



NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE (NAAC Accredited)

(Approved by AICTE, Affiliated to APJ Abdul Kalam Technological University, Kerala)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE MATERIALS



CS 309 GRAPH THEORY AND COMBINATORICS

VISION OF THE INSTITUTION

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

MISSION OF THE INSTITUTION

NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT DEPARTMENT

- ◆ Established in: 2002
- ◆ Course offered : B.Tech in Computer Science and Engineering
M.Tech in Computer Science and Engineering
M.Tech in Cyber Security
- ◆ Approved by AICTE New Delhi and Accredited by NAAC
- ◆ Affiliated to the University of A P J Abdul Kalam Technological University.

DEPARTMENT VISION

Producing Highly Competent, Innovative and Ethical Computer Science and Engineering Professionals to facilitate continuous technological advancement.

DEPARTMENT MISSION

1. To Impart Quality Education by creative Teaching Learning Process
2. To Promote cutting-edge Research and Development Process to solve real world problems with emerging technologies.
3. To Inculcate Entrepreneurship Skills among Students.
4. To cultivate Moral and Ethical Values in their Profession.

PROGRAMME EDUCATIONAL OBJECTIVES

- PEO1:** Graduates will be able to Work and Contribute in the domains of Computer Science and Engineering through lifelong learning.
- PEO2:** Graduates will be able to Analyse, design and development of novel Software Packages, Web Services, System Tools and Components as per needs and specifications.
- PEO3:** Graduates will be able to demonstrate their ability to adapt to a rapidly changing environment by learning and applying new technologies.
- PEO4:** Graduates will be able to adopt ethical attitudes, exhibit effective communication skills, Teamwork and leadership qualities.

PROGRAM OUTCOMES (POS)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Ability to Formulate and Simulate Innovative Ideas to provide software solutions for Real-time Problems and to investigate for its future scope.

PSO2: Ability to learn and apply various methodologies for facilitating development of high quality System Software Tools and Efficient Web Design Models with a focus on performance

optimization.

PSO3: Ability to inculcate the Knowledge for developing Codes and integrating hardware/software products in the domains of Big Data Analytics, Web Applications and Mobile Apps to create innovative career path and for the socially relevant issues.

COURSE OUTCOMES

CO1	Demonstrate the knowledge of properties of graphs.
CO2	Demonstrate the knowledge of characterization of graphs.
CO3	Demonstrate the knowledge of properties and characterization of trees.
CO4	Distinguish between planar and non-planar graphs and solve problems.
CO5	Use graphs for solving real life problems..
CO6	Develop the efficient algorithms for graph related problems in different domains of engineering and science.

MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
CO1	3	-	-	-	-	-	-	-	-	-	-	-
CO2	3	-	-	-	-	-	-	-	-	-	-	-
CO3	3	-	-	-	-	-	-	-	-	-	-	-
CO4	-	3	3	3	-	-	-	-	-	-	-	-
CO5	-	3	3	3	-	-	-	-	-	-	-	-
CO6	-	3	3	3	-	-	-	-	-	-	-	-

MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES

	PSO1	PSO2	PSO3
CO1	3	-	-
CO2	3	-	-
CO3	3	-	-
CO4	-	3	3
CO5	-	3	3
CO6	-	3	3

Note: H-Highly correlated=3, M-Medium correlated=2, L-Less correlated=1

SYLLABUS

Course code	Course Name	L-T-P Credits	Year of Introduction
CS309	GRAPH THEORY AND COMBINATORICS	2-0-2-3	2016
Prerequisite: Nil			
Course Objectives <ul style="list-style-type: none"> To introduce the fundamental concepts in graph theory, including properties and characterization of graphs/ trees and Graphs theoretic algorithms 			
Syllabus Introductory concepts of graphs, Euler and Hamiltonian graphs, Planar Graphs, Trees, Vertex connectivity and edge connectivity, Cut set and Cut vertices, Matrix representation of graphs, Graphs theoretic algorithms.			
Expected Outcome The Students will be able to <ol style="list-style-type: none"> Demonstrate the knowledge of fundamental concepts in graph theory, including properties and characterization of graphs and trees. Use graphs for solving real life problems. Distinguish between planar and non-planar graphs and solve problems. Develop efficient algorithms for graph related problems in different domains of engineering and science. 			
Text Books <ol style="list-style-type: none"> Douglas B. West, Introduction to Graph Theory, Prentice Hall India Ltd., 2001 Narasimha Deo, Graph theory, PHI, 1979. Robin J. Wilson, Introduction to Graph Theory, Longman Group Ltd., 2010 			
References <ol style="list-style-type: none"> R. Diestel, <i>Graph Theory</i>, free online edition, 2016: diestel-graph-theory.com/basic.html. 			

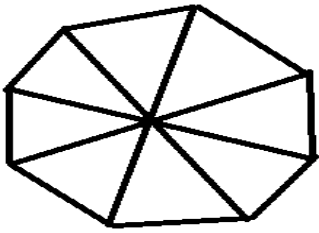

Module	Contents	Hours	End Sem. Exam Marks
I	Introductory concepts - What is graph – Application of graphs – finite and infinite graphs – Incidence and Degree – Isolated vertex, pendent vertex and Null graph. Paths and circuits – Isomorphism, sub graphs, walks, paths and circuits, Connected graphs, disconnect graphs.	09	15 %
II	Euler graphs, Hamiltonian paths and circuits, Dirac's theorem for Hamiltonicity, Travelling salesman problem. Directed graphs – types of digraphs, Digraphs and binary relation	10	15 %
FIRST INTERNAL EXAM			
III	Trees – properties, pendent vertex, Distance and centres - Rooted and binary tree, counting trees, spanning trees.	07	15 %
IV	Vertex Connectivity, Edge Connectivity, Cut set and Cut Vertices, Fundamental circuits, Planar graphs, Different representation of planar graphs, Euler's theorem, Geometric dual, Combinatorial dual.	09	15 %
SECOND INTERNAL EXAM			
V	Matrix representation of graphs- Adjacency matrix, Incidence Matrix, Circuit matrix, Fundamental Circuit matrix and Rank, Cut set matrix, Path matrix	08	20 %
VI	Graphs theoretic algorithms - Algorithm for computer representation of a graph, algorithm for connectedness and components, spanning tree, shortest path.	07	20 %
END SEMESTER EXAM			

Question Paper Pattern

1. There will be *five* parts in the question paper – A, B, C, D, E
2. Part A
 - a. Total marks : 12
 - b. Four questions each having 3 marks, uniformly covering modules I and II; Allfour questions have to be answered.
3. Part B
 - a. Total marks : 18
 - b. Three questions each having 9 marks, uniformly covering modules I and II; Two questions have to be answered. Each question can have a maximum of three subparts.
4. Part C
 - a. Total marks : 12
 - b. Four questions each having 3 marks, uniformly covering modules III and IV; Allfour questions have to be answered.
5. Part D
 - a. Total marks : 18
 - b. Three questions each having 9 marks, uniformly covering modules III and IV; Two questions have to be answered. Each question can have a maximum of three subparts.
6. Part E
 - a. Total Marks: 40
 - b. Six questions each carrying 10 marks, uniformly covering modules V and VI; four questions have to be answered.
 - c. A question can have a maximum of three sub-parts.
7. There should be at least 60% analytical/numerical questions.

