d_{\overline{G}}(x,u) + d_{\overline{G}}(u,y) = 2$.

Case ii): $x, y \in N_{\overline{G}}(v)$

In this case $d_{\overline{G}}(x,y) \leq d_{\overline{G}}(x,v) + d_{\overline{G}}(v,y) = 2$

Case iii): $x \in N_{\overline{G}}(u), y \in N_{\overline{G}}(v)$

In this case $d_{\overline{G}}(x,y) \leq d_{\overline{G}}(x,u) + d_{\overline{G}}(u,v) + d_{\overline{G}}(v,y) = 3$

Case iv): $x \in N_{\overline{G}}(v), y \in N_{\overline{G}}(u)$

This is similar to the previous case.

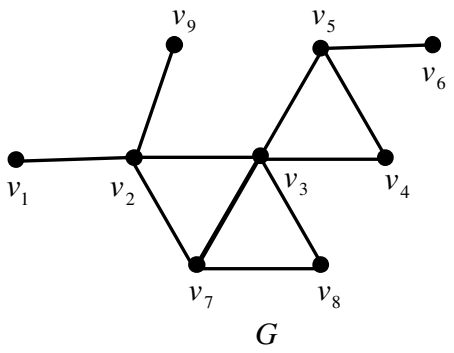
Thus in all the cases, we have shown that $d_{\overline{G}}(x,y) \leq 3$. Since x and y are arbitrary, we have $\max_{x,y \in V(G)} d_{\overline{G}}(x,y) \leq 3$. Therefore, $\text{diam}(\overline{G}) \leq 3$. ■

Just like the diameter, we can define the radius of a graph. See the definition below.

Definition: Let G be a graph. The **eccentricity** of a vertex v of G, denoted by $\varepsilon_G(v)$, is the maximum distance of v from any other vertex of G. The

radius of G, denoted by $\text{rad}(G)$, is the minimum eccentricity of a vertex of G.

Let us consider the graph G given below.



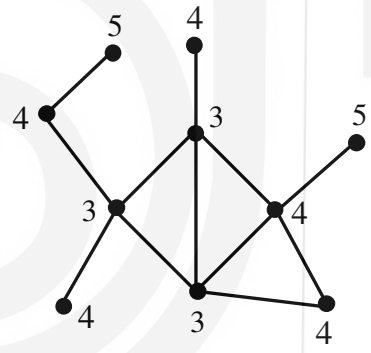
G

When the graph G is clear from the context, we shall write $\varepsilon(v)$, instead of $\varepsilon_G(v)$.

Note that
$\text{diam}(G) = \max_{v \in V(G)} \varepsilon_G(v)$

Here we can see that the eccentricity of v_1 is 4, i.e. $\varepsilon(v_1) = 4$ (This is because v_6 is the farthest vertex from v_1 at a distance of 4). Similarly we have $\varepsilon(v_2) = 3, \varepsilon(v_9) = 4, \varepsilon(v_7) = 3, \varepsilon(v_8) = 3, \varepsilon(v_5) = 3, \varepsilon(v_4) = 3, \varepsilon(v_6) = 4, \varepsilon(v_3) = 2.$
Thus, $\text{rad}(G) = 2$ and $\text{diam}(G) = 4$.

Example 2: Let us consider the following graph in which every vertex is labelled with its eccentricity.



Here the minimum eccentricity is 3, and the maximum eccentricity is 5. So, $\text{rad}(G) = 3$ and $\text{diam}(G) = 5$.

***

Thus, unlike geometry or analysis, the diameter of a graph need not equal twice the radius. Instead, we have the following relation between the radius and the diameter of a graph.

Theorem 3: If G is a connected graph, then

$$\text{rad}(G) \leq \text{diam}(G) \leq 2\,\text{rad}(G).$$

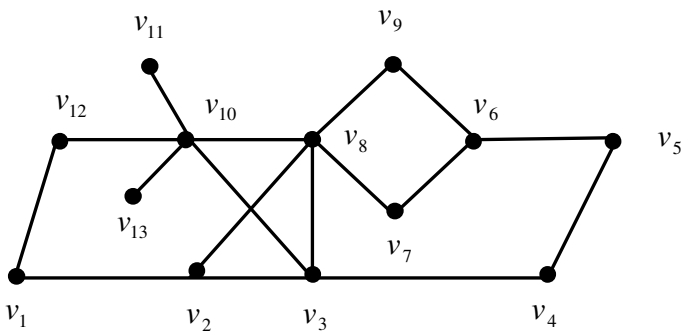
Proof: The inequality $\text{rad}(G) \leq \text{diam}(G)$ follows from the definition. We have to prove the second inequality $\text{diam}(G) \leq 2\,\text{rad}(G)$. So consider two vertices u and v in G, with $d(u,v) = \text{diam}(G)$. Let w be a vertex in G with $\varepsilon(w) = \text{rad}(G)$. Then

$$\begin{aligned} \text{diam}(G) = d(u,v) &\leq d(u,w) + d(v,w) \\ &\leq \varepsilon(w) + \varepsilon(w) = 2\,\text{rad}(G) \end{aligned}$$

■

Now, you may try these exercises.

E1) Find the diameter and radius of the following graph.



E2) Find the diameter and radius of the following graphs.

i) K_n ii) C_n iii) P_n iv) $K_{m,n}$

E3) Give an example of a graph, different from any of the graphs given in E2) such that

i) its radius and diameter are equal.

ii) its radius is half its diameter.

E4) Does there exist a graph with radius 2 and diameter 5? Justify.

In the next section, we shall discuss some special graphs.

2.3 SOME SPECIAL GRAPHS

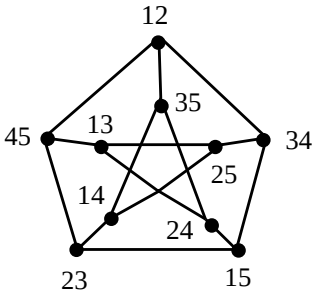
In this section, we shall study two special graphs: the Petersen graph and the hypercube graph, and some of their properties.

Petersen graph

Definition: The **Petersen graph** is a graph whose vertices are the 2-element subsets of a 5-element set $\{1,2,3,4,5\}$, and whose edges are pairs of disjoint 2-element subsets of $\{1,2,3,4,5\}$.

Note that a 2-element subset of $\{1,2,3,4,5,\}$ can be chosen in $\binom{5}{2}$ ways, i.e., in 10 ways. So there are 10 vertices in the Petersen graph. Now consider any vertex $\{u,v\}$. There are three remaining elements in $\{1,2,3,4,5\}$. Of these three elements, there are $\binom{3}{2}=3$ ways to pick a 2-element subset. This means, there are exactly three 2-element subsets disjoint from $\{u,v\}$, and hence $\{u,v\}$ is adjacent to all of them. Thus every vertex in the Petersen graph has degree 3, that is, it is 3-regular. Hence the number of edges in the Petersen graph is $\frac{3\times 10}{2}=15$.

The following is a drawing of the Peterson graph. (For convenience we have written $\{u,v\}$ as uv).



Petersen Graph

Look at any two nonadjacent vertices in the graph above. How many neighbours they have in common? Exactly one, right? This is guaranteed by the following proposition.

Proposition 1: If two vertices are nonadjacent in the Petersen Graph, then they have exactly one common neighbour.

Proof: Consider two nonadjacent vertices A and B of the Petersen graph. Then $A, B \subseteq \{1,2,3,4,5\}$, and $|A| = |B| = 2$. Since A and B are nonadjacent, $A \cap B \neq \emptyset$. But both A and B have cardinality 2, so $|A \cup B| = 3$. Let $C = \{1,2,3,4,5\} \setminus (A \cup B)$. Then $|C| = 2$. Also $A \cap C = B \cap C = \emptyset$. Thus C is the only common neighbour of A and B. ■

Recall that the **girth** of a graph is the length of a shortest cycle in it.

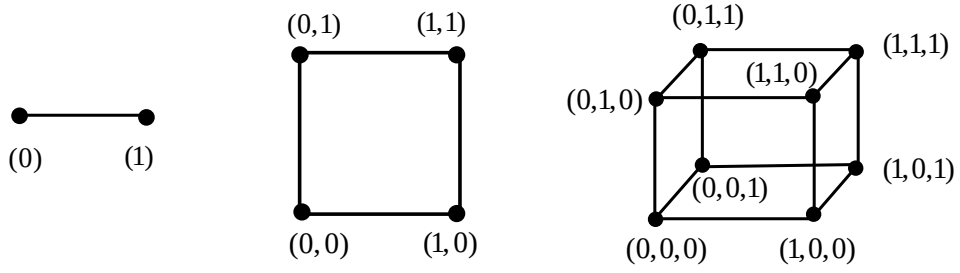
Proposition 2: The Petersen graph has girth 5.

Proof: By definition, the Petersen graph has no loops and parallel edges. This means it has no 1-cycle and 2-cycle. A 3-cycle would require 3 pairwise-disjoint 2-sets, which cannot occur among 5 elements. A 4-cycle in the absence of 3-cycles would require nonadjacent vertices with 2 common neighbours that is also not possible due to Proposition 1. The vertices 12, 34, 25, 14, 35 yield a 5-cycle, so the girth of the Petersen graph is 5. ■

Hypercube Graph

Definition: The k-**dimensional cube,** also called a **hypercube** of dimension k, is a graph whose vertices are the k-tuples $(a_1, a_2, \ldots, a_k)$, where $a_i \in \{0,1\}, \ \forall\ 1 \leq i \leq k;$ and the edges are the pairs of k-tuples that differ exactly at one position.

We denote a k-dimensional cube by Q_k. See below the drawing of Q_1, Q_2 and Q_3.



Q_1 Q_2 Q_3

Since each entry in a vertex $(a_1, a_2, \ldots a_k)$ can be 0 or 1, Q_k has 2^k vertices.

For example Q_1 has $2^1 = 2$ vertices, Q_2 has $2^2 = 4$ vertices and Q_3 has $2^3 = 8$ vertices. Thus $n(Q_k) = 2^k$. What is $m(Q_k)$? To compute, note that a vertex $(a_1, a_2, \ldots a_k)$ is adjacent to a vertex that differs at exactly one position. But there are k positions in $(a_1, a_2 \ldots a_k)$. So every vertex is adjacent to k vertices. This means Q_k is k-regular. Thus $m(Q_k) = \frac{2^k.k}{2} = k.2^{k-1}$. (Why?)

The following example gives more information about the structure of hypercubes.

Example 3: Show that Q_k is a bipartite graph.

Solution: Let X be the set of all vertices that contain an odd number of 1s, and Y the set of all those vertices that contain an even number of 1s. Then (X, Y) is a bipartition of Q_k.

***

The hypercubes naturally occur in computer architecture. Processors can communicate directly if they correspond to adjacent vertices in Q_k. The $k-$tuples that name the vertices serve as addresses for the processors.

Now, to check your understanding solve the following exercises.

E5) Determine whether the Petersen graph is bipartite.

E6) Find the diameter and radius of the Petersen graph.

E7) Give a drawing of Q_4.

In the next section, we will discuss how graphs can be represented by matrices.

2.4 REPRESENTATION OF GRAPHS

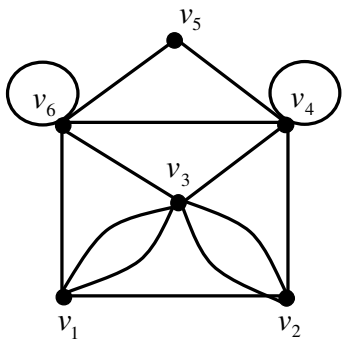
Pictorial representation of a graph is a very convenient approach for visual study of a graph, but not very good for computer processing. To process a graph in a computer all that is required is whether any two vertices are adjacent or not. Alternatively, we can process the incidence relation between an edge and a vertex. Such information can be represented through matrices. We shall discuss here two such matrices, called the adjacency matrix, and the incidence matrix.

2.4.1 Adjacency Matrix

Definition: Let G be a multigraph with $V(G) = \{v_1, v_2, v_3, \ldots, v_n\}$. The **adjacency** matrix of G is an $n \times n$ matrix defined as $A(G) = (a_{ij})_{n \times n}$, where

$$a_{ij} = \begin{cases} \text{no. of edges with endpoints } v_i \text{ and } v_j, & \text{if i} \neq \text{j} \\ \text{twice the no. of loops on } v_i, & \text{if i} = \text{j} \end{cases}$$

For example, look at the multigraph G and its adjacency matrix $A(G)$, given below.



G

$$A(G) = \begin{bmatrix} 0 & 1 & 2 & 0 & 0 & 1 \\ 1 & 0 & 2 & 1 & 0 & 0 \\ 2 & 2 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 2 \end{bmatrix}$$

$A(G)$

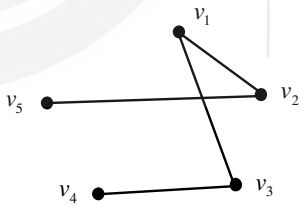
Let us consider some more examples.

Example 4: Draw a graph whose adjacency matrix is

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

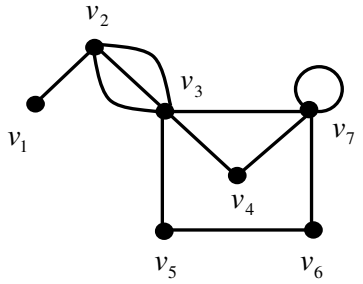
Solution: A has 5 rows and 5 columns, so the corresponding graph has 5 vertices. Let v_1, v_2, v_3, v_4, v_5 be the vertices of the graph. Then from A we find that $N(v_1) = \{v_2, v_3\}, N(v_2) = \{v_1, v_5\}, N(v_3) = \{v_1, v_4\}, N(v_4) = \{v_3\},$ and $N(v_5) = \{v_2\}$.

Using this information, we draw the graph as follows.



***

Example 5: Consider the following multigraph G.



i) Find $A(G)$.

ii) Is $A(G)$ symmetric ?

iii) What is the sum of the values of the entries in each row?, in each column?

iv) How would you interpret the sums in iii) above ?

Solution:

i) We have

$$A(G)=\begin{bmatrix}0&1&0&0&0&0&0\\1&0&3&0&0&0&0\\0&3&0&1&1&0&1\\0&0&1&0&0&0&1\\0&0&1&0&0&1&0\\0&0&0&0&1&0&1\\0&0&1&1&0&1&2\end{bmatrix}$$

ii) We can see that $a_{ij}=a_{ji}$ for all i and j. Hence $A(G)$ is symmetric.

iii) The sum of the entries of $A(G)$ in each row is given below:

row	1	2	3	4	5	6	7
sum	1	4	6	2	2	2	5

The column sums are also the same.

iv) Let us compare the degree of vertex v_i with the sum in row i, for each $i=1,2,\ldots 7,$ what we find is that

$$d(v_i)=\text{sum of the entries of } A(G) \text{ in row } i=\sum_{j=1}^{7}a_{ij}$$

***

In the example above, you saw that the sum of the entries in a row equals the degree of the corresponding vertex. This is indeed the case for all multigraphs. To see, consider a vertex v in a multigraph G. Then the quantity $\sum_{u\in V(G)}a_{uv}$ represents the number of edges incident on v, with each loop counted twice. This is exactly the degree of v. That is,

$$d(v)=\sum_{u\in V(G)}a_{uv}$$

Now in the equation above, let us take the sum over all $v\in V(G)$. We get

$$\sum_{v\in V(G)}d(v)=\sum_{v\in V(G)}\sum_{u\in V(G)}a_{uv},$$

which implies $2m(G)=\sum_{u,v\in V(G)}a_{uv}$, or

$$m(G)=\frac{1}{2}\sum_{u,v\in V(G)}a_{uv}.$$

This shows that we can compute the number of edges in a multigraph using its adjacency matrix.

Let us consider some examples.

Example 6: Let G be a graph with $V(G)=\{2,3,4,5,11,12,13,14\}$ and for any $u,v\in V(G)$

$$u\leftrightarrow v \;\Leftrightarrow\; \gcd(u,v)=1.$$

Find $A(G)$ and $m(G)$.

Solution: Note that for any $u,v\in V(G)$,

$$a_{uv}=1 \;\Leftrightarrow\; u\leftrightarrow v \;\Leftrightarrow\; \gcd(u,v)=1$$

Hence the adjacency matrix of G is

$$A(G)=\begin{bmatrix}0&1&0&1&1&0&1&0\\1&0&1&1&1&0&1&1\\0&1&0&1&1&0&1&0\\1&1&1&0&1&1&1&1\\1&1&1&1&0&1&1&1\\0&0&0&1&1&0&1&0\\1&1&1&1&1&1&0&1\\0&1&0&1&1&0&1&0\end{bmatrix}$$

Therefore, we have $m(G)=\frac{1}{2}\sum_{u,v\in V(G)}a_{uv}=\frac{42}{2}=21$.

***

2.4.2 Incidence Matrix

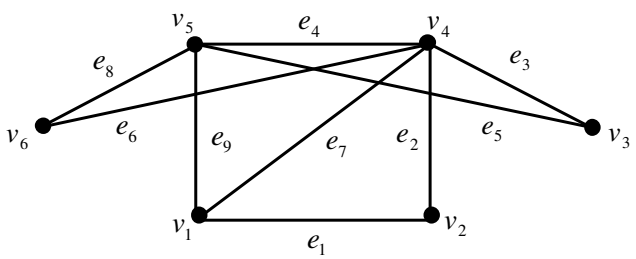
The incidence matrix is defined below.

Definition: Let G be a loopless multigraph with $V(G)=\{v_1,v_2,\ldots,v_n\}$ and $E(G)=\{e_1,e_2,\ldots,e_m\}$. The **incidence** matrix of G is an $n\times m$ matrix defined as $B(G)=(b_{ij})_{n\times m}$, where

$$b_{ij}=\begin{cases}1, \text{ if } e_j \text{ is incident on } v_i\\0, \text{ othewise.}\end{cases}$$

Note that in incidence matrix, n rows correspond to the n vertices and m columns correspond to the m edges.

Example 7: Find the incidence matrix of the following graph.



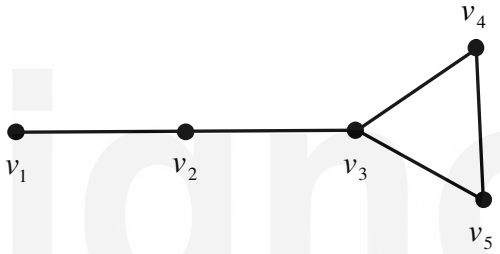
Solution: The incidence matrix is given below.

$$B(G)=\begin{bmatrix}1&0&0&0&0&0&1&0&1\\1&1&0&0&0&0&0&0&0\\0&0&1&0&1&0&0&0&0\\0&1&1&1&0&1&1&0&0\\0&0&0&1&1&0&0&1&1\\0&0&0&0&0&1&0&1&0\end{bmatrix}$$

***

In the incidence matrix $B(G)$ of a graph G, you can see that the sum of all the entries in a row represents the degree of the corresponding vertex. However, each column has exactly two 1s corresponding to the endpoints of the edge representing the column.

Since an adjacency matrix A is a square matrix, we can compute its powers $A^2, A^3,$ etc. For example, the adjacency matrix of the graph



is given below:

$$A=\begin{bmatrix}0&1&0&0&0\\1&0&1&0&0\\0&1&0&1&1\\0&0&1&0&1\\0&0&1&1&0\end{bmatrix}$$

In A, the $(i,j)^{th}$ entry is 0 or 1, which indicates the number of (v_i, v_j)- walks of length 1. The second and third powers of A are:

$$A^2=\begin{bmatrix}1&0&1&0&0\\0&2&0&1&1\\1&0&3&1&1\\0&1&1&2&1\\0&1&1&1&2\end{bmatrix}, \qquad A^3=\begin{bmatrix}0&2&0&1&1\\2&0&4&1&1\\0&4&2&4&4\\1&1&4&2&3\\1&1&4&3&2\end{bmatrix}$$

Now let us see what we can infer from the $(i,j)^{th}$ entry of A^2. For example, the $(1,1)^{th}$ entry is 1, which is the number of (v_1, v_1)-walks of length 2 in the graph. You can see there is exactly one such walk, that is, (v_1, v_2, v_1).
Now look at the $(1,2)^{th}$ entry which is 0. In the graph there are no (v_1, v_2)-walk of length 2. Similarly, $(3,3)^{th}$ entry is 3, and all the (v_3, v_3)-walks of length 2 are

$$(v_3, v_2, v_3), \qquad (v_3, v_5, v_3), \qquad (v_3, v_4, v_3).$$

You can look at other entries. What you will observe is that in $A^2, (i,j)^{th}$ entry represents the number of (v_i, v_j)-walks of length 2.

Now let us look at the entries of A^3.

The $(2,3)^{rd}$ entry is 4, and all (v_2, v_3)-walks of length 3 are:

$$(v_2, v_1, v_2, v_3),\ (v_2, v_3, v_5, v_3),\ (v_2, v_3, v_4, v_3),\ (v_2, v_3, v_2, v_3)$$

Thus in A^3, $(i, j)^{th}$ entry represents the number of (v_i, v_j)-walks of length 3.

Now can you guess what the $(i, j)^{th}$ entry of A^k would represent? The answer is given in the following theorem.

Theorem 4: Let A be the adjacency matrix of a graph G with $V(G) = \{v_1, v_2, \ldots, v_n\}$. Then the $(i, j)^{th}$ entry of A^k, for $k \geq 1$, represents the number of (v_i, v_j)-walks of length k in G.

Proof: We shall prove it by the Principle of Mathematical Induction on k. Let $a_{ij}^{(k)}$ denote the $(i, j)^{th}$ entry of A^k. When $k = 1, a_{ij}^{(1)} = a_{ij}$. So if $a_{ij} = 0$, then there is no (v_i, v_j)-walk of length 1. If $a_{ij} = 1$, then v_i and v_j are adjacent and hence there is exactly one (v_i, v_j)-walk of length 1. So, the result holds for $k = 1$.

Now, assume that $a_{ij}^{(k-1)}$ represents the number of (v_i, v_j)-walks of length $k-1$. Consider the $(i, j)^{th}$ entry $a_{ij}^{(k)}$ of A^k, which can be obtained by multiplying the i^{th} row of A^{k-1} with the j^{th} column of A. That is

$$a_{ij}^{(k)} = \sum_{r=1}^{n} a_{ir}^{(k-1)} a_{rj}.$$

Note that $a_{ir}^{(k-1)} a_{rj}$ is nonzero only when $a_{rj} \neq 0$, i.e. when $a_{rj} = 1$. But this means there is an edge between v_r and v_j. Thus $a_{ir}^{(k-1)} a_{rj}$ is the number of (v_i, v_j)-walks of length $(k-1)+1 = k$ through v_r. Hence $a_{ij}^{(k)}$ is the number of all (v_i, v_j)-walks of length k.

By the Principle of Mathematical induction the result holds for all $k \geq 1$. ■

The following result tells us more about the usefulness of adjacency matrices.

Theorem 5: Let G be a graph with $V(G) = \{v_1, v_2, \ldots, v_n\}$ and the adjacency matrix A. Then G is connected iff every nondiagonal entry of the matrix

$$M = A + A^2 + \ldots + A^{n-1}$$

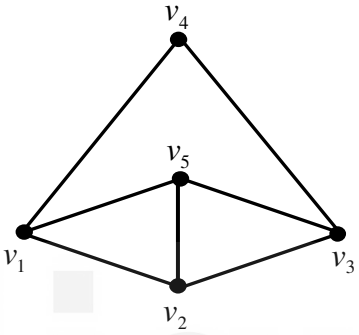
is nonzero.

Proof: First, let us assume that G is connected. Pick distinct v_i, v_j in $V(G)$ arbitrarily. Connectedness of G implies that there is a (v_i, v_j)-path of length k, where $1 \leq k \leq n-1$. This means the $(i, j)^{\text{th}}$ entry of A^k is nonzero. (Why?) Since $(i, j)^{\text{th}}$ entry of each power of A is nonnegative, $(i, j)^{\text{th}}$ entry of M is also nonzero. Since v_i and v_j are arbitrary, each non diagonal entry of M is nonzero.

Conversely, assume that each entry of M is nonzero. Pick any v_i and v_j in $V(G)$. Then $(i,j)^{\text{th}}$ entry of A^k is nonzero for some $k, 1 \le k \le n-1$. So by Theorem 4, there is a (v_i, v_j)- walk in G. Hence G is connected. ■

Theorem 5 is useful in specific settings. For example, the connectedness of a graph represented by the adjacency matrix can be checked by a computer program.

Example 8: Find the number of (v_1, v_3) -walks of length $1, 2, 3, 4$ and 5 in the following graph.



Solution: The adjacency matrix of the given graph is

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Here $a_{13} = 0$, which means there is no (v_1, v_3)-walk of length 1. Now we compute

Note that the $*$ed entries are left uncomputed.

$$A^2 = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 1 & 3 & 0 & 1 \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}$$

We can see that the number of (v_1, v_3)-walks of length 2 is $a_{13}^{(2)} = 3$. Now we compute

$$A^3 = \begin{bmatrix} 3 & 1 & 3 & 0 & 1 \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 7 & 2 & 6 & 7 \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}$$

From here we find that $a_{13}^{(3)} = 2$, which is the number of (v_1, v_3)-walks of length 3. Similarly, you can compute

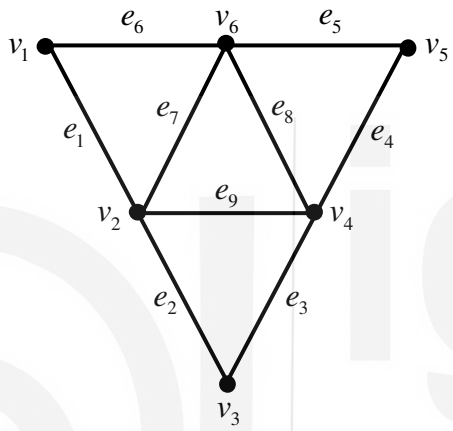
$$A^4 = \begin{bmatrix} 20 & 11 & 20 & 4 & 11 \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}, \text{ and } A^5 = \begin{bmatrix} 26 & 51 & 26 & 40 & 51 \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}$$

We find that $a_{13}^{(4)} = 20$ and $a_{13}^{(5)} = 26$, which are the number of (v_1, v_3)-walks of length 4 and 5, respectively.

***

Now, try the following exercises.

E8) Write down the adjacency and incidence matrices of the following graph:



E9) If A is the adjacency matrix of a graph, then A^k is symmetric for every $k \geq 1$. Prove the statement using the Principle of Mathematical Induction.

E10) Check whether the graphs or multigraphs represented by the following adjacency matrices are connected or not.

i) $\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$

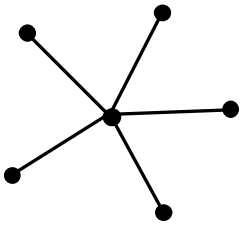
ii) $\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$

E11) Draw the graph or multigraph corresponding to each of the following adjacency matrices:

i) $\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$

ii) $\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$

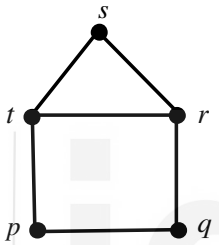
E12) Find the number of walks of length 4 between any two nonadjacent vertices of the following graph:



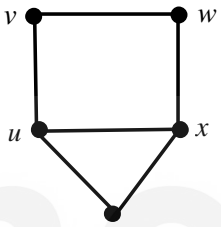
In the next section, we shall talk about the concept of isomorphism which tells us whether two graphs are structurally same or not.

2.5 ISOMORPHIC GRAPHS

Look at the following graphs.



G_1



G_2

Do you find them different, or the same? Clearly, $G_1 \neq G_2$ as $V(G_1) \neq V(G_2)$. Now, for a moment, ignore their labels. Then you will see that they are exactly the same. If you observe a relation between two vertices of G_1 you can see the same relation between the "corresponding" vertices in G_2. For example in G_1, t and r have a common neighbour s, and in G_2 the vertices u and x have a common neighbour y. Such graphs are called "isomorphic" to each other. The formal definition is given below:

Definition: Let G_1 and G_2 be two graphs. Then G_1 is said to be **isomorphic** to G_2, written as $G_1 \cong G_2$, if there exists a one-one onto function $\phi : V(G_1) \to V(G_2)$ such that

$$uv \in E(G_1) \Leftrightarrow \phi(u)\phi(v) \in E(G_2). \qquad (1)$$

The function ϕ is called an **isomorphism** from G_1 to G_2.

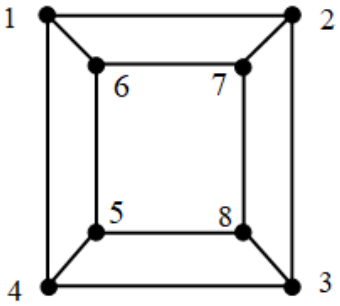
Let us resume our discussion on the graphs G_1 and G_2 given above. Let $\phi : V(G_1) \to V(G_2)$ be defined as $\phi(p) = w, \phi(q) = v, \phi(r) = u, \phi(s) = y, \phi(t) = x.$ Now for each edge in G_1, we verify the statement (1).

$$\begin{aligned}
pq \in E(G_1) &\Leftrightarrow \phi(p)\phi(q) = wv \in E(G_2)\\
qr \in E(G_1) &\Leftrightarrow \phi(q)\phi(r) = vu \in E(G_2)\\
rs \in E(G_1) &\Leftrightarrow \phi(r)\phi(s) = uy \in E(G_2)\\
st \in E(G_1) &\Leftrightarrow \phi(s)\phi(t) = yx \in E(G_2)\\
tp \in E(G_1) &\Leftrightarrow \phi(t)\phi(p) = xw \in E(G_2)\\
rt \in E(G_1) &\Leftrightarrow \phi(r)\phi(t) = ux \in E(G_2)
\end{aligned}$$

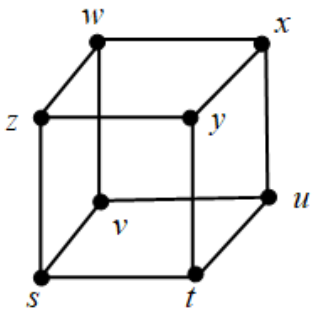
Thus we have proved that ϕ is an isomorphism from G_1 to G_2. Therefore, $G_1 \cong G_2$.

Are you wondering that $G_1 \cong G_2$ implies $G_2 \cong G_1$? It is true. You can verify that if ϕ is an isomorphism from G_1 to G_2 then ϕ^{-1} is an isomorphism from G_2 to G_1. Thus when $G_1 \cong G_2$, we call G_1 and G_2 are **isomorphic**, otherwise **nonisomorphic**. Let us see some examples.

Example 9: Show that the graphs shown below are isomorphic.



G_1



G_2

Solution: Let us define $\phi : V(G_1) \to V(G_2)$ as below:

$$\phi(1) = w,\ \phi(2) = x,\ \phi(3) = u,\ \phi(4) = v,$$
$$\phi(5) = s,\ \phi(6) = z,\ \phi(7) = y,\ \phi(8) = t$$

By definition it is clear that ϕ is one-one and onto. Now

$$12 \in E(G_1) \Leftrightarrow \phi(1)\phi(2) = wx \in E(G_2)$$
$$23 \in E(G_1) \Leftrightarrow \phi(2)\phi(3) = xu \in E(G_2)$$
$$34 \in E(G_1) \Leftrightarrow \phi(3)\phi(4) = uv \in E(G_2)$$
$$41 \in E(G_1) \Leftrightarrow \phi(4)\phi(1) = vw \in E(G_2)$$
$$56 \in E(G_1) \Leftrightarrow \phi(5)\phi(6) = sz \in E(G_2)$$
$$67 \in E(G_1) \Leftrightarrow \phi(6)\phi(7) = zy \in E(G_2)$$
$$78 \in E(G_1) \Leftrightarrow \phi(7)\phi(8) = yt \in E(G_2)$$
$$85 \in E(G_1) \Leftrightarrow \phi(8)\phi(5) = st \in E(G_2)$$
$$16 \in E(G_1) \Leftrightarrow \phi(1)\phi(6) = wz \in E(G_2)$$
$$27 \in E(G_1) \Leftrightarrow \phi(2)\phi(7) = xy \in E(G_2)$$
$$38 \in E(G_1) \Leftrightarrow \phi(3)\phi(8) = ut \in E(G_2)$$
$$45 \in E(G_1) \Leftrightarrow \phi(4)\phi(5) = vs \in E(G_2)$$

Thus we have proved that ϕ is an isomorphism from G_1 to G_2.
Hence $G_1 \cong G_2$.

***

If you have to prove that two graphs G_1 and G_2 are isomorphic you **have to** use the definition, there is no alternative. An isomorphism from G_1 to G_2 preserves the adjacency relation between G_1 and G_2. This implies, G_1 and G_2 must have the same order, the same number of components, the same number of triangles, the same girth, the same radius, the same diameter, etc. In short, we can say that

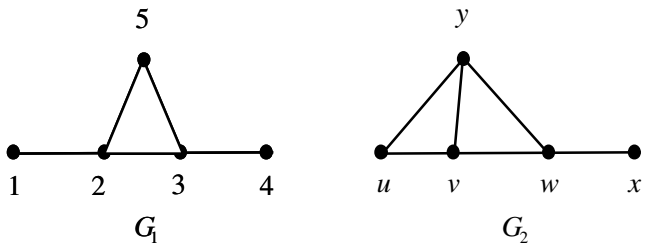
$G_1 \cong G_2 \Rightarrow$ If G_1 has a property P, then G_2 also has the property P.

Can you write down the contrapositive of the statement above? The contrapositive is

If G_1 has a property P, but G_2 does not have $P \Rightarrow G_1 \ncong G_2$

Here, P can be any property a graph may have. Let us look at some examples of nonisomorphic graphs.

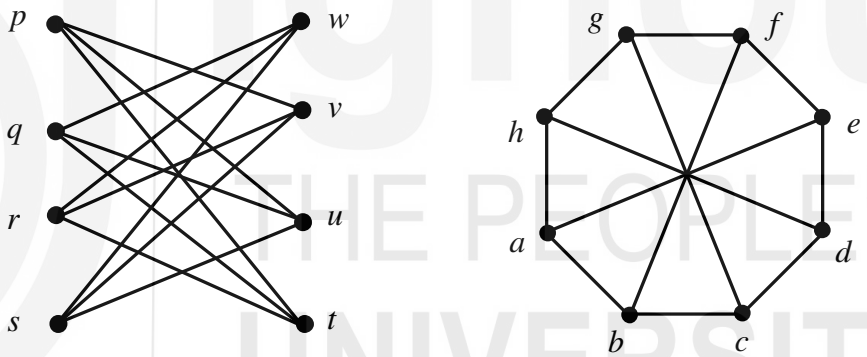
Example 10: Show that the graphs G_1 and G_2 given below are nonisomorphic.



Solution: Since G_1 has only one triangle and G_2 has two triangles, $G_1 \ncong G_2$.

***

Example 11: Show that the graphs below are nonisomorphic.



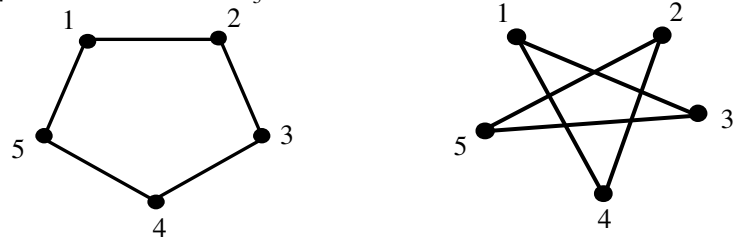
Solution: The graph on the left is bipartite, and so has no odd cycles. However, the graph on the right has a 5-cycle. Hence, they are not isomorphic.

***

By now you must have understood that an isomorphism between two graphs tells us that they are the same, a copy of each other, but labelled or drawn differently. Recall that the complement of a graph is obtained by removing the existing edges from the graph and adding the missing edges. This process may produce a copy of the original graph. What will you call such a graph? "Self-complementary"? Let us see the definition below.

Definition: A graph G is said to be **self-complementary** if $G \cong \overline{G}$.

For example, let us look at C_5 and $\overline{C}_5$ drawn below



C_5 $\overline{C}_5$

Note that $\overline{C}_5$ is the cycle $(1,3,5,2,4,1)$. Here $E(C_5)=\{12,23,34,45,51\}$, and $E(\overline{C}_5)=\{13,35,52,24,41\}$. We have one-one onto mapping $\phi:\{1,2,3,4,5,\}\rightarrow\{1,2,3,4,5\}$ defined as

$$\phi(1)=1,\phi(2)=3,\phi(3)=5,\phi(4)=2,\phi(5)=4.$$

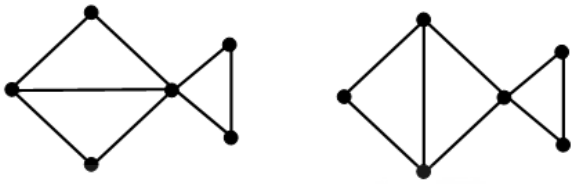
We leave it for you to check that for all $u,v\in\{1,2,3,4,5\}$ $uv\in E(C_5)$ if and only if $\phi(u)\phi(v)\in E(\overline{C}_5)$.

Two unlabelled graphs are isomorphic if their labelled versions are isomorphic.

Now you should be ready to do some exercises.

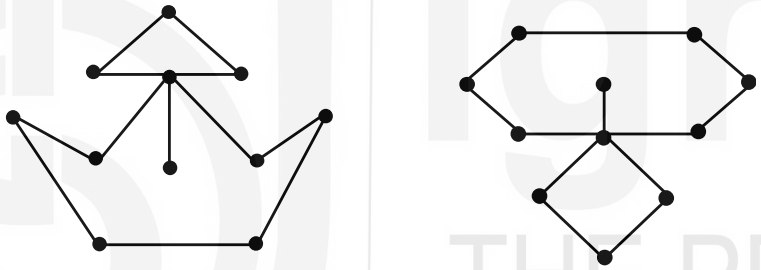
E13) Which of the following pairs of graphs are isomorphic ?

i)



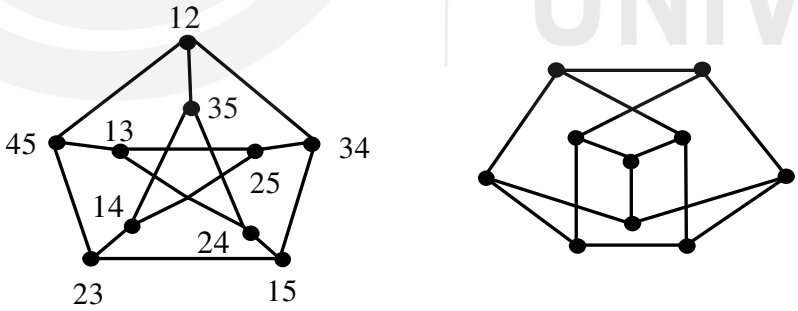
G_1 G_2

ii)



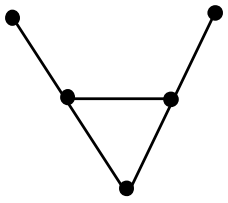
G_1 G_2

E14) Show that the following graphs are isomorphic.

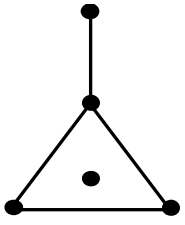


E15) Which of the following graphs are self-complementary? Justify your answer.

i)



ii)



E16) Let G be a self-complementary graph. Let H_1 be the graph obtained by joining each vertex of G to the two endpoints of P_4. Let H_2 be the graph obtained by joining each vertex of G to the two internal vertices of P. Show that both H_1 and H_2 are self-complementary.

E17) Prove that if G is an n-vertex self-complementary graph, then n or $n-1$ is divisible by 4.

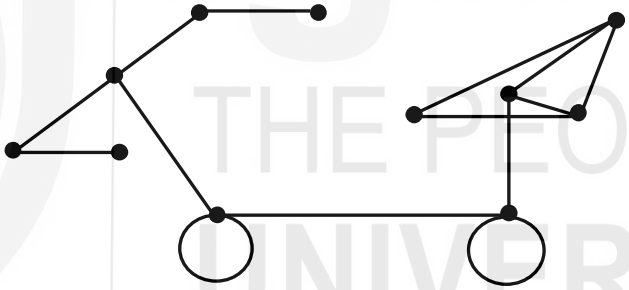
E18) Construct a self-complementary graph on 8 vertices.

In the next section, we shall discuss degree sequences and graphic sequences.

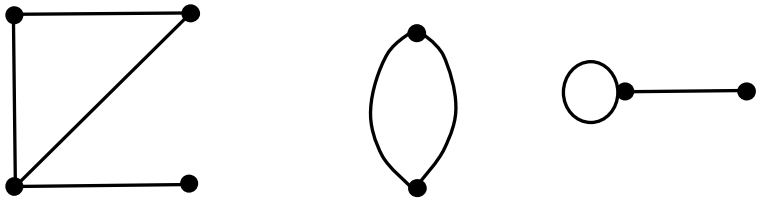
2.6 DEGREE SEQUENCES AND GRAPHIC SEQUENCES

In Section 1.5 of Unit 1, we have discussed degrees of vertices in graphs and multigraphs. Here, we shall study degrees of all vertices of a graph or multigraph together.

Definition 21: The **degree sequence** of a multigraph is the sequence of its vertex degrees $(d_1, d_2, \ldots, d_n)$, where $d_1 \geq d_2 \geq \ldots \geq d_n$.
For example, consider the following multigraph.



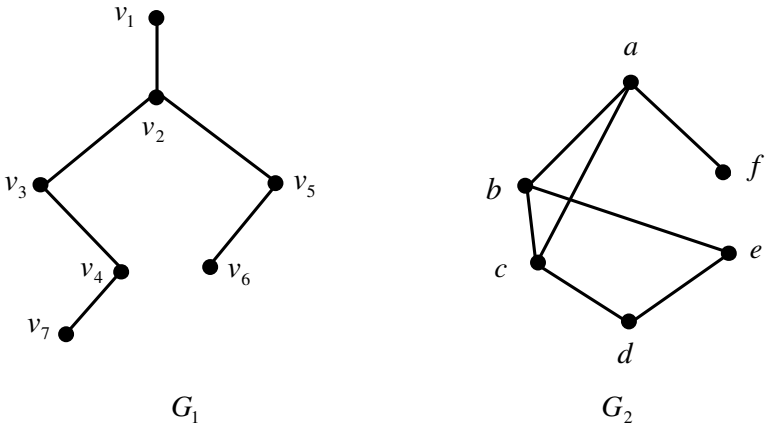
In this graph there are two vertices of degree 4, four vertices of degree 3, three vertices of degree 2, and two vertices of degree 1. Hence its degree sequence is $(4,4,3,3,3,3,2,2,2,1,1)$. Of course, every multigraph has exactly one degree sequence. However, two multigraphs may have the same degree sequence. For example, the graph G_1 and the multigraph G_2 given below have the same degree sequence $(3,2,2,1)$.



G_1 G_2

Let us consider one more example.

Example 12: Write the degree sequences of the following graphs.



G_1 G_2

Solution: In G_1, we find that $d(v_1)=1, d(v_2)=3$, $d(v_3)=d(v_4)=d(v_5)=2$ and $d(v_6)=d(v_7)=1$. Hence, the degree sequence of G_1 is $(3,2,2,2,1,1,1)$.

Similarly, in G_2, the degrees of the vertices a,b,c,d,e and f are $3,3,3,2,2$ and 1 respectively. Hence the degree sequence of G_2 is $(3,3,3,2,2,1)$.

***

Did you observe that the sum of the terms in each of the degree sequences found above is even? This is because of the following theorem.

Theorem 6: Let $d=(d_1,d_2,\ldots,d_n)$ be a decreasing sequence of nonnegative integers. Then d is a degree sequence iff $\sum_{i=1}^{n} d_i$ is even.

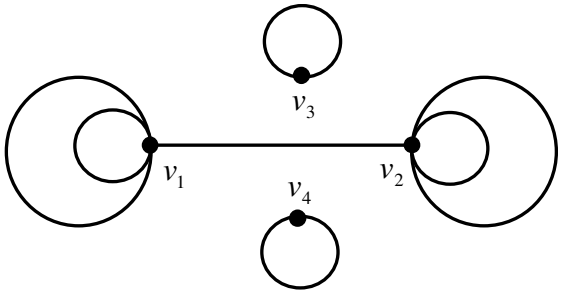
Proof: If $d=(d_1,d_2,\ldots d_n)$ is a degree sequence then there exists a multigraph with $d_i s$ as the degrees of its vertices. Hence by the Hand-shaking Lemma for multigraphs (E15 of Unit 1), $\sum_{i=1}^{n} d_i$ is even.