QUESTION BANK

MODULE I				
Q:NO :	QUESTIONS	CO	K L	PAGE NO:
1	Consider a graph G with 4 vertices: v_1, v_2, v_3 and v_4 and the degrees of vertices are 3, 5, 2 and 1 respectively. Is it possible to construct such a graph G? If not, why?	CO3	K5	1
2	Draw a disconnected simple graph G_1 with 10 vertices and 4 components and also calculate the maximum number of edges possible in G_1 .	CO1	K1	22

3	<p>List the basic conditions to be satisfied for two graphs to be isomorphic. Predict the following graphs are isomorphic. Explain with valid reasons.</p> <div style="display: flex; justify-content: space-around; align-items: center;">   </div>	CO1	K1	15
4	Write any two applications of graphs with sufficient explanation.	CO1	K3	3
5	Show that the number of vertices of odd degree in a graph is always even.	CO1	K2	12
6	Describe graph, subgraph, Edge –disjoint subgraph and vertex-disjoint subgraph with example.	CO1	K2	17

MODULE II

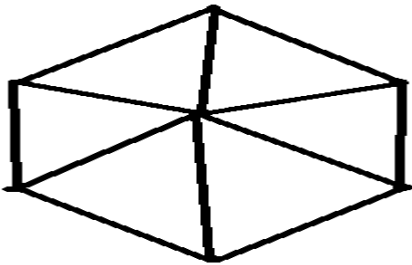
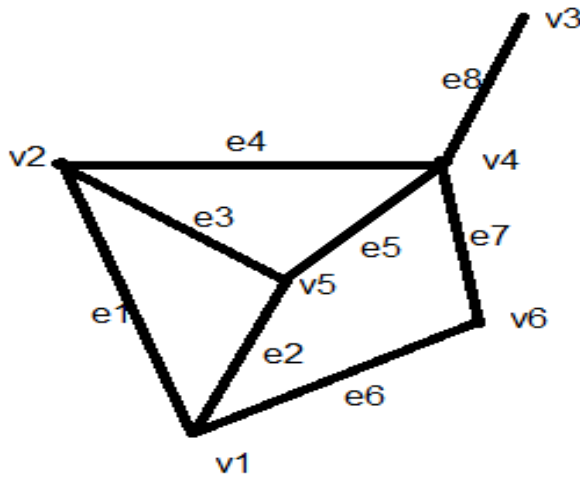
1	Give an example of a Hamiltonian graph and compare it with bipartite graph.	CO2	K2	35
2	Explain the terms: (a) Euler Graphs. (b) Digraphs.	CO2	K5	27
3	Distinguish between Directed and Undirected graphs with examples.	CO2	K2	43
4	Describe Euler line and Euler graph with example.	CO2	K4	27
5	Define Hamiltonian path and Hamiltonian circuit with example.	CO2	K2	35
6	Give an example of digraphs and list out the types of digraphs.	CO2	K1	43
7	Show that a given connected graph G is an Euler graph if and only if all vertices of G are of even degree.	CO2	K2	29
8	Show that in a complete graph with n vertices there are $(n-1)/2$ edge –disjoint Hamiltonian circuits, if n is an odd number ≥ 3 .	CO2	K2	38

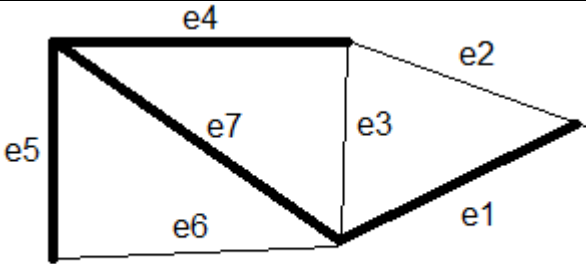
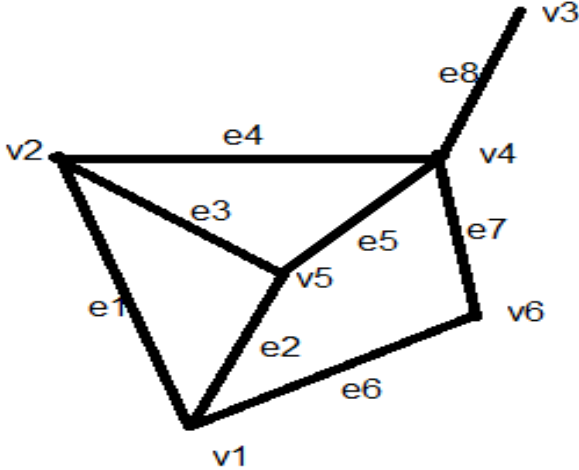
MODULE III