Conversely, let $\sum_{i=1}^{n} d_i$ be even. We shall construct a multigraph with vertices $v_1, v_2 \ldots, v_n$ such that $d(v_i)=d_i$ for all $i \in \{1,2,\ldots,n\}$. Note that $\sum_{i=1}^{n} d_i$ is even implies that the odd values in $(d_1,d_2,\ldots,d_n)$ occur even number of times. So the set $S=\{v_i : d_i \text{ is odd.}\}$ contains an even number of vertices, which can be paired in any order. Draw an edge between every such pair. The remaining degree at each vertex in S is even, and hence we can draw $\frac{d_i-1}{2}$ loops on each of these vertices. The vertices not in S already have even degree, so on each of these vertices draw $\frac{d_i}{2}$ loops, to get a multigraph. ■

The proof of Theorem 6 not just gives a characterisation of degree sequences of multigraphs, but provides a method as well for constructing a multigraph with the given degree sequence. Let us consider the following example.

Example13: Draw a multigraph with degree sequence $d=(5,5,2,2)$.

Solution: Since the sum of the terms of d is even, d is a degree sequence. So there exists a multigraph on 4 vertices, say v_1, v_2, v_3 and v_4. Let $d(v_1) = d(v_2) = 5$. We draw an edge between v_1 and v_2. The remaining degrees at each of v_1 and v_2 is 4, drawing two loops on each of them exhausts their degrees. Now the remaining two vertices v_3 and v_4 have degree 2 each, and so on each of them we draw a loop. The multigraph obtained is given below.

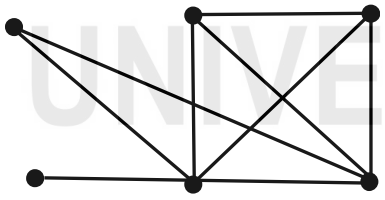


***

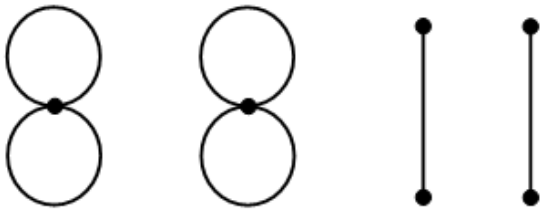
Let us now shift our attention to the degree sequences of graphs only. To distinguish between the degree sequence of a graph with that of a multigraph, you need the following definition.

Definition: A sequence of nonnegative integers $d = (d_1, d_2, \ldots, d_n)$ is said to be **graphic** if the permutation of d in the descending order is the degree sequence of an n-vertex graph.

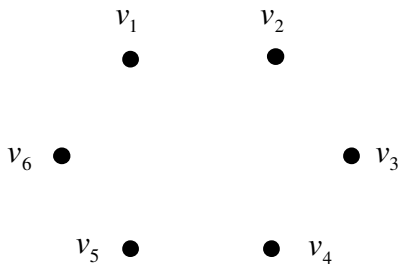
For example, the sequences $(1,2,3,4,5,3)$ and $(1,2,3,3,4,5)$ are graphic, because both can be permutated in the descending order as $(5,4,3,3,2,1)$, which is the degree sequence of the following graph.



Note that the sequence $(4,4,1,1,1,1)$ is a degree sequence of the following multigraph.



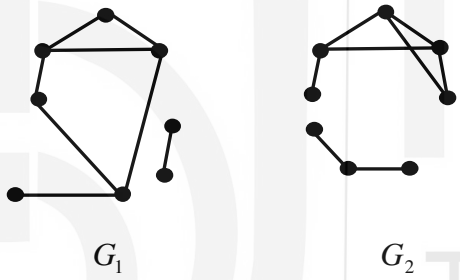
But $(4,4,1,1,1,1)$ is not graphic. That is, there exists no 6-vertex graph with $(4,4,1,1,1,1)$ as its degree sequence. To prove our claim, let us assume, for a moment that there is a 6-vertex graph with degree sequence $(4,4,1,1,1,1)$. Let $v_1, v_2, v_3, v_4, v_5, v_6$ be its vertices. We draw these vertices on a circle, as you can see below.



The graph has two vertices of degree 4, say v_1 and v_2. If v_1 and v_2 are nonadjacent, then both must be adjacent to the rest four vertices, leaving no vertex with degree 1. This is not possible. On the other hand, if v_1 and v_2 are adjacent, then both must have at least two common neighbours. This means, there would be at most two vertices of degree 1, which again is not possible. Thus $(4,4,1,1,1,1)$ is not a graphic sequence.

Example 14: Draw two nonisomorphic graphs with degree sequence $(3,3,3,2,2,1,1,1)$.

Solution: The following graphs G_1 and G_2 have the same degree sequence $(3,3,3,2,2,1,1,1)$.



We can see that G_1 has a 5-cycle, whereas G_2 has no 5-cycle. Thus $G_1 \cong G_2$.

***

From the Hand-shaking Lemma, it is clear that a necessary condition for a sequence $d=(d_1,d_2,...,d_n)$ of nonnegative integers to be graphic is that $\sum_{i=1}^{n} d_i$ is even. However, this is not sufficient. A necessary and sufficient condition was given by Havel and Hakimi, which we state in the following theorem.

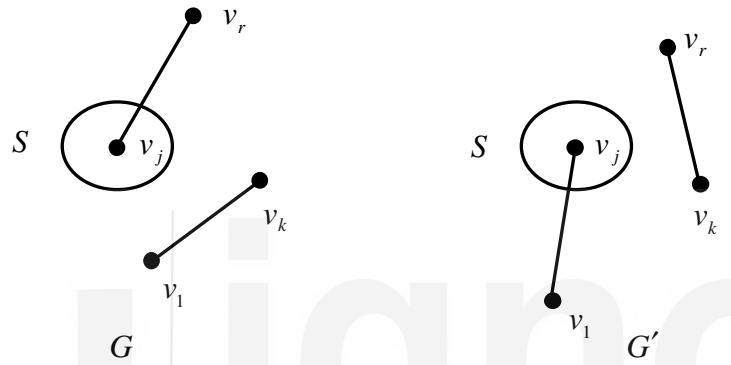
Theorem 7: (Havel-Hakimi Theorem): For $n>1$, a sequence $d=(d_1,d_2,...,d_n)$ is graphic if and only if the sequence d' is graphic, where d' is obtained from d by deleting its largest element (say Δ) and subtracting 1 from its Δ next largest elements. The only 1-element graphic sequence is (0).

Proof: First, let us prove the forward implication.

We know that there can be several graphs with the same degree sequence. Let G be a graph with vertices $v_1, v_2, \ldots, v_n$ such that $d(v_i)=d_i$ for $i=1,2,\ldots,n$. We have to choose G in such a way that the set $S=\{v_2, v_3, \ldots, v_{d_1+1}\}$ is the

neighbourhood of v_1. But the question is - can we do so? We use contradiction to show that this is possible. So, let us start with the assumption that this is not possible. Let G be chosen among all graphs with $d(v_i) = d_i$ such that v_1 is adjacent to as many vertices in S as possible. By definition, some vertex v_j in S is not a neighbour of v_1. But since $d(v_1) = d_1$, there must be some neighbour v_k of v_1 outside S. Since $d_j \geq d_k$, there exists a vertex v_r such that $v_j v_r \in E(G)$ but $v_k v_r \notin E(G)$.

Now, form a new graph G' from G by removing the edges $v_1 v_k$ and $v_j v_r$, and adding two new edges $v_1 v_j$ and $v_k v_r$. (See the picture below.)



It is obvious to see that d is a degree sequence of G', too. Observe that in G', v_1 has more neighbours in S as compared to those in G. This contradicts the choice of G. Hence, we conclude that G can be chosen so that $N(v_1) = S$. Now see that d' is a permutation of the degree sequence of $G - v_1$.

Now, let us prove the converse statement.

We are given a sequence $d = (d_1, d_2, ..., d_n)$ of nonnegative integers such that d' is graphic. So, there exists a graph G with vertex set $V(G) = \{v_1, v_2, \ldots, v_{n-1}\}$ and degree sequence d'. Note that d' is a permutation of

$$(d_2 - 1, d_3 - 1, ..., d_{d_1+1} - 1, d_{d_1+2}, ..., d_n)$$

Now let us relabel the vertices of G as $\{v_1', v_2', \ldots, v'_{n-1}\}$ such that

$$d(v_1') = d_2 - 1, d(v_2') = d_3 - 1, \ldots, d(v'_{d_1}) = d_{d_1+1} - 1, \ldots, d(v'_{n-1}) = d_n .$$

Add a vertex v_n' in G such that $N(v'_n) = \{v_1', v_2', \ldots, v_{d_1}'\}$. Let G' be the graph so obtained .
Clearly $d(v'_n) = d_1$ and $d(v_i') = d_{i+1}, \forall i = 1, \ldots, n-1$.
Thus $d = (d_1, d_2, \ldots d_n)$ is the degree sequence of G'. Hence d is graphic. ■

Now, let us consider some applications of the Havel-Hakimi Theorem.

Example 15: Show that the sequence $(6,6,6,6,4,3,3,0)$ is not graphic.

Solution: Here $d = (6,6,6,6,4,3,3,0)$. So,

$d' = (5,5,5,3,2,2,0)$

$d'' = (4,4,2,1,1,0)$

$d''' = (3,1,0,0,0)$

$d'''' = (0,0,-1,-1)$

Note that d'''' is not graphic, as it has negative entries. Using the Havel-Hakimi Theorem, d''' is not graphic, which implies d'' is not graphic, which implies d' is not graphic, which implies d is not graphic.

Remark 1: To show that a sequence is graphic or not, you can stop as soon as you find that the current sequence is graphic or not.

Example 16: Show that the sequence $(6,5,4,4,3,2,2,2,2)$ is graphic. Also find a graph realising to it.

We say a graph G **realises** a graphic sequence d if d is a permutation of the degree sequence of G.

Solution: We have

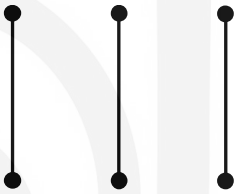
$d = (6,5,4,4,3,2,2,2,2)$

$d' = (4,3,3,2,2,2,1,1)$

$d'' = (2,2,2,1,1,1,1)$

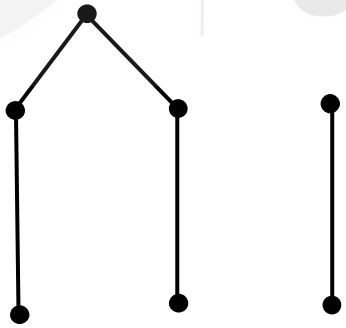
$d''' = (1,1,1,1,1,1)$

The graph corresponding to the sequence $d''' = (1,1,1,1,1,1)$ is given below.

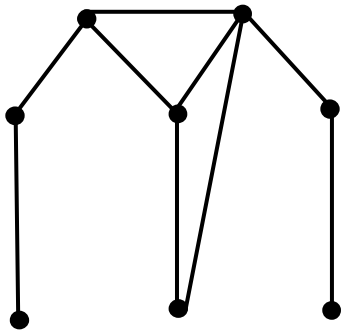


Using the Havel-Hakimi Theorem, d''' is graphic $\Rightarrow d''$ is graphic $\Rightarrow d'$ is graphic $\Rightarrow d$ is graphic.

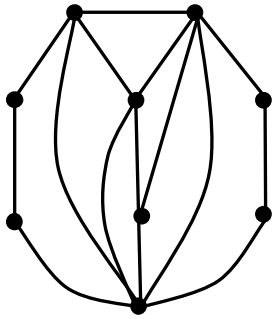
To obtain a graph corresponding to $d'' = (2,2,2,1,1,1,1)$, we add a vertex adjacent to two vertices with degrees $1,1$ in the graph above, and we obtain the graph as shown below:



To obtain a graph corresponding to sequence $d' == (4,3,3,2,2,2,1,1)$, add a vertex adjacent to four vertices with degrees 2,2,1,1 in the graph above. The resulting graph is:



To obtain a graph corresponding to sequence $d = (6,5,4,4,3,2,2,2,2)$, add a vertex adjacent to the six vertices with degrees 4,3,3,2,1,1 in the graph above. The resulting graph is:



***

You must have noted that the process shown in Example16 above does not produce a unique graph.

Now, you can try following exercises.

E19) Check whether the following sequences are graphic or not.

i) $(5,5,4,3,2,2,2,1)$ ii) $(5,5,4,4,2,2,1,1)$

iii) $(6,6,5,5,3,2,2,1,1)$ iv) $(5,5,5,4,2,1,1,1)$

E20) If $(d_1,d_2,\ldots,d_n)$ is a graphic sequence, then so is $(d_1+1,d_2+1,\ldots,d_n+1)$. True or false? Justify.

E21) Let $m<n$. If $(d_1,d_2,\ldots,d_m)$ and $(d_{m+1},d_{m+2}\ldots,d_n)$ are graphic sequences, then is $(d_1,d_2,\ldots,d_n)$ a graphic sequence? Justify your answer.

E22) Draw two nonisomorphic graphs with degree sequence $(2,2,2,2,2,2)$.

E23) Draw a graph having the given properties or explain why no such graph exists.

i) multigraph with four vertices of degree $1,1,2$ and 3.

ii) multigraph with four vertices of degree $1,1,3$ and 3.

iii) graph with four vertices of degree $1,1,3$ and 3.

With this we come to the end of this section, and also to this unit. We shall now summarise the important points discussed in this unit.

2.7 SUMMARY

In this unit, we have covered the following points:

1. Distance between two vertices of a graph, and its applications.
2. Special graphs such as Petersen graph and hypercube graph.

3. Graph representation through adjacency and incidence matrices.
4. Some properties of adjacency matrices.
5. Graph isomorphic and self-complementary graphs.
6. Degree sequences and graphic sequences.
7. Havel-Hakimi Theorem.

2.8 SOLUTION/ ANSWERS

E1) We can compute the eccentricities of the vertices as follows:

$\varepsilon(v_1)=4,\ \varepsilon(v_2)=3,\ \varepsilon(v_3)=3,\ \varepsilon(v_4)=3,\ \varepsilon(v_5)=4,\ \varepsilon(v_6)=4,\ \varepsilon(v_7)=3,$

$\varepsilon(v_8)=3,\ \varepsilon(v_9)=3,\ \varepsilon(v_{10})=3,\ \varepsilon(v_{11})=4$

Thus the diameter is 4, and the radius is 3.

E2) i) Note that every vertex of K_n has eccentricity 1, which implies $\text{diam}(K_n)=\text{rad}(K_n)=1$.

ii) Let $(v_0,v_1,v_2,\ldots,v_{n-1},v_0)$ be a representation of C_n. There are two cases according to whether n is odd or even.

Case 1: n is odd

In this case we have for all $1\le i\le\frac{n-1}{2}$, $d(v_i,v_{i+(n-1)/2})=\frac{n-1}{2}$. Thus for $1\le i\le\frac{n-1}{2}$, the maximum distance of v_i from any other vertex is $\frac{n-1}{2}$, i.e., $\varepsilon(v_i)=\frac{n-1}{2}$. Since the distance function is symmetric, for $1\le i\le\frac{n-1}{2}$, we also have $d(v_{i+(n-1)/2},v_i)=\frac{n-1}{2}$. This implies for $\frac{n-1}{2}\le i\le n-1,\ \varepsilon(v_i)=\frac{n-1}{2}$. Thus

$\text{diam}(C_n)=\text{rad}(C_n)=\frac{n-1}{2}$.

Case 2: When n is even.

In this case we have for $0\le i\le\frac{n}{2}-1,\ d(v_i,v_{i+n/2})=\frac{n}{2}$, which is the farthest distance of v_i from any other vertex. Thus for $0\le i\le\frac{n}{2}$, $\varepsilon(v_i)=\frac{n}{2}$. Likewise for $\frac{n}{2}\le i\le n-1,\ \varepsilon(v_i)=\frac{n}{2}$. Thus we have

$\text{diam}(C_n)=\text{rad}(C_n)=\frac{n}{2}$.

Combining both the cases we can write

$$\text{diam}(C_n) = \text{rad}(C_n) = \begin{cases} \dfrac{n-1}{2}, & \text{if } n \text{ is odd} \\ \dfrac{n}{2}, & \text{if } n \text{ is even.} \end{cases}$$

iii) Let $(v_1, v_2, \ldots, v_n)$ be the path P_n. The vertices v_1 and v_n are at the maximum distance, which is $n-1$. Therefore, $\text{diam}(P_n) = n-1$.

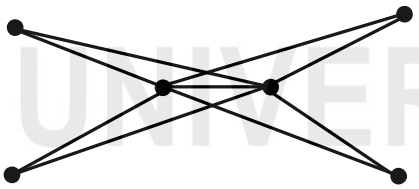
To find the radius we consider two cases according to whether n is even or odd.

When n is odd, $v_{(n+1)/2}$ is the only vertex with minimum eccentricity $\dfrac{n-1}{2}$, and hence $\text{rad}(P_n) = \dfrac{n-1}{2}$. On the other hand, when n is even the vertices of minimum eccentricity are $v_{n/2}$ and $v_{n/2+1}$, and $\varepsilon(v_{n/2}) = \varepsilon(v_{n/2+1}) = \dfrac{n}{2} - 1$.

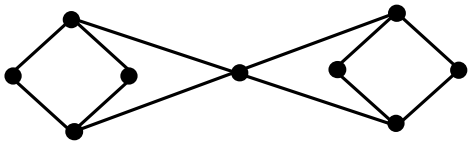
Hence $\text{rad}(P_n) = \dfrac{n}{2} - 1$.

iv) We know that the maximum distance between any two vertices of $K_{m,n}$ is 2. Hence $\text{diam}(K_{m,n}) = 2$. Also we know that every vertex has eccentricity 2. Therefore, $\text{rad}(K_{m,n}) = 2$.

E3) i) One such graph is given below, with radius and diameter equal to 2.



ii) One such graph is given below, with radius 2 and diameter 4.

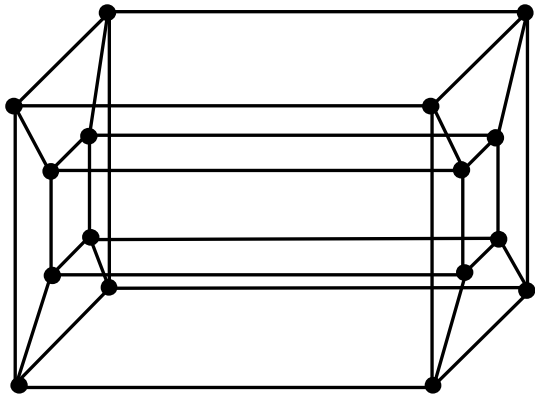


E4) No. Because $\text{diam}(G) \leq 2\ \text{rad}(G)$ for all graphs G.

E5) The Petersen graph is not bipartite, because it contains a 5-cycle, which is odd.

E6) Note that any two nonadjacent vertices of the Petersen graph have a common neighbour. Hence, its diameter is 2. To find the radius, let us compute the eccentricity of a vertex u. There are 3 vertices adjacent to it. The remaining 6 vertices are nonadjacent to u, and hence each of them has a common neighbour with u. Therefore, $\varepsilon(u) = 2$. Since u is arbitrary, every vertex has eccentricity 2. Thus the radius of the graph is 2.

E7) A drawing of Q_4 is given below.



E8) Let us take the vertex and edge labellings in the order $V(G)=\{v_1,v_2,v_3,v_4,v_5,v_6\}$, and $E(G)=\{e_1,e_2,e_3,\ldots,e_9\}$. Then the adjacency and incidence matrices are as follows.

$$A(G)=\begin{bmatrix}0&1&0&0&0&1\\1&0&1&1&0&1\\0&1&0&1&0&0\\0&1&1&0&1&1\\0&0&0&1&0&1\\1&1&0&1&1&0\end{bmatrix},\ B(G)=\begin{bmatrix}1&0&0&0&0&1&0&0&0\\1&1&0&0&0&0&1&0&1\\0&1&1&0&0&0&0&0&0\\0&0&1&1&0&0&0&1&1\\0&0&0&1&1&0&0&0&0\\0&0&0&0&1&1&1&1&0\end{bmatrix}$$

E9) Since A is the adjacency matrix, we know that $a_{ij}=1$ iff $v_i \leftrightarrow v_j$ iff $v_j \leftrightarrow v_i$ iff $a_{ji}=1$. Therefore, $a_{ij}=a_{ji}$ for all i,j. Hence A is symmetric. Assume that A^k is symmetric for some k. Then $a_{ij}^{(k)}=a_{ji}^{(k)}$ for all i,j. Now $a_{ij}^{(k+1)}=\sum_{r=1}^{n}a_{ir}^{(k)}a_{rj}=\sum_{r=1}^{n}a_{ri}^{(k)}a_{jr}=\sum_{r=1}^{n}a_{jr}a_{ri}^{(k)}=a_{ji}^{(k+1)}$

Thus $A^{(k+1)}$ is also symmetric. Hence by the Principle of Mathematical Induction, A^k is symmetric for all $k\geq 1$.

E10) i) Note that the adjacency matrix A represents a multigraph. We can see that

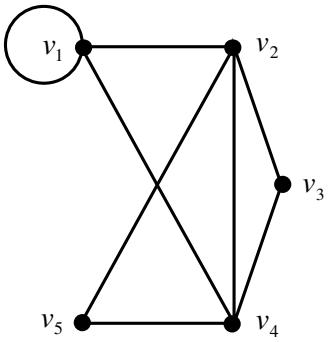
$$M=A+A^2+A^3=\begin{bmatrix}1&1&0&1\\1&1&0&1\\0&0&1&0\\1&1&0&1\end{bmatrix}+\begin{bmatrix}3&3&0&3\\3&3&0&3\\0&0&1&0\\3&3&0&3\end{bmatrix}+\begin{bmatrix}9&9&0&9\\9&9&0&9\\0&0&1&0\\9&9&0&9\end{bmatrix}$$

$$=\begin{bmatrix}13&13&0&13\\13&13&0&13\\0&0&3&0\\13&13&0&13\end{bmatrix}$$

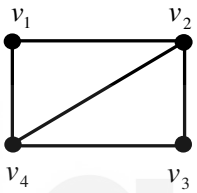
We note that the $(1,3)^{rd}$ entry of M is 0. Hence the multigraph is disconnected.

ii) Attempt as in part i). The graph is connected.

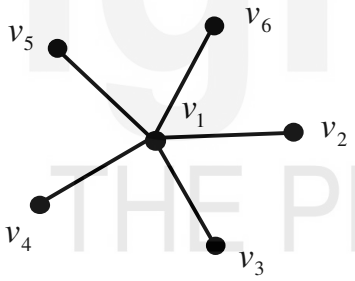
E11) i) Let $A=[a_{ij}]$ be the given adjacency matrix. Since $a_{11}=1$ it represents a multigraph. Let $V(G)=\{v_1,v_2,v_3,v_4,v_5\}$ be the vertex set of the multigraph. Drawing them on a circle we get



ii) The graph with labelling $V(G)=\{v_1,v_2,v_3,v_4\}$ is drawn below.



E12) Let us label the vertices of the graph as follows:



With respect to the labelling $V(G)=\{v_1,v_2,v_3,v_4,v_5,v_6\}$ the adjacency matrix is

$$A=\begin{bmatrix}0&1&1&1&1&1\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\end{bmatrix}$$

Note that between any pair of vertices of the set $\{v_2,v_3,v_4,v_5,v_6\}$, there are the same number of walks of length 4. So, let us choose the vertices v_2 and v_3. Then, we have to compute the $(2,3)^{rd}$ entry of A^4. We have

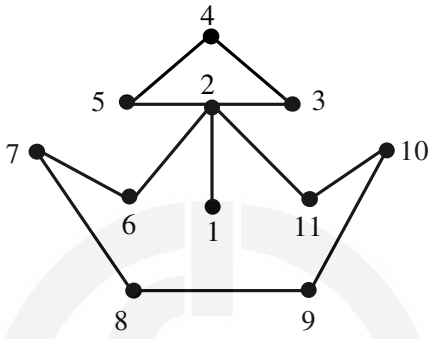
$$A^2=\begin{bmatrix}*&*&*&*&*&*\\0&1&1&1&1&1\\*&*&*&*&*&*\\*&*&*&*&*&*\\*&*&*&*&*&*\\*&*&*&*&*&*\end{bmatrix},\qquad A^3=\begin{bmatrix}*&*&*&*&*&*\\5&0&0&0&0&0\\*&*&*&*&*&*\\*&*&*&*&*&*\\*&*&*&*&*&*\\*&*&*&*&*&*\end{bmatrix},\text{ and}$$

$$A^4 = \begin{bmatrix} * & * & * & * & * & * \\ 0 & 5 & 5 & 5 & 5 & 5 \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{bmatrix}$$

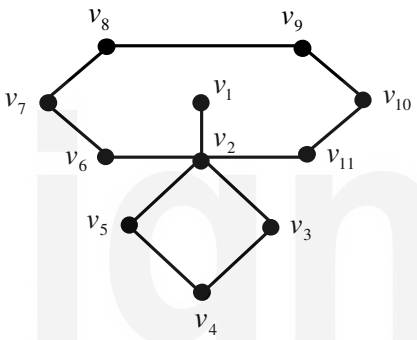
Thus $(2,3)^{rd}$ entry of A^4 is 5, which is the numbers of walks of length 4 between v_2 and v_3.

E13) i) We can see that $\Delta G_1) = 5$ and $\Delta(G_2) = 4$. Hence $G_1 \ncong G_2$.

ii) Let us label the given graphs as



G_1



G_2

Now let us define the map $\phi : V(G_1) \to V(G_2)$ by $\phi(i) = v_i$ for $i \in \{1,2,3,\ldots,11\}$. By definition, ϕ is one-one onto. Now

$12 \in E(G_1) \Leftrightarrow \phi(1)\phi(2) = v_1 v_2 \in E(G_2)$

$23 \in E(G_1) \Leftrightarrow \phi(2)\phi(3) = v_2 v_3 \in E(G_2)$

$34 \in E(G_1) \Leftrightarrow \phi(3)\phi(4) = v_3 v_4 \in E(G_2)$

$45 \in E(G_1) \Leftrightarrow \phi(4)\phi(5) = v_4 v_5 \in E(G_2)$

$25 \in E(G_1) \Leftrightarrow \phi(2)\phi(5) = v_2 v_5 \in E(G_2)$

$26 \in E(G_1) \Leftrightarrow \phi(2)\phi(6) = v_2 v_6 \in E(G_2)$

$67 \in E(G_1) \Leftrightarrow \phi(6)\phi(7) = v_6 v_7 \in E(G_2)$

$78 \in E(G_1) \Leftrightarrow \phi(7)\phi(8) = v_7 v_8 \in E(G_2)$

$89 \in E(G_1) \Leftrightarrow \phi(8)\phi(9) = v_8 v_9 \in E(G_2)$

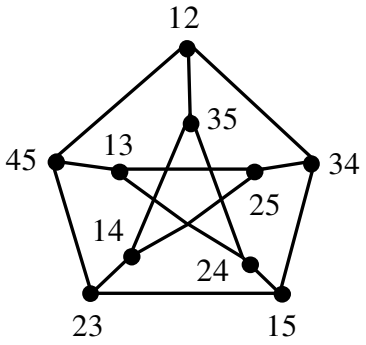
$9(10) \in E(G_1) \Leftrightarrow \phi(9)\phi(10) = v_9 v_{10} \in E(G_2)$

$10(11) \in E(G_1) \Leftrightarrow \phi(10)\phi(11) = v_{10} v_{11} \in E(G_2)$

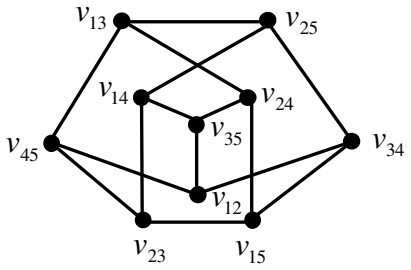
$2(11) \in E(G_1) \Leftrightarrow \phi(2)\phi(11) = v_2 v_{11} \in E(G_2).$

Thus for all $uv \in E(G_1)$, we have shown that $\phi(u)\,\phi(v) \in E(G_2)$.
Therefore, ϕ is an isomorphism from G_1 to G_2, and hence $G_1 \cong G_2$.

E14) Let us draw the first graph as it is, and the second one with the labels as shown below.



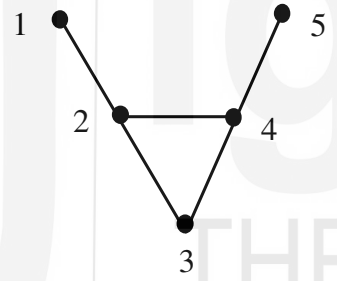
G_1



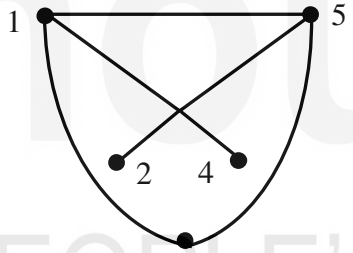
G_2

Now define the map $\phi : V(G_1) \to V(G_2)$ by $\phi(k) = v_k$, for $k = \{12,13,14,15,23,24,25,34,35,45\}$. Now verify that ϕ is an isomorphism from G_1 to G_2. Note that G_1 and G_2 are two different drawings of the Petersen graph.

E15) i) Labelling the graph with $1,2,3,4,5,$ and drawing its complement we get



G

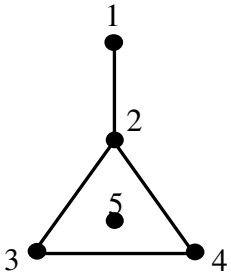


$\overline{G}$

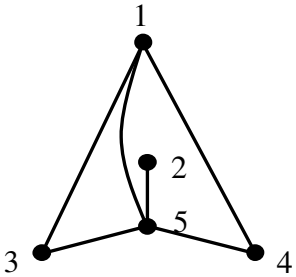
Let us define $\phi : V(G) \to V(\overline{G})$ by
$\phi(1) = 2,\ \phi(2) = 5,\ \phi(3) = 3,\ \phi(4) = 1,\ \phi(5) = 4$

Now verify that ϕ is an isomorphism from G to $\overline{G}$. So, $G \cong \overline{G}$, and hence G is self-complementary.

ii) Labelling the vertices as $1,2,3,4,5,$ and drawing the complement we get



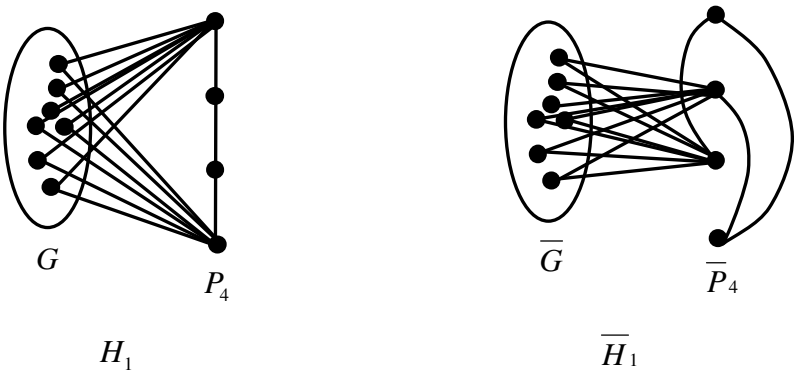
G



$\overline{G}$

We find that G is disconnected, whereas $\overline{G}$ is connected. Therefore, $G \not\cong \overline{G}$, and hence G is not self-complementary.

E16) A geometrical depiction of H_1 and $\overline{H}_1$ is given below



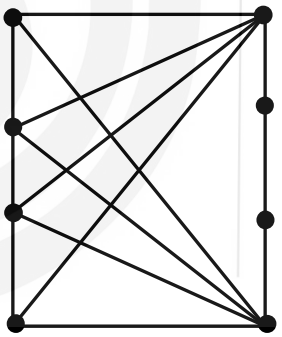
H_1 $\overline{H}_1$

We have $G \cong \overline{G}$ and $P_4 \cong \overline{P_4}$. Thus $H_1 \cong \overline{H}_1$. Hence H_1 is self-complementary.

Use similar reasoning to prove that H_2 is self-complementary.

E17) We are given a self-complementary n-vertex graph G, and we have to prove that either n or $n-1$ is divisible by 4. By definition, we know that $m(G)+m(\overline{G})=\frac{n(n-1)}{2}$. Since $G \cong \overline{G}$, we have $m(G)=m(\overline{G})$. Hence $m(G)=\frac{n(n-1)}{4}$. This implies that either n or $n-1$ is divisible by 4.

E18) Taking $G=P_4$, and using the construction of the graph H_1 as shown in E16, we get the following self-complementary graph on 8 vertices.



P_4 P_4

E19) i) We have $d=(5,5,4,3,2,2,2,1)$

$d'=(4,3,2,2,1,1,1)$

$d''=(2,1,1,1,1,0)$

$d'''=(1,1,0,0,0)$

We can see that d''' is graphic as it realises to the following graph.

Thus d''' is graphic $\Rightarrow$ d'' is graphic $\Rightarrow$ d' is graphic $\Rightarrow$ d is graphic.

ii) Here $d=(5,5,4,4,2,2,1,1)$

$d' = (4,3,3,1,1,1,1)$

$d'' = (2,2,1,1,0,0)$

$d''' = (1,1,0,0,0)$

Since d''' is graphic, d is graphic.

iii) Here $d = (6,6,5,5,3,2,2,1,1)$

$d' = (5,4,4,2,1,1,1,1)$

$d'' = (3,3,1,1,1,0,0)$

$d''' = (2,1,0,0,0,0)$

$d'''' = (0,0,0,0,-1)$

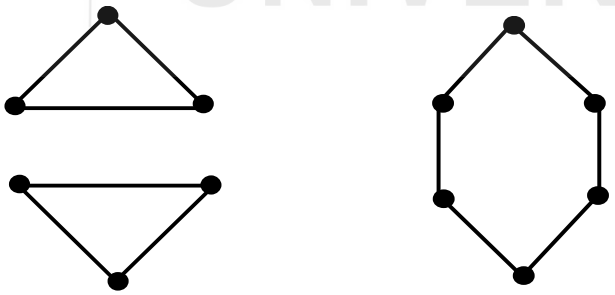
We find that d'''' contains a negative entry. Hence d'''' is not graphic $\Rightarrow$ d''' is not graphic $\Rightarrow$ d'' is not graphic $\Rightarrow$ d' is not graphic $\Rightarrow$ d is not graphic.

iv) Here we find that $d''' = (1,0,0,0,-1)$, which is not graphic. Hence d is not graphic.

E20) False. Because, for example we know that $(1,1)$ is graphic but $(2,2)$ is not.

E21) Since $(d_1, d_2, \ldots, d_n)$ is graphic there exists an m-vertex graph, say G_1, with $V(G_1) = \{v_1, v_2, \ldots, v_m\}$ where $d(v_i) = d_i$ for $1 \le i \le m$. Likewise, since $(d_{m+1}, d_{m+2}, \ldots, d_n)$ is graphic, there exists a graph, say G_2, with $V(G_2) = \{v_{m+1}, v_{m+2}, \ldots, v_n\}$, where $d(v_i) = d_i$ for $m+1 \le i \le n$. Then the graph $G_1 \cup G_2$ with $V(G_1 \cup G_2) = \{v_1, v_2, \ldots, v_n\}$ realises to the sequence $(d_1, d_2, \ldots, d_n)$. Hence $(d_1, d_2, \ldots, d_n)$ is graphic.

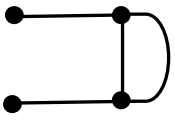
E22) Two such graphs are



G_1 G_2

E23) i) No such multigraph exists because $3+2+1+1=7$, which is odd.

ii) One such graph is



iii) Here $d = (3,3,1,1)$. So, $d' = (2,0,0)$, which is not graphic. Hence d is not graphic, and so no such graph exists.

UNIT 3

TREES

Structure **Page Nos.**

3.1 INTRODUCTION

In the previous unit, we have concentrated on some basics of graphs. In this unit we shall focus on one particularly important and useful type of graph, called 'tree'. The word "tree" suggests branching out from a root and never completing a cycle. Although trees are relatively simple structures, they form the basis of many of the practical techniques used to model and to design large scale systems. In chemistry, for example, the chemical compounds such as Methane, Ethane etc. can be represented by trees. Trees provide a range of useful applications as simple as a family tree to as complex as trees in data structures of computer science. Trees as graphs have many applications, especially in data storage, searching and communication.

In Section 3.2, we shall discuss the basic properties and characterisation of trees. Section 3.3 is devoted to spanning trees and their enumeration. In Section 3.4, we shall discuss k-ary trees, binary trees and their properties.

Objectives

After studying this unit, you should be able to:

- distinguish trees from other graphs;
- state several properties of trees and give equivalent definitions of a tree;
- find different spanning trees of a graph;
- understand k-ary trees, binary trees and their properties.

3.2 TREE AND ITS CHARACTERISATION

In this section, we shall discuss what we mean by a tree, some properties of trees and a few characterisations.

The concept of a tree is one of the most important and commonly used ideas in graph theory. It arose in connection with the work of Gustav Kirchhoff on electrical networks in 1840, and later with Cayley's work on the enumeration of molecules in 1870. He was engaged in the enumeration of the isomers of saturated hydrocarbons $C_n H_{2n+2}$ with a given number n of carbon atoms (see Fig. 1)

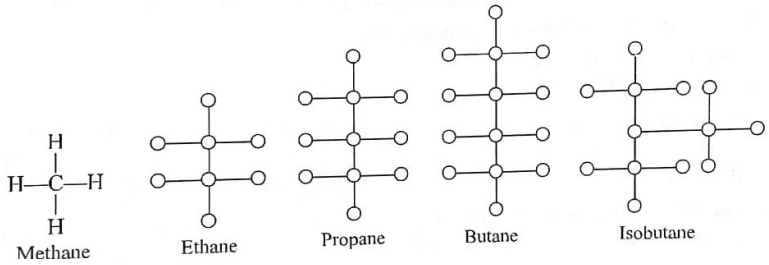


Fig. 1

More recently, trees have proved to be useful in areas such as computer science, decision making, linguistics and the design of gas pipeline systems.

So, let us start with a formal definition.

Definition: A **tree** is a connected graph that has no cycles.

For example, all graphs in Fig. 2 are trees.

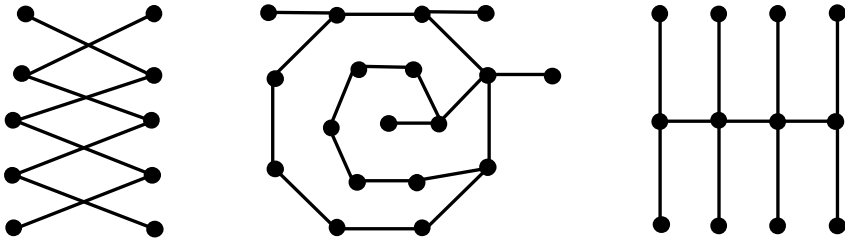
Fig. 2

Note that a trivial graph is a tree.

Note that a graph is called **acyclic** if it contains no cycles. Hence we say that a connected acyclic graph is called a **tree**. Interestingly, an acyclic graph is called a **forest**. Therefore, every component of a forest is a tree.

Let us consider an example.

Example 1: Which of the following graphs are trees? Why?



G_1 G_2 G_3

Solution: You can see that G_1 is not a tree because it is disconnected.

G_2 is not a tree because it contains a cycle. Can you find a cycle in it?

G_3 is a tree because it is connected and has no cycles.

***

From Unit 1, you can recall that a leaf is not a cut-vertex. This means, if v is a leaf of a tree T, then $T-v$ is connected. Also $T-v$ is acyclic, as removal of a vertex does not create a cycle. Thus $T-v$ is a tree. In general, you can prove that $T-v$ is a forest for any vertex v of T.

Next we shall prove the following theorem.

Theorem 1: Every tree with n vertices, where $n \geq 2$, has at least two leaves.

Proof: Consider a maximal path P in T. If length $(P)=2$ the endpoints of P are two leaves in T. If length $(P) \geq 3$, then let $P=(u, w_1, w_2, \ldots, w_k, v)$, for some $k \geq 1$. Let $w \in V(T) \setminus V(P)$.

Now ask to yourself: can w be a neighbour of u, or of v? If $w \in N(u)$, then $P+w$ is a path longer than P, which is impossible. Hence $w \notin N(u)$. Similarly, $w \notin N(v)$. Also note that u has no neighbour other than w_1 in P. Similarly, v has no neighbour other than w_k in P. (Why?) Thus u and v are leaves. This completes the proof. ■

Proposition 1: Every edge of a tree is a cut-edge.

Proof: Recall from Unit 1 that an edge is a cut-edge if and only if it belongs to no cycles. Since a tree has no cycles every edge of a tree is a cut-edge. ■

Trees have many equivalent characterisations, any of which could be taken as the definition. Such characterisations are useful because we need only verify that a graph satisfies any one of them to prove that it is a tree. The Following theorem characterises trees.

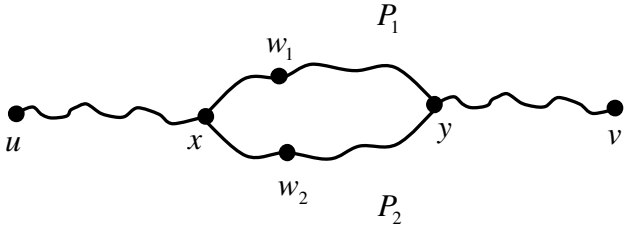
Theorem 2: Let T be a graph on n vertices. The following statements are equivalent

i) T is a tree.

ii) There is a unique path between any two vertices of T.

iii) T is connected and $m(T)=n-1$.

Proof: i) $\Rightarrow$ ii) We are given that T is a tree. Consider two vertices u and v in T such that P_1 and P_2 are two distinct (u,v)-paths in T. If u and v are the only common vertices of P_1 and P_2, then $P_1=P_2$. So, let x be a vertex such that the (u,x)-path lying in both P_1 and P_2 is maximal. (See the picture below.)



If $x = v$, then $P_1 = P_2$ again, which is not possible. So, there exist $w_1, w_2 \in N(x)$ such that $w_1 \in V(P_1)$ and $w_2 \in V(P_2)$. Now consider two subpaths $P_1' = (x, w_1, \ldots, v)$ and $P_2' = (x, w_2, \ldots, v)$ in $P_1 \cup P_2$. Then P_1' and P_2' must have a common vertex, in addition to x. Let y be the first common vertex of P_1' and P_2' after x. Then the paths $(x, w_1, \ldots, y)$ and $(x, w_2, \ldots, y)$ form a cycle, which is a contradiction.

ii) $\Rightarrow$ iii) Since there is a unique path between every two vertices of T, T is connected. Also T is a cyclic. (Why?) This means, T is a tree. Now we show that the number of edges in T is $n-1$ by the Principle of Mathematical Induction on n, the number of vertices of T.

If $n = 1$, then $m(T) = 0 = n-1$. Now, assume the result holds for all the graphs with fewer than n vertices. Let T be a connected graph on n vertices such that every pair of vertices is connected by a unique path in T. By Theorem 1, there exists a leaf v in T. Then $T - v$ is a tree with $n-1$ vertices. Hence by the induction hypothesis.

$$m(T - v) = (n-1) - 1 = n - 2 .$$

Since T has only one edge more than $T - v$, we have $m(T) = m(T - v) + 1 = n - 1$.

(iii) $\Rightarrow$ i) We are given that T is connected, and $m(T) = n - 1$. To show that T is a tree, we have to just show that T is acyclic. So, assume if possible, that $T_0 = T$ has a cycle C_0. Let e_0 be some edge of C_0. Then $T_1 = T - e_0$ is connected as e_0 is not a cut-edge of T. If T_1 has a cycle C_1, then pick an edge e_1 of C_1. We can see that $T_2 = T_1 - e_1$ is connected as e_1 is not a cut-edge of T_1. We can continue this process as long as the subgraph $T_k = T_{k-1} - e_{k-1}$ for $k \geq 1$, has a cycle. But T has at most finitely many cycles. Hence for some k, T_k must be acyclic, as well as connected, and hence a tree. We have already proved i) $\Rightarrow$ ii) and ii) $\Rightarrow$ iii). Therefore we have i) $\Rightarrow$ iii). This means $m(T_k) = n - 1 = m(T)$, which implies $k = 0$. Therefore, T is acyclic, and hence a tree. ■

Let us look at an example now.

Example 2: Consider a tree T with 3 vertices of degree 2, 4 vertices of degree 3, and 3 vertices of degree 4. Calculate the number of leaves in T.

Solution: Let the number of leaves in T be k. Total number of vertices in T is $k + 3 + 4 + 3 = (10 + k)$ and hence $m(T) = k + 9$

By the Handshaking Lemma,

$$\sum_{v \in V(T)} d(v) = 2m(T) \Rightarrow k + 3 \times 2 + 4 \times 3 + 3 \times 4 = 2(k + 9) \Rightarrow k = 12$$

Hence, T has12 leaves.

***

Apart from the above characterisations, the degree sequence of a graph can also tell us whether the graph is a tree or not. To see how consider the following theorem.

Theorem 3: Let G be a connected graph with at least two vertices and degree sequence $(d_1, d_2, ..., d_n)$, $d_i \geq 1$ for each i. Then G is a tree if and only if $\sum_{i=1}^{n} d_i = 2(n-1)$.