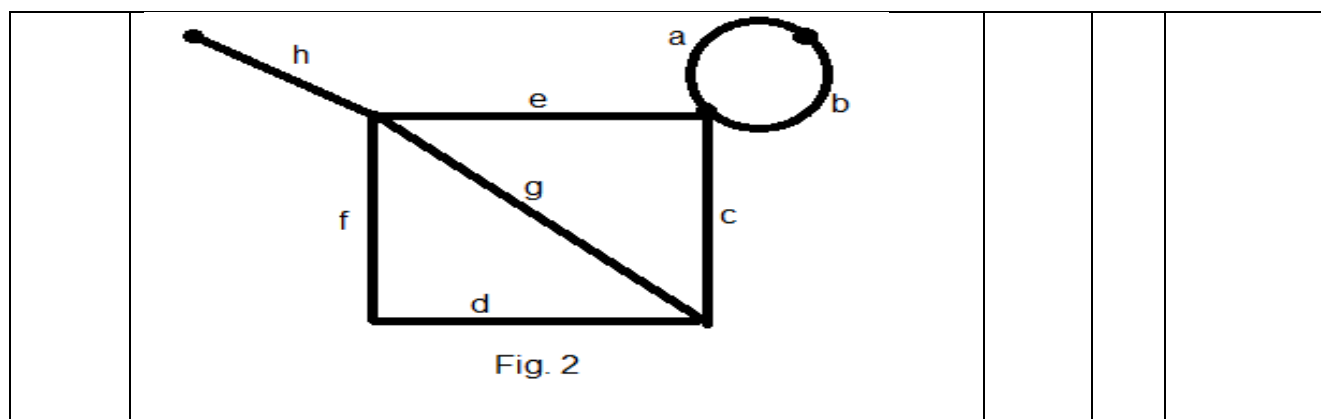
1	Define tree and draw trees with one, two, three and four vertices.	CO3	K1	57
2	Show that there is one and only one path between every pair of vertices in a tree, T.	CO3	K2	58
3	Show that a tree with n vertices has n-1 edges.	CO3	K2	59
4	Define spanning tree and show that every connected graph has at least one spanning tree.	CO3	K1	80
5	Define distance in a tree and show that the distance between vertices of a connected graph is a metric.	CO3	K1	84
6	List the properties of a binary tree, how to find the path length of a binary tree and what is the importance of it.	CO3	K1	69
7	Define tree, null tree and decision tree.	CO3	K1	57
8	List the conditions to say that a graph G with n vertices is called a tree.	CO3	K1	62

MODULE IV

1	Write about how to find dual of a subgraph.	CO4	K3	128
2	Show that Kuratowski's second graph is nonplanar.	CO4	K2	115
3	Assess the relationship between a planar graph G and its dual G^* .	CO4	K5	129
4	Draw the geometric dual (G^*) of the graph G given below and also check whether G and G^* are self dual or not, substantiate your answer clearly.	CO4	K1	128

				
5	Define planar graph and prove that the complete graph of five vertices is nonplanar.	CO4	K1	112
6	Show that a connected planar graph with n vertices and e edges has $e - n + 2$ regions.	CO4	K2	122
MODULE V				
1	Analyze the observations can be made immediately about the incidence matrix A of the graph in Fig.1.  Fig.1	CO5	K4	143
2	Write down the fundamental circuit matrix with respect to the spanning tree shown in heavy lines of the graph in Fig.2 and prove that “If B is a circuit matrix of a connected graph G with e edges and n vertices, rank of $B = e - n + 1$.”	CO5	K3	148