Proof: Let G be a tree with at least two vertices and degree sequence $(d_1, d_2, ..., d_n)$, $d_i \geq 1$ for each i. Then by Theorem 2, G has $(n-1)$ edges. So by the Handshaking Lemma, $\sum_{i=1}^{n} d_i = 2(n-1)$.

Conversely, Let G be a graph with at least two vertices and degree sequence $(d_1, d_2, ..., d_n)$ such that $\sum_{i=1}^{n} d_i = 2(n-1)$.

This implies $2m(G) = 2(n-1) \Rightarrow m(G) = n-1$. Hence by Theorem 2, G is a tree. ■

Example 3: Check whether there exists a tree with degree sequence $(7,3,2,1,1,1,1,1)$.

Solution: Let T be a tree with the degree sequence $(7,3,2,1,1,1,1,1)$ So $n(T) = 9$, and $m(T) = 8$.

By Theorem 3 above, $\sum_{i=1}^{9} d_i = 18 = 2 \times 8$, which is false. Hence, there is no tree with the given degree sequence.

***

Now, you can try the following exercises.

E1) Which of the following statements are true, and which are false? Justify your answers.

i) A connected forest is a tree.

ii) Every bipartite graph is a forest.

iii) If T is a tree, then $\overline{T}$ is a forest.

iv) There exists a tree with all vertices as leaves.

v) If e is an edge of a tree T, then $T - e$ is a forest.

E2) Let T be a tree with 50 edges. The removal of certain edge from T yields two disjoint trees T_1 and T_2. Given that the number of vertices in T_1 equals the number of edges in T_2, determine the number of vertices and the number of edges in T_1 and T_2.

E3) What is the maximum possible number of leaves in a tree with n vertices? Is there any other name for such a tree?

E4) Is it possible to draw a tree with the degree sequence $(4,2,2,1,1)$?

E5) Show that every tree is a bipartite graph.

E6) Let T_1 and T_2 be two trees in a graph sharing exactly one vertex. Is $T_1 \cup T_2$ a tree? Justify.

E7) Let G be a graph such that $m(G) < n(G)$. Show that G has a component that is a tree.

E8) Let T be a tree. Show that for every pair of distinct nonadjacent vertices u and v of T, $T + uv$ has exactly one cycle.

In the next section we shall discuss spanning trees in graphs.

3.3 SPANNING TREE

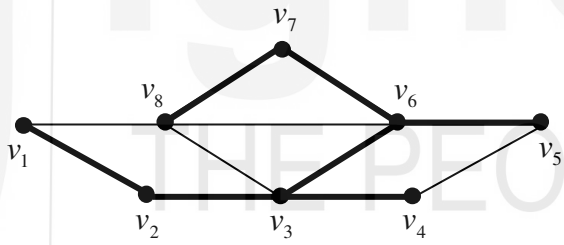
Recall that a spanning subgraph of a graph G is the subgraph that contains all the vertices of G. In connected graphs, the concept of spanning tree arises naturally. So, first let us look at the definition of "spanning tree".

Definition: Let G be a graph. A subgraph T of G is said to be a **spanning tree** of G, if

i) T is a spanning subgraph of G,

ii) T is a tree.

For example, the following graph G has a spanning tree T shown in bold edges. Here $V(T) = V(G)$ and $E(T) = \{v_1v_2, v_2v_3, v_3v_4, v_3v_6, v_5v_6, v_6v_7, v_7v_8\}$.



Just replace v_7v_8 by v_1v_8 to get another spanning tree.

Let us consider the following result that characterises connected graphs through spanning trees.

Proposition 2: A graph is a connected graph if and only if it contains a spanning tree.

Proof: Let G be a connected graph. If G is acyclic, G itself is a spanning tree of G. If G contains a cycle C then delete one edge, say e, from C. Since no edge of a cycle is a cut-edge, $G - e$ is connected. Repeat this process until you find an acyclic subgraph H of G. Graph H also contains all vertices of G. Hence, H is a spanning tree of G.

Conversely if G contains a spanning tree then clearly, their exists a path between every pair of vertices of G. Hence G is a connected graph. ■

From Proposition 2, it is clear that disconnected graphs do not have spanning trees.

Now, using some results discussed in Section 3.2, we can prove some more results about pairs of spanning trees as follows:

Proposition 3: If T, and T' are spanning trees of a connected graph G and $e \in E(T) \setminus E(T')$, then there is an edge $e' \in E(T') \setminus E(T)$ such that $T - e + e'$ is a spanning tree of G.

Proof: Suppose $e \in E(T) \setminus E(T')$. Since e is a cut edge of T, $T - e$ is disconnected. Let H_1 and H_2 be the components of $T - e$ and let $u \in H_1, v \in H_2$. Then T' contains a (u,v)-path. This path contains an edge e' with one endpoint in H_1 and the other endpoint in H_2. Hence, H_1 and H_2 with e' form a connected acyclic subgraph of G, i.e., $T - e + e'$ is a spanning tree of G. ■

Proposition 4: If T and T' are spanning trees of a connected graph G and $e \in E(T) \setminus E(T')$, then there is an edge $e' \in E(T') \setminus E(T)$ such that $T' + e - e'$ is a spanning tree of G.

Proof: From E8) you know that $T' + e$ contains exactly one cycle, say C. Note that all the edges of C cannot be in T. Hence there exists some edge e' in C, but not in T. Then $T' + e - e'$ is connected and acyclic containing all vertices of G. Hence, $T' + e - e'$ is a spanning tree. ■

We shall write $\tau(G)$ to denote the number of spanning trees of a labelled graph G. Note that $\tau(G) \geq 1$, whenever G is connected. For a disconnected graph G, we write $\tau(G) = 0$. Now we state, without proof, the theorem that gives us a formula for $\tau(G)$.

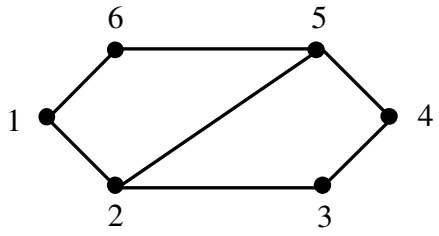
Theorem 4 (Matrix Tree Theorem): Let G be a connected graph with vertex set $V(G) = \{v_1, v_2, v_3, ..., v_n\}$, and adjacency matrix $A(G) = (a_{ij})_{n \times n}$. Define a matrix

$$Q = \begin{bmatrix} d(v_1) & -a_{12} & -a_{13} & \cdots & -a_{1n} \\ -a_{21} & d(v_2) & -a_{23} & \cdots & -a_{2n} \\ \vdots & & \ddots & & \vdots \\ -a_{n1} & -a_{n2} & \cdots & & d(v_n) \end{bmatrix}$$

Let Q_{rs} be the submatrix of Q obtained by deleting the row r and the column s. Then $\tau(G) = (-1)^{r+s} \det(Q_{rs})$, for any row r and any column s.

Note that the quantity $(-1)^{r+s} \det(Q_{rs})$ is same for all $r, s \in \{1,2,\ldots,n\}$. So, while computing the number of spanning trees we can choose any value of r and s, convenient to us. Let us have an example.

Example 4: Find the number of spanning trees of the graph given below. Also draw them.



Solution: We have

$$Q = \left[\begin{array}{c|c|ccc|c} 2 & -1 & 0 & 0 & 0 & -1 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ \hline 0 & 1 & 0 & -1 & 3 & -1 \\ \hline -1 & 0 & 0 & 0 & -1 & 2 \end{array}\right]$$

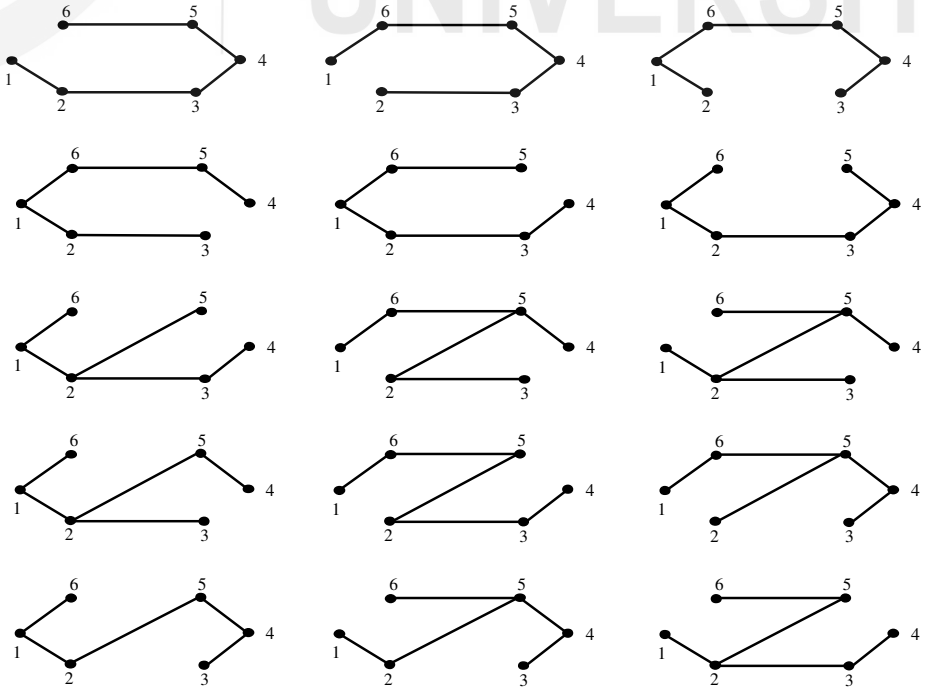
Let us delete the 5^{th} row and the 2^{nd} column from Q. So we have

$$\tau(G) = (-1)^{5+2} det\,(Q_{52}) = -\begin{vmatrix} 2 & 0 & 0 & 0 & -1 \\ -1 & -1 & 0 & -1 & 0 \\ 0 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 0 & 0 & -1 & 2 \end{vmatrix} \quad (C_5 \to C_5 + \frac{1}{2}C_1)$$

$$= -\begin{vmatrix} 2 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & -1 & -\frac{1}{2} \\ 0 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 0 & 0 & -1 & \frac{3}{2} \end{vmatrix} = -2\begin{vmatrix} -1 & 0 & -1 & -\frac{1}{2} \\ 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & \frac{3}{2} \end{vmatrix} \quad (R_1 \to R_1 + \frac{1}{3}R_4)$$

$$= -2\begin{vmatrix} -1 & 0 & -\frac{4}{3} & 0 \\ 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & \frac{3}{2} \end{vmatrix} = 15$$

All the 15 spanning trees are drawn below.



***

Using the Matrix Tree Theorem, we can find the number of spanning trees of any graph. However, for complete graphs we can even get a simpler formula as you can see in the corollary given below.

Corollary 1: For any labeled complete graph K_n, $\tau(K_n) = n^{n-2}$.

Proof: We know that each vertex of K_n has degree $n-1$. So

$$Q = \begin{bmatrix} n-1 & -1 & -1 & \ldots. & -1 \\ -1 & n-1 & -1 & \ldots. & -1 \\ -1 & -1 & n-1 & \ldots & -1 \\ \vdots & \vdots & \ddots & & \vdots \\ -1 & -1 & -1 & \ldots. & n-1 \end{bmatrix}$$

Using the Matrix Tree Theorem

$$\tau(\mathrm{K_n}) = (-1)^{1+1} \det(Q_{11})$$

$$= \begin{vmatrix} n-1 & -1 & \cdots & -1 \\ -1 & n-1 & \cdots & -1 \\ \vdots & & & \\ -1 & -1 & \cdots & n-1 \end{vmatrix}_{(n-1)\times(n-1)}$$

Now you can use elementary row or column operations to evaluate this determinant. You must get $\tau(K_n) = n^{n-2}$. (See E12.) ■

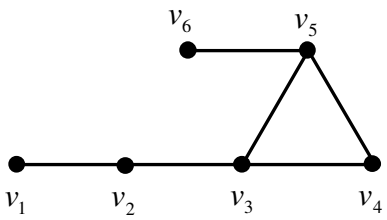
Using corollary 1 we can see, for example, $\tau(K_5) = 5^{5-2} = 125.$

Now, you can try the following exercises:

E9) If e is a cut-edge of a connected graph G, then e belongs to every spanning tree of G. True or false? Justify.

E10) Let G be a connected graph, and T a maximal acyclic subgraph of G. Show that T is a spanning tree of G.

E11) Find the number of the spanning trees of the graph G given below. Also draw the spanning trees.



E12) Complete the proof of Corollary 1, by showing that $\tau(K_n) = n^{n-2}$.

E13) How many different spanning trees do the following labelled graphs have?

i) K_3 ii) K_4 iii) $K_{2,4}$

E14) How many spanning trees does C_5 have when its vertices are

i) labelled?

ii) unlabelled?

In the next section we shall discuss k-ary trees, binary trees and their properties.

3.4 BINARY TREES, k-ARY TREES, AND THEIR PROPERTIES

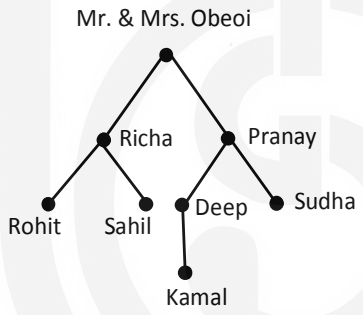
In this section, we shall, first discuss rooted trees, and then we shall come to binary and k-ary trees.

Rooted Trees

The nomenclature around trees is overwhelming. You have seen leaves in trees. Here we shall see "roots" in trees. Let us see the definition.

Definition: A tree is called a **rooted tree** if exactly one of its vertices is designated as the **root**.

We shall often write (T, r) to denote a rooted tree, where T is a tree and r is the root of T. When drawing a rooted tree, especially an unlabelled one, we need to have some way to make the root distinguishable from the remaining vertices. For example, the shape of the root can be taken to be different from the remaining vertices. The following figure shows five rooted trees each of order 5, where the root in each tree has triangle shape.



A rooted tree of Mr. and Mrs. Oberoi

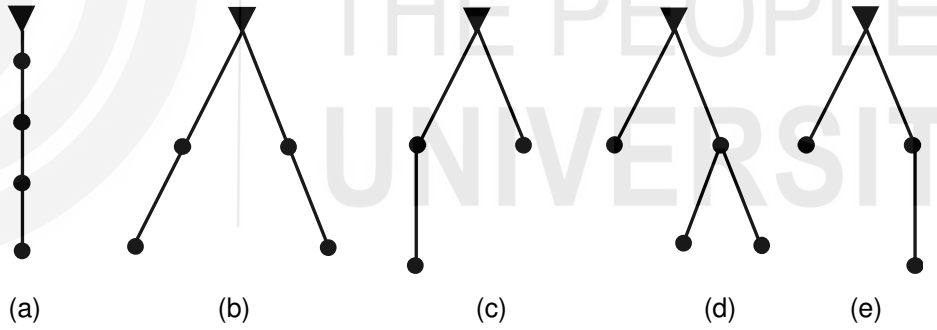


Fig. 3

Note that in Fig. $3(a)$ and Fig. 3(b) above, the two rooted trees are not isomorphic, because the roots have different degrees. However, trees in Fig, $3(c)$ and $3(e)$ are isomorphic.

Let us see how the concept of root gives rise to other important notions. In Section 3.2, you saw that there is exactly one path between any two vertices of a tree. Now consider a rooted tree (T, r). One can reach to any vertex v of T from r via the unique (r, v)-path. Any vertex on the (r, v)-path other than v is called an ancestor of v, and the neighbour of v on this path is called the **parent** of v. If u is a parent of v, we call v a **child** of u. A vertex w is called a **descendent** of v if v lies on the (r, w)-path. Two vertices are said to be **siblings** if they have the same parent. For example, look at the rooted tree T with root r drawn below.

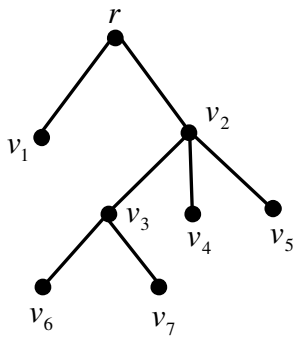
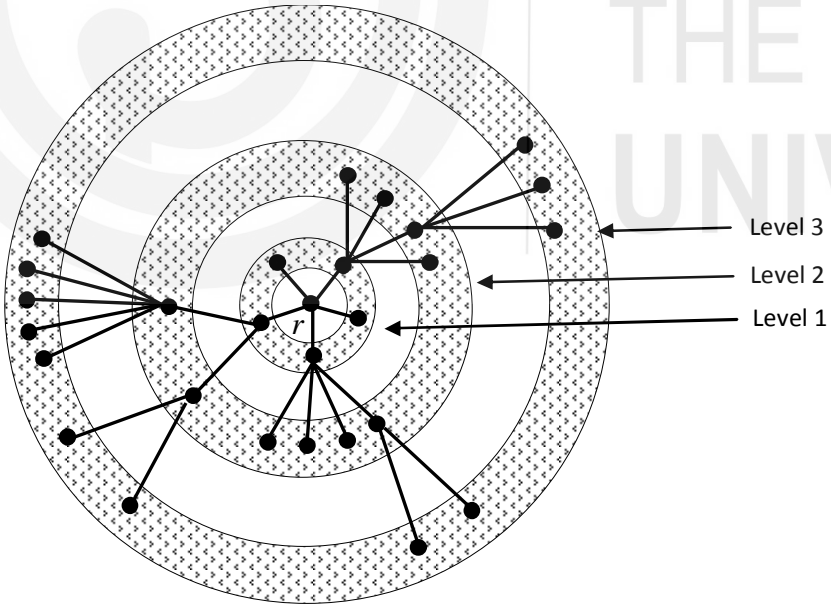


Fig. 4

Here, r is a parent of v_1 and v_2, and v_1 and v_2 are the children of r. Likewise, v_3, v_4 and v_5 are the children of v_2. Note that v_1, v_4, v_5, v_6 and v_7 have no children. The vertex v_2 is an ancestor of all the vertices from v_3 to v_7. In other words, the vertices from v_3 to v_7 are the descendents of v_2. The vertex v_3 has two ancestors, two siblings, and two children. Can you find out them?

Another interesting thing about rooted trees is that one can group vertices according to their distances from the root. We say a vertex v of a rooted tree (T, r) is at **level** ℓ if $d(v, r) = \ell$. The root r lies at the level 0. The highest level of a vertex in a rooted tree T is called the **height** of T. A rooted tree with height h is also called an h-**level** tree. For example, the tree in Fig. 4 above has height 3 as the vertices v_6 and v_7 are at the highest level, namely 3, in it. So, it is a 3-level tree. Now consider the following rooted tree (T, r).



The only vertex at level 0 is r. The vertices at level 1, 2, 3 are enclosed by the first, second, and third inner rings, respectively.

Can you guess the maximum distance between any two vertices at the same level in a rooted tree? Look at the following example.

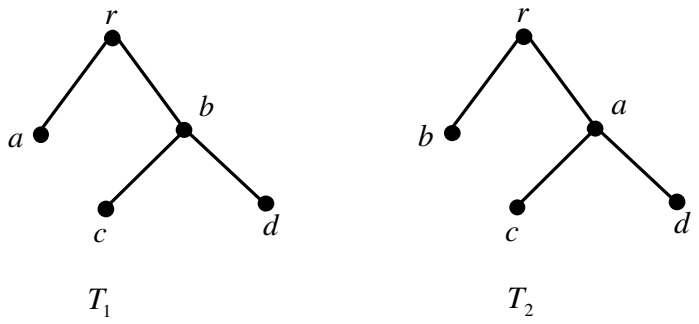
Example 5: Show that if u and v are two vertices in a rooted tree at level ℓ, then $d(u, v) \leq 2\ell$.

Solution: Let (T, r) be a rooted tree. If $u, v \in V(T)$, then we have

$$d(u,v) \le d(u,r) + d(r,v) = \ell + \ell = 2\ell.$$

***

Note that a rooted tree T is said to be **ordered** if the children of every vertex of T have a left -to-right ordering. For example, look at the following trees.



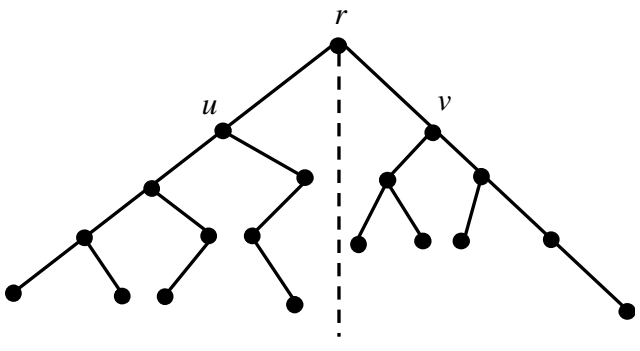
T_1 T_2

In T_1, the children of r occur in the order (a,b), whereas in T_2 the children of r occur in the order (b,a). So T_1 and T_2 as ordered trees are nonisomorphic, although they are isomorphic if treated unordered. Let us now look at a special class of rooted trees.

Binary Trees

Definition: A **binary tree** is a rooted plane tree where each vertex has at most two children, and each child of a vertex is designated as its **left child** or its **right child**.

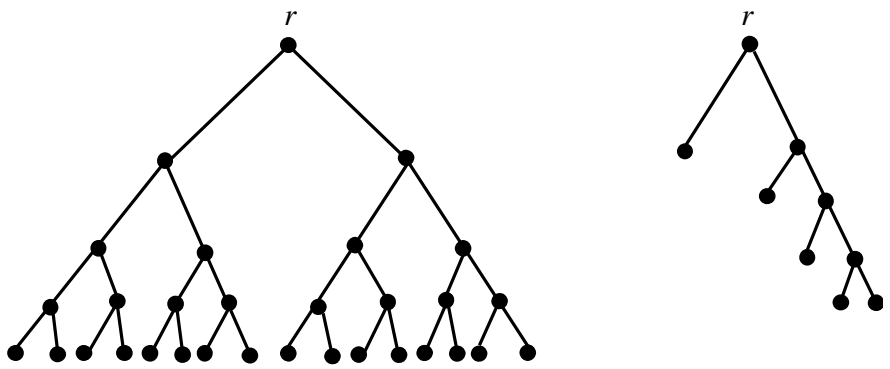
We draw the left child of a binary tree in the left below of the parent, and the right child to the right below. For example in the binary tree (T, r) given below, u is the left child of r, and v the right child of r. The subtree (T_1, u) of T, which consists of u and all the descendents of u is called the **left subtree** of T. Likewise, the subtree (T_2, v) of T consisting of v and all the descendents of v, is called the **right subtree** of T. The two subtrees of T can be separated by a vertical line passing through the root r.



In the binary tree drawn above, not all the parents have two children. For instance, the right child of u has just one child. Binary trees where all parents have two children are given a specific name as defined below.

Definition: A binary tree is called a **complete binary tree** if every vertex has no child or exactly two children.

Two complete binary trees are drawn below.



(a) (b)

Fig. 5: Two complete binary trees

The following examples tell us more about the complete binary trees.

Example 6: Show that the total number of vertices in a complete binary tree is always odd.

Solution: Assume that a complete binary tree (T, r) has n vertices. Then r has degree 2. The remaining $n-1$ vertices have degree 1 or 3, which are odd numbers. We know that the number of odd vertices in a graph is always even. So $n-1$ is even, and hence n is odd.

***

Example 7: Show that a complete binary tree has $\frac{n+1}{2}$ leaves, if it has n vertices.

Solution: Let p be the number of leaves in a complete binary tree. Then, the number of vertices having degree 3 is $n-p-1$. We know that the total number of edges in a tree are $n-1$. So by the Hand-shaking Lemma

$$p + 3(n-p-1) + 2 = 2(n-1),$$

which gives $p = (n+1)/2$.

***

Let us revisit the two binary trees in Fig. 5. The tree in Fig. $5(a)$ is a 4-level complete binary tree, with maximum possible number of vertices at each level. The tree in Fig. 5 (b) too is a 4-level complete binary tree, but with the minimum possible number of vertices at each level. To be sure you can draw any other 4-level complete binary tree with as few vertices as possible. You must come up with an isomorphic copy of this tree.

We are now ready to compute the maximum and minimum possible heights of a complete binary tree. The following theorem gives us this information.

Theorem 5: For every h-level complete binary tree on n vertices,

$$\log_2(n+1) - 1 \leq h \leq \frac{n-1}{2}.$$

Proof: First we prove the inequality $\log_2(n+1)-1 \le h$. Let a_ℓ be the maximum number of vertices in T at level ℓ. Clearly $a_0 = 1$. Each vertex at level ℓ has at most two children. So, $a_{\ell+1}$ must be twice of a_ℓ. Thus, we get the recurrence relation

$$a_{\ell+1} = 2a_\ell, \qquad a_0 = 1.$$

It is easy to see that $a_\ell = 2^\ell$ satisfies this relation. Now we have

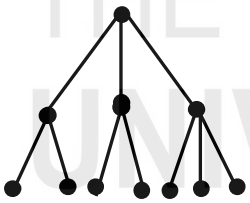
$$\begin{aligned} n \le \sum_{k=0}^{h} a_\ell \Rightarrow n \le \sum_{\ell=0}^{h} 2^\ell &= 2^{h+1} - 1 \\ \Rightarrow n+1 &\le 2^{h+1} \\ \Rightarrow \log_2(n+1) &\le h+1 \\ \Rightarrow \log_2(n+1) - 1 &\le h. \end{aligned}$$

To prove the inequality $h \le \frac{n-1}{2}$ just note that each level has at least 2 vertices, except level 0 which has 1 vertex. So, $n \ge 2h+1$ which implies the inequality. ■

k-ary Trees

Now, we shall discuss k-ary trees.

Definition: Let $k \ge 2$. A k **-ary** tree is a rooted tree in which each vertex has at most k children. A binary tree is the special case where $k = 2$. A 3-ary tree is called a **ternary** tree. For example, a ternary tree of height 2 is drawn below.



Let us now look at the following result.

Theorem 6: An h-level k-ary tree has at most k^h leaves.

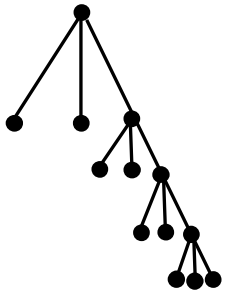
Proof: We shall prove this theorem by mathematical induction on h. If $h = 0$ then tree consists only $n^0 = 1$ leaf.

Let the result be true for $h = k$. We shall show that it is also true for $h = \ell + 1$.

A k-ary tree of height ℓ has k^ℓ leaves. If we add k vertices to each leaf then maximum number of possible leaves is $k^\ell k = k^{\ell+1}$ leaves. Thus, the result is proved. ■

Definition: A k-ary tree is said to be **complete** if each vertex has either no child or has exactly k children.

A complete 4 - level ternary tree is drawn below.



Note that in the above tree each level except the zeroth level has exactly 3 vertices. So it has $3\times 4+1=13$ vertices. More generally, in an n -vertex h -level complete k - ary tree we have $n \geq kh+1$. So $h \leq \frac{n-1}{k}$. It is not difficult to obtain the lower bound for h, if you have gone through the proof of Theorem 5. So, we state the result below without proof.

Theorem 7: For every n -vertex h - level complete k - ary tree

$$\log_k (n+1)-1 \leq h \leq \frac{n-1}{k}$$

You can see that Theorem 7 is a generalisation of Theorem 5. For example, a 99-vertex complete 10-ary tree has height

$\log_{10}(99+1)-1 \leq h \leq \frac{99-1}{10}$, i.e., $1 \leq h \leq 9$.

Now, you can try the following exercises.

E15) How many nonleaves are there in a complete binary tree on 5 vertices? On n vertices?

E16) There exists a rooted tree with more parents than children. True or false?

E17) Show that in a complete binary tree the number of leaves is exactly one more than the number of nonleaves.

E18) Draw a rooted tree on 5 vertices where every vertex, except the root and the leaves, has exactly the same number of ancestors and descendents.

E19) Does there exist a complete binary tree on 10 vertices? Why?

E20) Draw two different binary trees with 5 vertices having maximum number of leaves.

E21) What is the maximum order of a 5- level binary tree? Draw such a tree.

E22) Show that the number of leaves in an n - vertex complete k - ary tree is $(nk-n+1)/k$.

E23) Calculate the number of vertices in a complete binary tree of height 4.

Let us now summarise what we have covered in this unit.

3.5 SUMMARY

In this unit we have covered the following points:

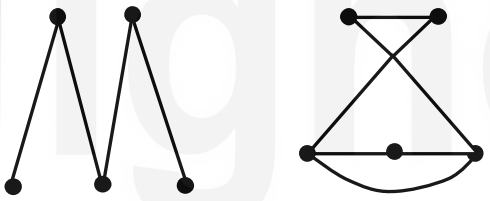
1. Definitions and example of tree and forest, and basic result
2. Characterisations of tree
3. Spanning trees and their enumeration
4. Rooted trees, binary trees and k -ary trees, and some results on them.

3.6 SOLUTIONS / ANSWERS

E1) i) True. A connected forest is the same as a connected acyclic graph, which is a tree.

ii) False. A counterexample is $K_{2,2}$ which contains a cycle.

iii) False. For a counterexample, consider the tree T and its complement $\overline{T}$ given below.



T $\overline{T}$

Clearly $\overline{T}$ contains a cycle, and hence is not a forest.

iv) True. K_2 is one such tree. (In fact, it is the only one.)

v) True. Since T is acyclic, $T-e$ is also acyclic.

E2) Since the removal of any edge from a graph does not remove any vertex from the graph.

$$\begin{aligned} n(T) &= n(T_1)+n(T_2) \\ &= m(T_2)+n(T_2) && (\text{Given } n(T_1)=m(T_2)) \\ &= (n(T_2)-1)+n(T_2) \\ &= 2n(T_2)-1 \end{aligned}$$

Since T is a tree, so $n(T)=m(T)+1=50+1=51.$ Hence $2n(T_2)-1=51,$ which implies $n(T_2)=26$ and $n(T_1)=25.$ Therefore, T_1 has 25 vertices and 24 edges, and T_2 has 26 vertices and 25 edges.

E3) For $n=2, K_2$ is a tree with both vertices as leaves. For $n\geq 3, K_{1,n-1}$ is a tree on n vertices with $n-1$ leaves. No tree on $n\geq 3$ vertices has n leaves (Why?) Hence $n-1$ is the maximum possible number of leaves in a tree on n vertices, You can recall that $K_{1,n-1}$ is called an n-star.

E4) Here $\sum_{i=1}^{5} d_i = 10 \neq 2(5-1),$ Hence by Theorem 3, there is no such tree.

E5) We know that a tree has no cycles, and hence no odd cycles. This implies every tree is a bipartite graph.

E6) Yes. Let x be the unique common vertex of T_1 and T_2. Clearly, there is a unique path from x to every vertex of T_1 as well as to every vertex of T_2. This implies for each $u \in V(T_1)$ and $v \in V(T_2)$, there is a unique (u,v)-path in $T_1 \cup T_2$. Therefore, by Theorem $2, T_1 \cup T_2$ is a tree.

E7) Let $H_1, H_2 \ldots, H_k$ be the components of G. If for every $i \in \{1,2,\ldots,k\}$, H_i is not a tree, then $m(H_i) > n(H_i) - 1$. This means

$$m(H_i) \geq n(H_i) \Rightarrow \sum_{i=1}^{k} m(H_i) \geq \sum_{i=1}^{k} n(H_i)$$
$$\Rightarrow m(G) \geq n(G),$$

which is a contradiction. Hence some H_i is a tree.

E8) Let u and v be two nonadjacent vertices of T. Then T has a unique (u,v)-path, say P. Then $P + uv$ is a unique cycle in $T + uv$.

E9) Let T be a spanning tree of G, such that $e \notin E(T)$. Then $T + e$ has a cycle containing e, which is impossible as e is a cut-edge. Hence $e \in E(T)$. Since T is arbitrary every spanning tree of G contains e.

E10) Let T be a maximal acyclic subgraph of G. If G contains no cycles, then $T = G$, and hence T is a spanning tree of G. Otherwise, we can remove an edge from a cycle of G to get a subgraph G_1. Note that G_1 is connected. If G_1 has any cycle, we can remove an edge from a cycle of G_1 to get a subgraph G_2. We can repeate this process as long as the resulting subgraph G_i, for $i \geq 1$ has a cycle. But G can have at most finitely many cycles. So let k be the least k such that G_k has no cycles. Then $T = G_k$. Since $V(T) = V(G)$. Hence T is a spanning tree of G.

E11) Here

$$Q = \left[\begin{array}{cc|c|ccc} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ \hline 0 & -1 & 3 & -1 & -1 & 0 \\ \hline 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{array}\right]$$

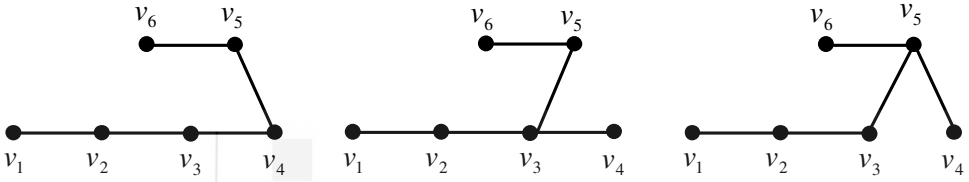
The 3^{rd} row and 3^{rd} column contain the most number of nonzero entries. So, let us remove them from Q to obtain Q_{33}.

$$\therefore \tau(G) = (-1)^{3+3} \, det \, Q_{33}$$
$$= \begin{vmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{vmatrix} \qquad (C_2 \to C_2 + C_1)$$

$$= \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{vmatrix} \qquad (R_2 \to R_1 + R_2)$$

$$= \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{vmatrix} = \begin{vmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 1 \end{vmatrix} = 3$$

The spanning trees are drawn below



E12) We have

$$\tau(K_n) = \begin{vmatrix} n-1 & -1 & -1 & \cdots & -1 \\ -1 & n-1 & -1 & \cdots & -1 \\ -1 & -1 & n-1 & \cdots & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & \cdots & n-1 \end{vmatrix}_{(n-1)\times(n-1)} \qquad R_1 \to \sum_{i=1}^{n-1} R_i$$

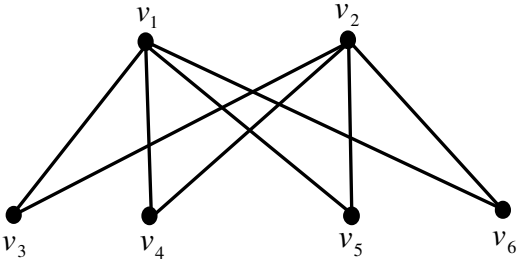
$$= \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ -1 & n-1 & -1 & \cdots & -1 \\ -1 & -1 & n-1 & \cdots & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & \cdots & n-1 \end{vmatrix}_{(n-1)\times(n-1)} \qquad C_i \to C_i - C_1, \forall i \geq 2$$

$$= \begin{vmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & n & 0 & \cdots & 0 \\ -1 & 0 & n & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & \cdots & n \end{vmatrix}_{(n-1)\times(n-1)} = \begin{vmatrix} n & 0 & \cdots & 0 \\ 0 & n & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & n \end{vmatrix}_{(n-2)\times(n-2)} = n^{n-2}$$

E13) i) $\tau(K_3) = 3^{3-2} = 3$

ii) $\tau(K_4) = 4^{4-2} = 16$

iii) To compute $\tau(K_{2,4})$, let us label the vertices of $K_{2,4}$ follows.



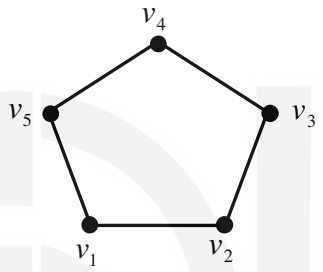
Then

$$Q = \begin{vmatrix} 4 & 0 & -1 & -1 & -1 & -1 \\ 0 & 4 & -1 & -1 & -1 & -1 \\ -1 & -1 & 2 & 0 & 0 & 0 \\ -1 & -1 & 0 & 2 & 0 & 0 \\ -1 & -1 & 0 & 0 & 2 & 0 \\ -1 & -1 & 0 & 0 & 0 & 2 \end{vmatrix}$$

Hence

$$\tau(K_{2,4}) = \begin{vmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 2 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 & 0 \\ -1 & 0 & 0 & 2 & 0 \\ -1 & 0 & 0 & 0 & 2 \end{vmatrix} = 32$$

E14) i) Let the vertices of C_5 be labelled as



Then

$$Q = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}$$

$$\therefore \tau(C_5) = (-1)^{1+1} \det Q_{11} = \begin{vmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{vmatrix} = 5$$

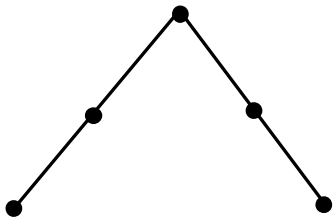
ii) When the vertices are unlabelled we cannot apply the Matrix Tree Theorem. We can observe that any spanning tree of C_5 in this case is an isomorphic copy of P_5 .So there is only 1 spanning tree, which is P_5 .

E15) We know that a complete binary tree on n vertices has $\frac{n+1}{2}$ leaves, and hence $n - \frac{n+1}{2} = \frac{n-1}{2}$ nonleaves. Thus, a complete binary tree on 5 vertices has $\frac{5-1}{2} = 2$ nonleaves.

E16) False. Every parent has at least one child. Thus the number of children in a tree are never less than the number of parents.

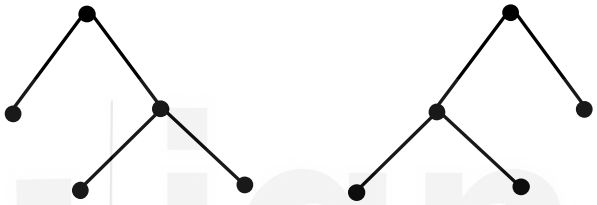
E17) From E15, we can see the difference between the number of leaves and the number of nonleaves is $\frac{n+1}{2}-\frac{n-1}{2}=1.$

E18) One such tree is given below.



E19) No. See Example 6.

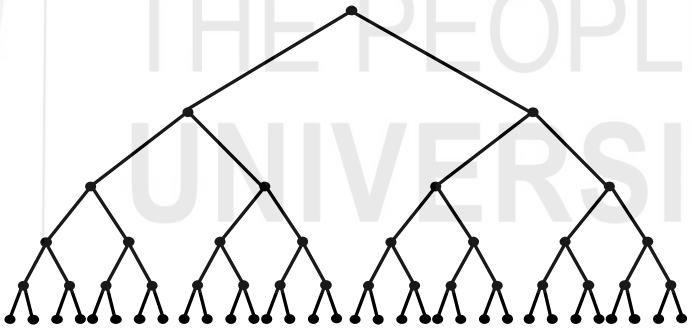
E20) The tree with the required properties are given below.



E21) A 5- level binary tree has the maximum order when it is complete. From the proof of Theorem 5, we have

$$n \leq 2^{h+1}-1=2^{5+1}-1=63.$$

Such a tree is drawn below.



E22) Let p be the number of leaves. We know that the root has degree k. So, the remaining $n-p-1$ vertices have degree $k+1$. Therefore, by the Handshaking Lemma

$$p+k+(n-p-1)\ (k+1)=2n-2$$

Solving this for p, we get $p=\frac{nk-n+1}{k}$, as desired.

E23) From the proof Theorem 5, we know that the number of vertices at level ℓ of a complete binary tree is 2^{ℓ}. Therefore, the total number of vertices in a complete binary tree of height 4 is

$$2^0+2^1+2^2+2^3+2^4=31.$$

UNIT 4

OPTIMISATION IN TREES

Structure **Page Nos.**

4.1 INTRODUCTION

In the last unit, you have learnt about trees. You have seen some properties of trees such as trees are minimally connected, in the sense that there is exactly one path between any two vertices of a tree, trees are bipartite graphs and so on. In this unit, we shall discuss a number of algorithms related to spanning trees. In Section 4.2, we begin with two widely known algorithms namely, BFS and DFS for graph traversal. Both these algorithms construct spanning trees by employing two different techniques. In Section 4.3, we shall discuss spanning trees in weighted graphs, and shall describe what we mean by a minimum weight spaning tree. In this connection, we shall discuss two algorithms namely Kruskal's Algorithm and Prim's Algorithm to find minimum weight spanning trees. Section 4.4 introduces the Dijkstra's Algorithm which is used to find shortest paths from a given vertex of a weighted graph to all the other vertices.

Objectives

After studying this unit, you should be able to

- describe, and apply BFS and DFS algorithms on a graph;
- find minimum weight spanning tree of a weighted graph using Kruskal's Algorithm and Prim's Algorithm;
- find shortest paths using Dijkstra's Algorithm in a weighted graph.

4.2 CONSTRUCTING SPANNING TREES

Here we shall discuss in detail two algorithms for finding spanning trees of a given connected graph. These algorithms are breadth first search (BFS) and depth first search (DFS).

4.2.1 Breadth-First-Search (BFS) Algorithm

The BFS Algorithm visits the vertices level by level from a given root of the graph. The search begins at the root. After visiting the root all the neighbours of the root are visited. Then the neighbours of the neighbours of the root are visited. The process continues as long as there remain unvisited vertices in the graph. The procedure is listed below. It uses two containers T and Q. Here T contains the edges of the spanning tree. The container Q is a 'queue' data structure. You can recall from the course Programming and Data Structures (MMT-001) that a queue data structure works on the principle of first-in-first-out (FIFO). It means elements are added at the back of Q, and removed from the front of Q. The spanning tree obtained by this process is called a **BFS tree**. Now let us look at the procedure.

Procedure (BFS Algorithm):

Input: A connected graph G, with a root vertex r.

Output: A rooted spanning tree (T, r).

i) Initialise T and Q by empty sets, i.e., $T = \emptyset$, $Q = \emptyset$, and mark all the vertices as unvisited.

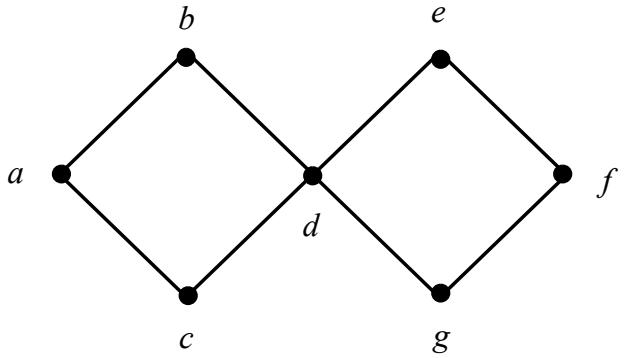
ii) Append r into Q, and mark r as visited.

iii) If Q is not empty, remove a vertex u from Q. Otherwise, stop.

iv) For each unvisited neighbour v of u add the edge uv to T, and mark v as visited, and append v into Q. Go to step iii).

Let us illustrate BFS Algorithm with the help of an example.

Example 1: Construct a rooted spanning tree of the following graph with root a.



Solution: First let us mark all the vertices as unvisited. Let $T = \emptyset$, $Q = \emptyset$. We begin by appending a into Q, and marking a as visited. Now Q is not empty. So a is removed from Q. The neighbours of a are b and c, which are both unvisited. We select b and add the edge ab to T, append b to Q, and mark

b as visited. Then we select c, add the edge ac to T, append c to Q, and mark c as visited. Thus we have

$$Q = \boxed{b}\boxed{c}\boxed{\ }, \; T = \{ab, ac\}$$

Here we have selected b arbitrarily. We could have selected c as well, instead of b. But then, the resulting spanning tree could be different. Just try!

Next we remove b from Q. The unvisited neighbour of b is d. So bd is added to T, d is appended to Q, and d is marked as visited. Now we have

$$Q = \boxed{c}\boxed{d}\boxed{\ }, \; T = \{ab, ac, bd\}.$$

Next we remove c from Q. Since all the neighbours of c are already visited we leave T unchanged. So we have

$$Q = \boxed{d}\boxed{\ }, \; T = \{ab, ac, bd\}.$$

Now we remove d from Q. The unvisited neighbours of d are e and g. We select e. The edge de is added to T, e is appended to Q, and e is marked as visited. Similarly the edge dg is added to Q, g is marked as visited, and g is appended to Q. Accordingly Q and T are updated as follows:

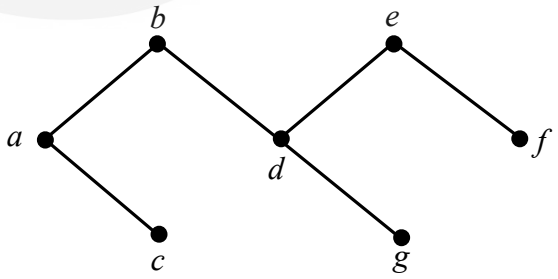
$$Q = \boxed{e}\boxed{g}\boxed{\ }, \; T = \{ab, ac, bd, de, dg\}.$$

Next we remove e from Q. The only unvisited neighbour of e is f. So the edge ef is added to T, f is appended to Q, and f is marked as visited. Now we have

$$Q = \boxed{g}\boxed{f}\boxed{\ }, \; T = \{ab, ac, bd, de, dg, ef\}.$$

Although at this point all the vertices are visited, Q is still nonempty. So g is removed from Q. Since all the neighbours of g are already visited no change take place in T. Finally f is removed from Q. All the neighbours of f are already visited, so no change takes place in T.

Thus finally we obtain $T = \{ab, ac, bd, de, dg, ef\}$ which are the edges of the spanning tree. The corresponding tree is given below.



***

Now let us come to the DFS algorithm.

4.2.2 Depth First Search (DFS) Algorithm

Unlike BFS, the DFS Algorithm visits only one vertex at a time, and finds a nonextendable path from a given vertex by visiting only unvisited vertices. From the last vertex on this path, the algorithm goes back to the previous vertex, and from there again finds a nonextendable path by visiting only unvisited vertices. This process continues till the algorithm reaches back to the

root after visiting all the vertices in the graph. This technique is often called 'backtracking'. The spanning tree obtained by this process is called a **DFS tree.** The detailed procedure is given below.

Procedure (DFS Algorithm):

Input: A connected graph G with a root vertex r.

Output: A spanning tree (T, r)

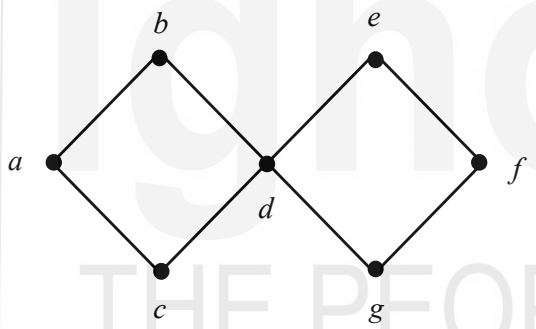
i) Initialise T as empty set, i.e., $T = \emptyset$, and mark all the vertices as unvisited.

ii) Call DFS (r), and stop.

iii) DFS (u): Mark u as visited. For each unvisited neighbour v of u, add uv to T, and call DFS (v).

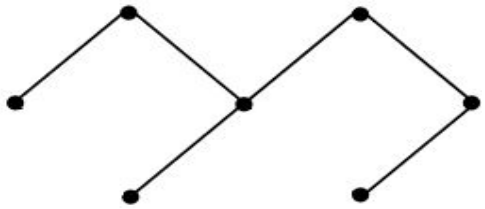
Let us illustrate the DFS Algorithm with an example.

Example 2: Find a spanning tree using DFS Algorithm of the following graph starting with a.



Solution: Mark all the vertices as unvisited, and take $T = \emptyset$. The vertex a is the root of the tree, so we call DFS (a). Now a is marked as visited. We have b and c as the unvisited neighbours of a, we select b and add the edge ab to T, and call DFS (b). Now b is marked as visited. Since d is an unvisited neighbour of b, we add the edge bd to T, and call DFS (d). Now d is marked as visited. The unvisited neighbours of d are c, e and g. Let us select c for the next visit. Hence the edge cd is added to T, and call DFS (c) is made. Now c is marked as visited. There are no unvisited neighbours of c, and we backtrack to d. Then we select the unvisited neighbour e of d for the next visit. We add the edge de to T, all call DFS (e). Now the unvisited neighbour of e is f, and hence we add the edge ef to T, and call DFS (f). Now f is marked as visited. The only unvisited neighbour of f is g. So the edge fg is added to T, and call DFS (g) is made. Now g is marked as visited. There are no unvisited neighbours of g, so we backtrack to f. Since there are no unvisited neighbours of f, we backtrack to e. Since there are no unvisited neighbours of e, we backtrack to d. Since there are no unvisited neighbours of d, we backtrack to b, and finally to a.

Thus the edges of the DFS tree so obtained are $T = \{ab, bd, cd, de, ef, fg\}$. The corresponding tree is drawn below.

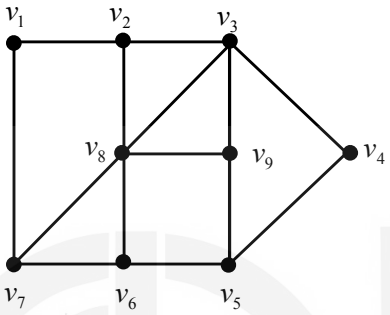


***

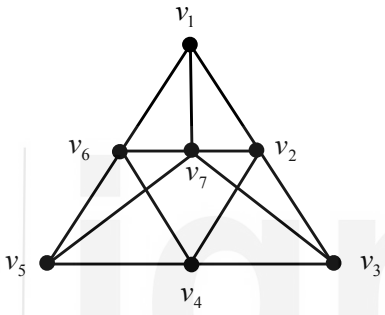
Now, try the following exercises.

E1) Produce a spanning tree for the following graphs, using

i) BFS Algorithm, ii) DFS Algorithm.



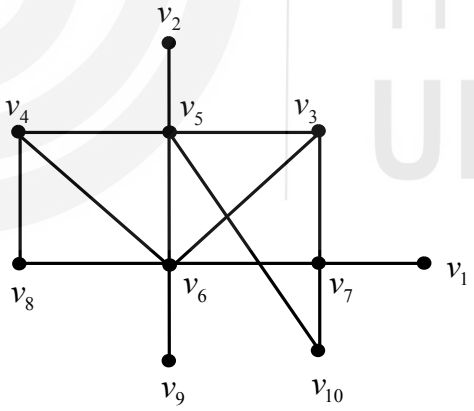
G_1



G_2

E2) Taking v_1 as the root, find a spanning tree of the following graph by BFS and DFS algorithms

i) BFS Algorithm, ii) DFS Algorithm.



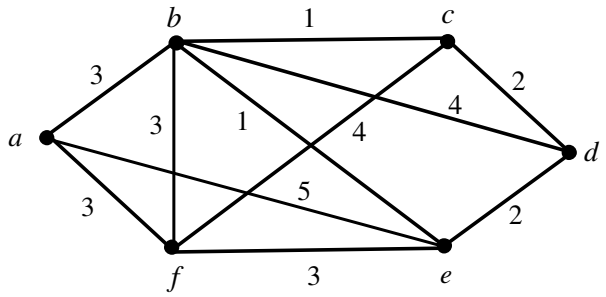
In the next section, we shall discuss about algorithms that find a minimum weight spanning tree in a weighted graph.

4.3 MINIMUM WEIGHT SPANNING TREES

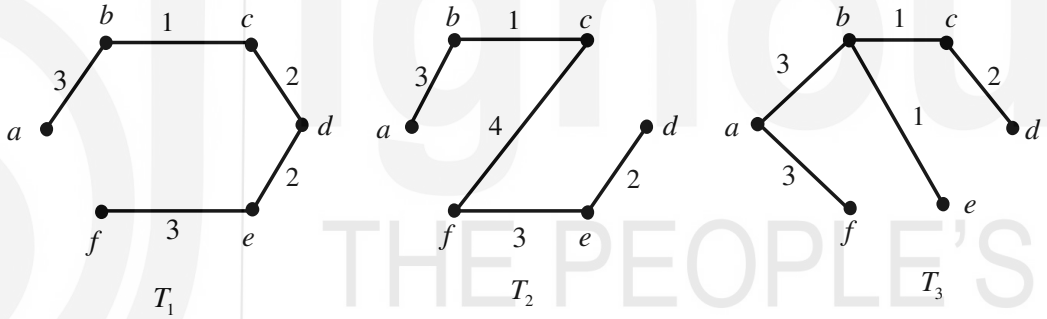
Recall that a weighted graph is a pair (G,W), where G is a graph, and $W : E(G) \to \mathbb{R}$ is a weight function. In this section, and in the next section as well, we shall assume the weight function to be nonnegative. That is $W(e) \geq 0$ for every edge e of G. Here we will discuss spanning tree for a weighted

for every edge e of G. Here we will discuss spanning tree for a weighted graph and two algorithms due to Kruskal and Prim to find minimum weight spanning tree in weighted graphs.

Let G be a connected weighted graph, and T a spanning tree of G. Then the **weight** of T is defined as the sum of the weights of all the edges in T. Note that different spanning trees in G may have different weights.
For example, consider the following graph G.



Three different spanning trees T_1, T_2 and T_3 of G are shown below, having the weights $11, 13$ and 10, respectively.



Here T_3 has the **smallest weight**, among "all" the spanning trees of G. (Just verify, using the Matrix Tree Theorem.)

Note that a minimum weight spanning tree need not be unique.

Definition: Let G be a weighted graph. A spanning tree of G which has the minimum weight is called a **minimum weight** spanning tree of G.

There are a number of methods available to find a minimum weight spanning tree in a given weighted graph. The two important algorithms to find out the minimal spanning tree in a given graph G are **Kruskal's Algorithm** and **Prim's Algorithm.**

4.3.1 Kruskal's Algorithm

This algorithm sequentially adds an edge of minimum weight to an existing collection of edges such that at no stage, a cycle is produced. The algorithm stops as soon as the collection has $n-1$ edges, where n is the order of the graph. The procedure is given below.

Procedure (Kruskal's Algorithm)

Input: A connected weighted graph (G,W)

Output: A minimum weight spanning tree of G.

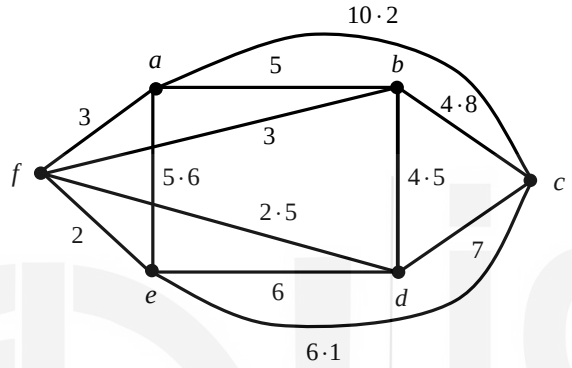
i) Sort the edges of G in the nondecreasing order of their weights. Initialise $T = \emptyset$.

ii) If $|T| = n(G) - 1$, stop. Otherwise, let e be an edge in G not in T such that $W(e)$ is minimum and addition of e to T does not create a cycle. If there are more than one such edge, select any one of them randomly.

iii) $T = T \cup \{e\}$, and go to Step ii).

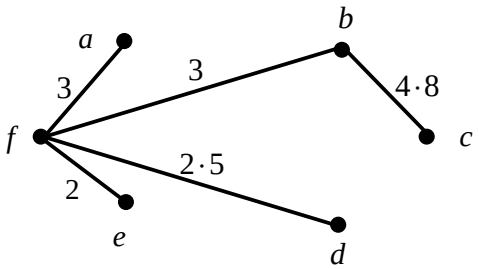
Let us consider an example.

Example 3: Find a minimum weight spanning tree of the following graph G using Kruskal's Algorithm.



Solution: The graph has 6 vertices. Note that $T = \emptyset$ in the beginning. The smallest weight edge in the graph is ef with $W(ef) = 2$. So, $T = \{ef\}$. Since $|T| \neq 5$, we select the smallest weight edge in G not in T, which does not create a cycle. It is fd with $W(fd) = 2 \cdot 5$. Hence, $T = \{ef, fd\}$.

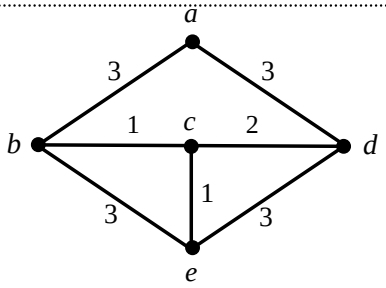
Again we have $|T| \neq 5$. So we select the edge with smallest weight not in T, which does not create a cycle. We get $W(af) = W(bf) = 3$. Let us select af, and so $T = \{ef, df, af\}$. Again we have $|T| \neq 5$. So, the next edge with smallest weight to be selected is bf, and now $T = \{ef, df, af, bf\}$. Finally, we select the edge bc to get $T = \{ef, df, af, bf, bc\}$. The resulting tree is shown below.



The weight of the tree is
$W(T) = W(ef) + W(df) + W(af) + W(bf) + W(bc) = 15.3.$

***

Example 4: Use Kruskal's Algorithm to find a minimum weight spanning tree in the following graph



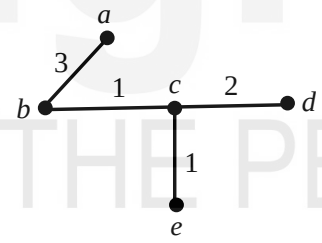
Solution: For convenience, we write the edges with their weights in increasing order, as shown in the following table

edge	bc	ce	cd	ab	ed	ad	be
weight	1	1	2	3	3	3	3

We begin with $T = \emptyset$. There are two edges namely, bc and ce that have smallest weight. Let us select bc, so that $T = \{bc\}$. Next, select ce to get $T = \{bc, ce\}$. Then the smallest weight edge is cd which does not create a cycle. So, $T = \{bc, ce, cd\}$. Next, we have the edge ab with smallest weight equal to 3, and which does not create a cycle. So, $T = \{bc, ce, cd, ab\}$. Since now $|T| = 4$, we stop. The weight of the tree is

$$W(T) = W(bc) + W(ce) + W(cd) + W(ab) = 1 + 1 + 2 + 3 = 7.$$

The resulting spanning tree is shown below.



***

As you have seen, in Kruskal's Algorithm a spanning tree is obtained at the final step. At intermediary steps, what it maintains is a forest. In contrast to this, the next algorithm we are going to discuss begins with a tree, and enlarges it to get a minimum weight spanning tree.

4.3.2 Prim's Algorithm

The procedure of Prim's algorithm is as follows.

Procedure (Prim's Algorithm):

Input: A connected weighted graph G

Output: A minimum weight spanning tree of G.

i) Let $T = (V_T, E_T)$, where $V_T = \emptyset$, $E_T = \emptyset$. Choose a vertex u arbitrarily. Then $V_T = \{u\}$.

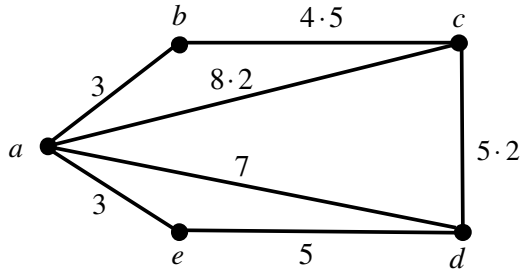
ii) If $|E_T| = n(G) - 1$, stop. Otherwise, let xy be an edge with smallest weight such that $x \in V_T$ and $y \notin V_T$. If there are more than one such edge, select any one arbitrarily.

iii) $E_T = E_T \cup \{xy\}$ and $V_T = V_T \cup \{y\}$. Go to step ii).

***

Let us illustrate this method of finding a minimum weight spanning tree through an example.

Example 5: Find a minimum weight spanning tree of the following weighted graph, using Prim's Algorithm.



Solution: The weights of the edges of the graph G are tabulated below

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} \infty & 3 & 8.2 & 7 & 3 \\ 3 & \infty & 4.5 & \infty & \infty \\ 8.2 & 4.5 & \infty & 5.2 & \infty \\ 7 & \infty & 5.2 & \infty & 5 \\ 3 & \infty & \infty & 5 & \infty \end{bmatrix} \end{array}$$

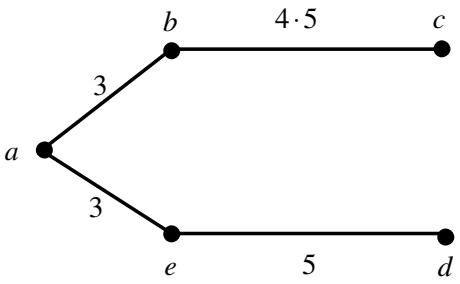
Here ∞ in an entry indicates that the corresponding edge does not exist.

Let us begin with the vertex a. So $V_T = \{a\}$ and $E_T = \phi$. The edge ab and ae have the smallest weight. Let us select ab. Then $V_T = \{a,b\}, E_T = \{ab\}$.

Now, we have to select an edge with the smallest weight having exactly one endpoint in V_T. Such an edge is ae. So we have $V_T = \{a,b,c\}$ and $E_T = \{ab, ae\}$. Then we select the edge bc as it has the smallest weight among all the edges having exactly one endpoint in V_T.

So $V_T = \{a,b,c,e\}$ and $E_T = \{ab, ae, bc\}$. Next, we have to select an edge with smallest weight with exactly one endpoint in V_T. Such an edge is de. So, we get $V_T = \{a,b,c,d,e\}$ and $E_T = \{ab, ae, bc, de\}$. Now $| E_T | = 4$, so the algorithm stops.

The spanning tree so obtained is

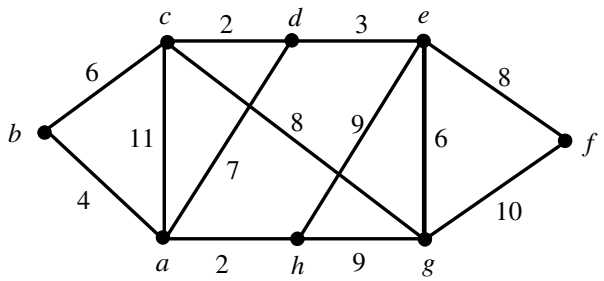


We have $W(T) = 15 \cdot 5$.

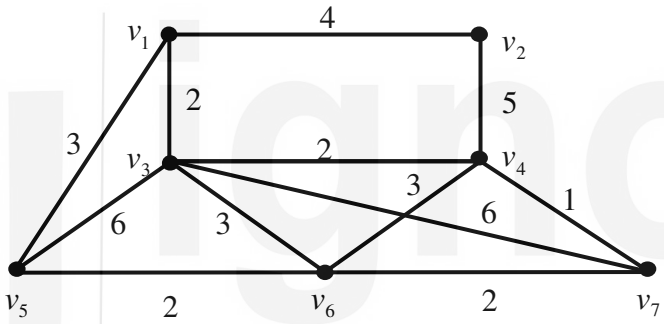
***

Now, try the following exercises.

E3) Use Kruskal's Algorithm to find a minimum weight spanning tree in the following graph, taking a as the root.



E4) Use Prim's Algorithm to find a minimum weight spanning tree of the following graph, starting with v_1.



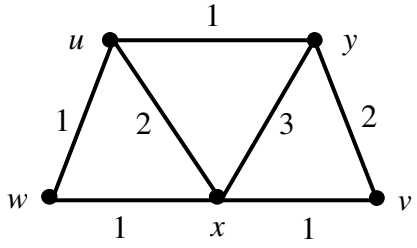
Thus far we have seen algorithms that find spanning trees in weighted or unweighted graphs.

In the next section, we shall discuss an algorithm that finds a shortest path from a given vertex to every vertex in a weighted graph.

4.4 DIJKSTRA'S ALGORITHM

Dijkstra's Algorithm is one of the most efficient method for finding a shortest path, and hence the distance, from a root vertex to any other vertex in the given weighted graph. Of course, the graph must be connected. Before we discuss the algorithm, let us understand what we mean by the distance between two vertices of a weighted graph (G, W). Here we assume that $W(e) \geq 0$ for every edge e of G. Recall that the weight of a path in G is the sum of the weights on the edges of the path. A **shortest** path is one whose weight is the smallest. Then, the **distance** between any two vertices u and v of a weighted graph G is the weight of a shortest (u, v)-path in G. For example, in the weighted graph G given below, the shortest (u, v)-paths are:

$P_1 = (u, w, x, v),\ P_2 = (u, x, v),\ P_3 = (u, y, v).$



We have $W(P_1) = W(P_2) = W(P_3) = 3$. All other (u,v)-paths have length more than 3. Thus the distance between u and v is 3.

Now we describe the procedure of the Dijkstra's Algorithm.

Procedure (Dijkstra's Algorithm):

Input: A connected weighted graph (G,W), and a root vertex u of G.

Output: Distances from u to all the vertices of G.

Here, for each $x \in V(G), d(x)$ indicates the temporary distance of x from u.

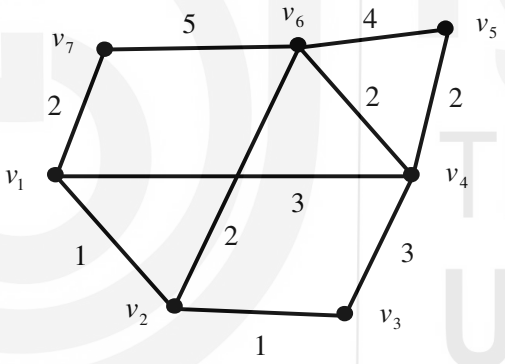
i) Set $d(u) = 0$, and $S = \{u\}$. For each vertex $x \notin S$, set $d(x) = W(ux)$.

ii) If $S = V(G)$, stop. Otherwise, let $v \notin S$ be a vertex such that $d(v) = \min_{x \notin S} d(x)$. Add v to S.

iii) For each neighbour x of v that does not belong to S, set $d(x) = \min\{d(x), d(v) + W(vx)\}$. Go to Step ii).

Let us consider an example, to see how Dijkstra's Algorithm works.

Example 6: Using Dijkstra's Algorithm find the distances of all the vertices from v_1, in the weighted graph given below.



Solution: In the beginning we have $S = \{v_1\}$. The current distances of all the vertices from v_1 are shown in the below given distance matrix.

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$d(x)$	0	1	∞	3	∞	∞	2

Here ∞ in a column indicates that there is no edge between the corresponding vertex and v_1. Since $S \neq V(G)$, we select the vertex $v_2 \notin S$ as $d(v_2) = 1 = \min_{x \notin S} d(x)$. So, S becomes $S = \{v_1, v_2\}$. The distance matrix updates to

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$d(x)$	0	1	2	3	∞	3	2

Note that $S \neq V(G)$. Now the vertices $v_3, v_7 \notin S$, are such that $d(v_3) = d(v_7) = 2 = \min_{x \notin S} d(x)$. Adding v_3 to S, we get $S = \{v_1, v_2, v_3\}$. The distance matrix then updates to

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$d(x)$	0	1	2	3	∞	3	2

Again $S \neq V(G)$. So, we select the vertex $v_7 \notin S$, as $d(v_7) = 2 = \min_{x \notin S} d(x)$. Then we have $S = \{v_1, v_2, v_3, v_7\}$, and the distance matrix as

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$d(x)$	0	1	2	3	∞	3	2

Still, we have $S \neq V(G)$. Now $v_4, v_6 \notin S$ are such that $d(v_4) = d(v_6) = 3 = \min_{x \notin S} d(x)$. Let us select v_4. Then we get $S = \{v_1, v_2, v_3, v_4, v_7\}$, and the distance matrix as

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$d(x)$	0	1	2	3	5	3	2

Still $S \neq V(G)$. Now the vertex $v_6 \notin S$ is such that $d(v_6) = 3 = \min_{x \notin S} d(x)$. Then we get $S = \{v_1, v_2, v_3, v_4, v_6, v_7\}$ and the distance matrix as

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$d(x)$	0	1	2	3	5	3	2

Finally, since $S \neq V(G)$, we select v_5 as $d(v_5) = 5 = \min_{x \notin S} d(x)$. Then $S = V(G)$, and the distance becomes as given below.

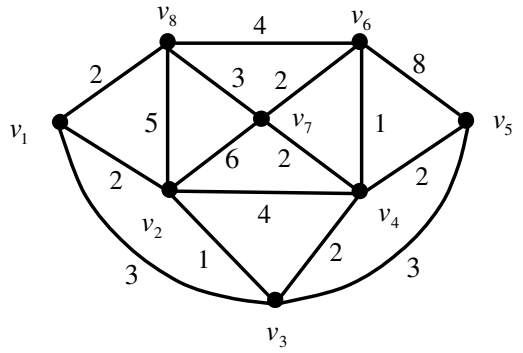
x	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$d(x)$	0	1	2	3	5	3	2

Thus we have $d(v_1, v_2) = 1, d(v_1, v_3) = 2, d(v_1, v_4) = 3, d(v_1, v_5) = 5, d(v_1, v_6) = 3$ and $d(v_1, v_7) = 2$.

***

Now try the following exercise.

E5) Find the distance of the vertex v_1 from each of the vertices of the following weighted graph, using Dijkstra's Algorithm.



4.6 SUMMARY

In this unit, we have covered the following points:

1. BFS algorithm, to find a spanning tree in a connected graph.
2. DFS algorithm, to find a spanning tree in a connected graph.
3. Kruskal's algorithm, to find a minimum weight spanning tree in a connected weighted graph.
4. Prim's algorithm, to find a minimum weight spanning tree in a connected weighted graph.
5. Dijkstra's algorithm to find the distances for all the vertices of a weighted graph from a given vertex.

4.6 SOLUTIONS / ANSWERS

E1) i) Let us first apply BFS Algorithm on G_1. Let v_1 be the root. At the beginning $T = \emptyset$, $Q = \emptyset$ and all the vertices are unvisited. Then v_1 is appended to Q, and marked as visited.

Now $Q \neq \emptyset$ So v_1 is removed from Q. The unvisited neighbours of v_1 are v_2 and v_7. Let us first visit v_2, and then v_7. Accordingly Q and T are

$$Q = \boxed{v_2 \mid v_7 \mid \ }, \quad T = \{v_1v_2,\ v_1v_7\}$$

Now we remove v_2 from Q. The unvisited neighbours of v_2 are v_3 and v_8, visiting them in the order we get

$$Q = \boxed{v_7 \mid v_3 \mid v_8 \mid \ }, \quad T = \{v_1v_2,\ v_1v_7,\ v_2v_3,\ v_2v_8\}$$

Next, we remove v_7 from Q. Now only v_6 remains as the unvisited neighbour of v_7. Accordingly, we get

$$Q = \boxed{v_3 \mid v_8 \mid v_6 \mid \ }, \quad T = \{v_1v_2,\ v_1v_7,\ v_2v_3,\ v_2v_8,\ v_6v_7\}$$

Next we remove v_3 from Q. The unvisited neighbours of v_3 are v_4 and v_9, visiting them in the order we get

$$Q = \boxed{v_8 \mid v_6 \mid v_4 \mid v_9 \mid \ }, \text{ and}$$

$$T = \{v_1v_2, v_1v_7, v_2v_3, v_2v_8, v_6v_7, v_3v_4, v_3v_9\}.$$

Next, we remove v_8 from Q. All the neighbours of v_8 are already visited, so no change takes place in T. Then we remove v_6. The only unvisited neighbour of v_6 is v_5. So we get

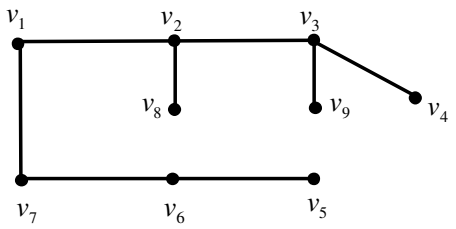
$$Q = \boxed{v_4 \mid v_9 \mid v_5 \mid \ }, \text{ and}$$

$$T = \{v_1v_2, v_1v_7, v_2v_3, v_2v_8, v_6v_7, v_3v_4, v_3v_9, v_5v_6\}.$$

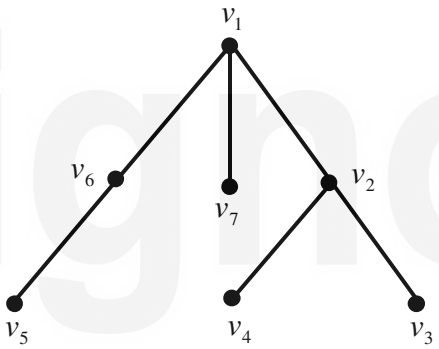
Next, we remove v_4 from Q. All the neighbours of v_4 are already visited. So no change takes place in T. Similarly, v_9 and v_5 are removed from Q without affecting T. Finally $Q = \phi$. The algorithm stops, and we have

$$T = \{v_1v_2, v_1v_7, v_2v_3, v_2v_8, v_6v_7, v_3v_4, v_3v_9, v_5v_6\}.$$

The resulting spanning tree is drawn below.



Similarly, when we apply the BFS algorithm on G_2 we get the spanning tree as shown below.

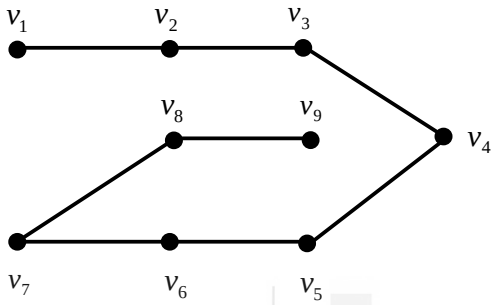


ii) Let us apply DFS on G_1. Let v_1 be the root. First we shall mark all the vertices as unvisited, and take $T = \phi$. We call DFS(v_1). Now v_1 is marked as visited. Then v_2 and v_7 are the unvisited neighbours of v_1. Let us select v_2, add the edge v_1v_2 to T, and call DFS(v_2). Now v_2 is marked as visited. The unvisited neighbours of v_2 are v_3 and v_8. Let us select v_3, add the edge v_2v_3 to T, and call DFS(v_3). Now v_3 is marked as visited. The unvisited neighbours of v_3 are v_4, v_8 and v_9. Let us select v_4, add the edge v_3v_4 to T, and call DFS(v_4). Now v_4 is marked as visited. The vertex v_5 is the only unvisited neighbour of v_4. So the edge v_4v_5 is added to T, and call DFS(v_5) is made. Then v_5 is marked as visited. The unvisited neighbours of v_5 are v_6 and v_9. Let us select v_6, add the edge v_5v_6 to T, and call DFS(v_6). Then v_6 is marked as visited. The unvisited neighbours of v_6 are v_7 and v_8. Let us select v_7, add the edge v_6v_7 to T, and call DFS(v_7). Then v_7 is marked as visited. The unvisited neighbour of v_7 is v_8 only. Then the edge v_7v_8 is added to T, and call DFS(v_8) is made. Then v_8 is marked as visited. The unvisited neighbour of v_8 is v_9. So the edge v_8v_9 is added to T, and call DFS(v_9) is

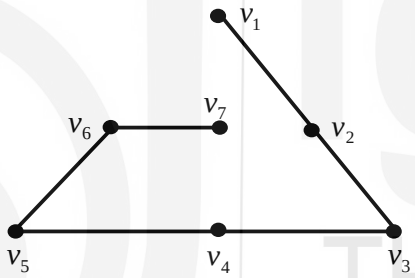
made. Since v_9 has no unvisited neighbours, we backtrack to the parent of v_9, i.e., to v_8. Since all the neighbours of v_8 are already visited, we backtrack to the parent of v_8, i.e., to v_7. Continuing this way we reach to the root v_1. The algorithm stops, and the edges of the spanning tree are

$$T = \{v_1v_2, v_2v_3, v_3v_4, v_4v_5, v_5v_6, v_6v_7, v_7v_8, v_8v_9\}$$

The resulting tree is drawn below.



Similarly apply DFS on G_2 to get the following spanning tree.



E2) i) Let $T = \emptyset$, $Q = \emptyset$. Initially all the vertices are unvisited. Since v_1 is the root, v_1 is appended to Q. Now $Q \neq \emptyset$. So remove v_1 from Q. Now v_7 is the only neighbour of v_1, which is unvisited. So the edge v_1v_7 is added to T, v_7 is marked as visited, and v_7 is appended to Q. Thus we have

$$Q = \boxed{v_7}\boxed{\ }, \quad T = \{v_1v_7\}$$

Now $Q \neq \emptyset$ so we remove v_7 from Q. The unvisited neighbours of v_7 are v_3, v_6 and v_{10}. So let us add the edges v_3v_7, v_6v_7 and v_7v_{10} to T, mark v_3, v_6 and v_{10} as visited, and append v_3, v_6 and v_{10} into Q, in the order. We get

$$Q = \boxed{v_3}\boxed{v_6}\boxed{v_{10}}\boxed{\ }, \quad T = \{v_1v_7,\ v_3v_7,\ v_6v_7,\ v_7v_{10}\}$$

Now we remove v_3 from Q, as $Q \neq \emptyset$. The unvisited neighbour of v_3 is v_5 only. So the edge v_3v_5 is added to T, v_5 is marked as visited, and then v_5 is appended into Q. Thus Q and T become

$$Q = \boxed{v_6}\boxed{v_{10}}\boxed{v_5}\boxed{\ }, \quad T = \{v_1v_7,\ v_3v_7,\ v_6v_7,\ v_7v_{10},\ v_3v_5\}$$

Now we remove v_6 from Q. The unvisited neighbours of v_6 are v_4, v_8 and v_9, visiting them in order we get

$Q =$ | v_{10} | v_5 | v_4 | v_8 | v_9 | |, and

$T = \{v_1v_7, v_3v_7, v_6v_7, v_7v_{10}, v_3v_5, v_4v_6, v_6v_8, v_6v_9\}$

Next, we remove v_{10} from Q. All the neighbours of v_{10} are already visited, so no change takes place in T. Next, we remove v_5 from Q. The only unvisited neighbour of v_5 is v_2. Visiting v_2 we get

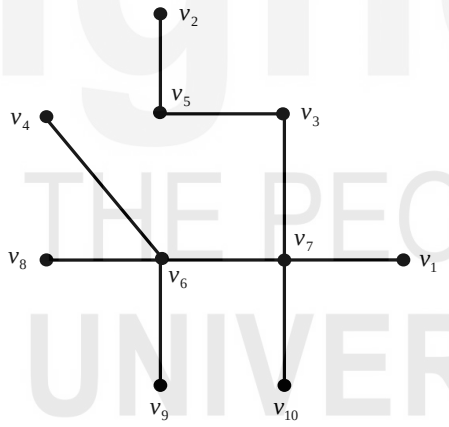
$Q =$ | v_4 | v_8 | v_9 | v_2 | |, and

$T = \{v_1v_7, v_3v_7, v_6v_7, v_7v_{10}, v_3v_5, v_4v_6, v_6v_8, v_6v_9, v_2v_5\}$

Although now all the vertices are visited, Q still remains nonempty. We remove the vertices v_4, v_8, v_9 and v_2 from Q in order to reach $Q = \emptyset$. Now the algorithm stops. Thus the edges of the spanning tree are

$T = \{v_1v_7, v_3v_7, v_6v_7, v_7v_{10}, v_3v_5, v_4v_6, v_6v_8, v_6v_9, v_2v_5\}$

The resulting spanning tree is drawn below.



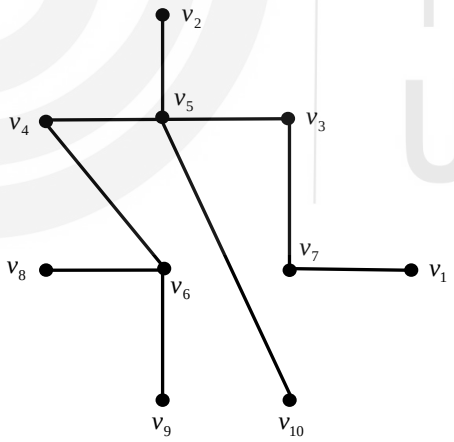
ii) Mark all the vertices as visited. At the beginning we have $T = \emptyset$. Starting with the root v_1, we call DFS(v_1). Then v_1 is marked as visited. The unvisited neighbour of v_1 is v_7, so we add the edge v_1v_7 to T and call DFS(v_7). Then v_7 is marked as visited. The unvisited neighbours of v_7 are v_3, v_6 and v_{10}. Let us select v_3. So the edge v_3v_7 is added to T, and call DFS(v_3) is made. Then v_3 is marked as visited. The unvisited neighbours of v_3 are v_5 and v_6. Let us select v_5, add the edge v_3v_5 to T, and call DFS(v_5). Then v_5 is marked as visited. The unvisited neighbour of v_5 are v_2, v_4 and v_6. Let us select v_2, add the edge v_2v_5 to T, and call DFS(v_2). Then v_2 is marked as visited. Since v_2 has no unvisited neighbours, we backtrack to its parent v_5. Now v_5 still has unvisited neighbours v_4 and v_6. Let us select v_4, add the edge v_4v_5 to T, and call DFS(v_4). So, v_4 is marked as visited. The

unvisited neighbours of v_4 are v_6 and v_8. Let us select v_6, add the edge v_4v_6 to T, and call DFS(v_6). Then v_6 is marked as visited. The unvisited neighbours of v_6 are v_8 and v_9. Let us select v_8, add the edge v_6v_8 to T, and call DFS(v_8). Then v_8 is marked as visited.

Now there are no unvisited neighbours of v_8, so we backtrack to its parent v_6. Since v_6 still has the unvisited neighbour v_9, we add the edge v_6v_9 to T, and call DFS(v_9). Now v_9 is marked as visited. There are no unvisited neighbours of v_9, so we backtrack to its parent v_6. Since all the neighbours of v_6 have already been visited, we backtrack to its parent v_4. Since all the neighbours of v_4 have also been already visited, we backtrack to its parent v_5. Note that v_5 still has the neighbour v_{10} unvisited. So, we add the edge v_5v_{10} to T, and call DFS(v_{10}). Now v_{10} is marked as visited. There are no unvisited neighbours of v_{10}. So we backtrack to its parent v_5. Now v_5 has no unvisited neighbours, so we backtrack to its parent v_3. Since v_3 has no unvisited neighbours, we backtrack to its parent v_7 has no unvisited neighbours we backtrack to v_1. The algorithm stops. The edges of the spanning tree are

$$T = \{v_1v_7, v_3v_7, v_3v_5, v_2v_5, v_4v_5, v_4v_6, v_6v_8, v_6v_9, v_5v_{10}\}$$

The resulting spanning tree is drawn below.

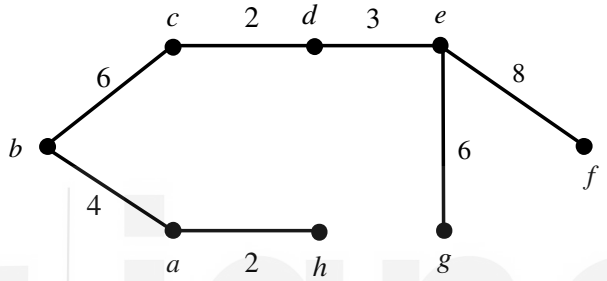


E3) Let us write the edge in increasing order of their weights, as shown below.

edge	ah	cd	de	ab	bc	eg	ad	ef	cg	eh	gh	fg	ac
weight	2	2	3	4	6	6	7	8	8	9	9	10	11

The graph has 8 vertices. So, we have to select 7 edges. Let us begin with $T = \emptyset$. There are two edges namely, ah and cd that have the

smallest weight. Selecting ah and cd both we get $T=\{ah,cd\}$. Next, the smallest weight edge is de, which does not create a cycle. Hence, $T=\{ah,cd,de\}$. The next smallest weight edge is ab, which does not create a cycle. Hence, $T=\{ah,cd,de,ab\}$. Then, we have the edge bc, which has the smallest weight and does not create a cycle. So, $T=\{ah,cd,de,ab,bc\}$. Then we select the edge eg so that $T=\{ah,cd,de,ab,bc,eg\}$. Now we have the edge ad which has the smallest weight among the edges not in T. But the addition of ad to T creates the cycle (a,b,c,d,a). So we discard ad. Next, smallest weight edge is ef, which does not create a cycle. So we get $T=\{ah,cd,de,ab,bc,eg,ef\}$. Now we have $|T|=7$, and hence the algorithm stops. The resulting spanning tree is



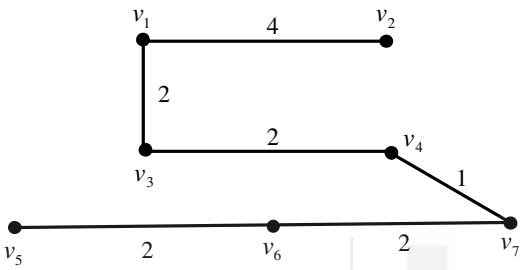
E4) Let us tabulate the weights of the edges as below

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array}\begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{bmatrix} \infty & 4 & 2 & \infty & 3 & \infty & \infty \\ 4 & \infty & \infty & 5 & \infty & \infty & \infty \\ 2 & \infty & \infty & 2 & 6 & 3 & 6 \\ \infty & 5 & 2 & \infty & \infty & 3 & 1 \\ 3 & \infty & 6 & \infty & \infty & 2 & \infty \\ \infty & \infty & 3 & 3 & 2 & \infty & 2 \\ \infty & \infty & 6 & 1 & \infty & 2 & \infty \end{bmatrix} \end{array}$$

Here ∞ in a particular entry indicates that the corresponding edge is missing. Let us begin with the vertex v_1. So $V_T=\{v_1\}$ and $E_T=\phi$. The edge v_1v_3 has the smallest weight. So we get $V_T=\{v_1,v_3\}$ and $E_T=\{v_1v_3\}$. Now we have to select an edge with the smallest weight having exactly one endpoint in V_T. So, we search the rows corresponding to v_1 and v_3. We find that v_3v_4, with $W(v_3v_4)=2$, is such an edge. Thus, we get $V_T=\{v_1,v_3,v_4\}$, and $E_T=\{v_1v_3,v_3v_4\}$. Next, we search the rows corresponding to v_1,v_3 and v_4 to find an edge with the smallest weight having exactly one endpoint in V_T. Such an edge is v_4v_7, with $W(v_4v_7)=1$. Then, we have $V_T=\{v_1,v_3,v_4,v_7\}$, and $E_T=\{v_1v_3,v_3v_4,v_4v_7\}$. Next, we search the rows corresponding to v_1,v_3,v_4 and v_7 to find an edge with the smallest weight, having exactly one endpoint in V_T. Such an edge is v_6v_7, with $W(v_6v_7)=2$. So, we get $V_T=\{v_1,v_3,v_4,v_6,v_7\}$, and $E_T=\{v_1v_3,v_3v_4,v_4v_7,v_6v_7\}$. Now, we search the rows corresponding to v_1,v_3,v_4,v_6, and v_7 to find an edge with the

smallest weight, having exactly one endpoint in V_T. Such an edge is v_5v_6. So, we get $V_T = \{v_1, v_3, v_4, v_5, v_6, v_7\}$, and $E_T = \{v_1v_3, v_3v_4, v_4v_7, v_5v_6, v_6v_7\}$.

We have currently $|E_T| = 5$. So we need to find one more edge. We search the rows corresponding to v_1, v_3, v_4, v_5, v_6 and v_7 to find an edge with the smallest weight, having exactly one endpoint in V_T. We find $W(v_1v_2) = 4$. Then we get $V_T = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and $E_T = \{v_1v_3, v_1v_2, v_3v_4, v_4v_7, v_5v_6, v_6v_7\}$. Now $|E_T| = 6$, hence the algorithm stops. The resulting minimum weight spanning tree is drawn below.



E5) In the beginning we have $S = \{v_1\}$. The current distances of all the vertices from v_1 are shown below, in the distance matrix.

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$d(x)$	0	2	3	∞	∞	∞	∞	2

Here ∞ in a column indicates that there is no edge between v_1 and the vertex corresponding to that column. Since $S \neq V(G)$, we find the vertices $v_2, v_8 \notin S$ such that $d(v_2) = d(v_8) = 2 = \min_{x \notin S} d(x)$. Let us select v_2. Then we get $S = \{v_1, v_2\}$, and the distance matrix as follows.

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$d(x)$	0	2	3	6	∞	∞	8	2

Now $S \neq V(G)$. So we have $v_8 \notin S$ such that $d(v_8) = 2 = \min_{x \notin S} d(x)$.
Then we get $S = \{v_1, v_2, v_8\}$, and the distance matrix as follows.

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$d(x)$	0	2	3	6	∞	6	5	2

Again, $S \neq V(G)$. So we select the vertex $v_3 \notin S$ as $d(v_3) = 3 = \min_{x \notin S} d(x)$. Thus $S = \{v_1, v_2, v_3, v_8\}$ and the distance matrix updates to

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$d(x)$	0	2	3	5	6	6	5	2

Still $S \neq V(G)$. Now the vertices v_4 and v_7 are such that $d(v_4) = d(v_7) = 5 = \min_{x \notin S} d(x)$. Selecting v_4, we get $S = \{v_1, v_2, v_3, v_4, v_8\}$, and the distance matrix remains unchanged.

Still we have $S \neq V(G)$. So we select the vertex v_7 as $d(v_7) = 5 = \min_{x \notin S} d(x)$. Then $S = \{v_1, v_2, v_3, v_4, v_7, v_8\}$, and the distance matrix remains unchanged.

Again $S \neq V(G)$. Now the vertices $v_5, v_6 \notin S$ are such that $d(v_5) = d(v_6) = 6 = \min_{x \notin S} d(x)$. Selecting v_5 we get $S = \{v_1, v_2, v_3, v_4, v_5, v_7, v_8\}$, and the distance matrix remains unchanged.

Again, $S \neq V(G)$. So we select $v_6 \notin S$ to get $S = V(G)$. The distance matrix remains unchanged. The algorithm stops. The distances of the vertices from v_1 are as follows:

$d(v_1, v_2) = 2,\ d(v_1, v_3) = 3,\ d(v_1, v_4) = 5,\ d(v_1, v_5) = 6,\ d(v_1, v_6) = 6,$

$d(v_1, v_7) = 5,\ d(v_1, v_8) = 2.$

ignou THE PEOPLE'S UNIVERSITY