	 <p>Fig. 2</p>			
3	Write down the circuit matrix and path matrix $P(v_2, v_6)$ for the graph in Fig.1 and write down the observations made from the circuit matrix and path matrix.	CO5	K3	148
4	Analyze the observations can be made immediately about the adjacency matrix X of the following graph.	CO5	K4	138
				
5	Explain in detail about the relationships among A_f , B_f , and C_f .	CO6	K5	158
6	Write down the cut-set matrix and fundamental cut-set matrix for the following graph.	CO6	K3	154



MODULE VI

1	Write down Dijkstra's algorithm and find out the shortest path between B and G vertices of the following graph.	CO6	K3	176
<p>Fig. 3</p>		CO6	K5	169
2	Explain the working of connectedness and components algorithm with flow chart.	CO6	K5	164
3	What do you mean by efficiency of an algorithm and explain in detail about the input and output computer representation of a graph.	CO6	K3	171
4	Write about spanning tree algorithm with flow chart.	CO6	K5	175

6	Explain all pair shortest path algorithm with flow chart.	CO6	K5	184
---	---	-----	----	-----

APPENDIX 1

CONTENT BEYOND THE SYLLABUS

SL.NO:	TOPIC	PAGE NO:
1	THE GIRTH OF A GRAPH	188
2	ON-LINE COLORING LINE COLORING – A TWO PERSON GAME	188
3	INTERVAL GRAPH	191
4	VISIBILITY GRAPH	191
5	THRESHOLD GRAPH	192

MODULE NOTES

CS309 GRAPH THEORY AND COMBINATORICS

Module I

Introduction

1. What is graph?

A linear graph (or simply a graph) $G = (V, E)$ consists of a set of objects $V = \{v_1, v_2, \dots\}$ called vertices, and another set $E = \{e_1, e_2, \dots\}$, whose elements are called edges, such that each edge e_k is identified with an unordered pair (v_i, v_j) of vertices.

- i. The vertices v_i, v_j associated with edge e_k are called the end vertices of e_k .
- ii. The most common representation of a graph is by means of a diagram, in which the vertices are represented as points and each edge as a line segment joining its end vertices.

2. Self-loop

An edge having the same vertex as both its end vertices is called a self-loop (or simply a loop).

3. Parallel edges

More than one edge associated with a given pair of vertices are referred to as parallel edges.

①

4. Simple graph

A simple graph that has neither self-loops nor parallel edges is called a simple graph.

5. General graph

Graphs that allow parallel edges and self-loops are known as general graph.

i. It is immaterial whether the lines of a graph are drawn straight or curved, long or short.

ii. The incidence between the edges and vertices is important.

6. Graph = linear complex, 1-complex or a one-dimensional complex.

Vertex = node, junction, point, 0-cell or an 0-complex.

edge = branch, a line, an element, a 1-cell, an arc and a 1-simplex.

APPLICATIONS OF GRAPHS

Graph theory has a very wide range of applications in

- engineering,
- physical,
- social,
- biological sciences,
- linguistics,
- numerous other areas.

A graph can be used to represent almost any physical situation involving discrete objects and a relationship among them.

Examples :

1. Königsberg Bridge Problem.

- solved by Leonhard Euler (1707-1783) in 1736, by means of a graph.

Two islands, C and D, formed by the Pregel River in Königsberg (then the capital of East Prussia but now renamed Kaliningrad and in West Soviet Russia) were connected to each other and to the banks A and B with seven bridges, as shown in Fig.1

- a) The problem was to start at any of the four land areas of the city A, B, C or D, walk over each of the seven bridges exactly once, and return to the starting point (without swimming across the river, of course).
- b) Euler represented this situation by means of a graph, as shown in Fig. 2.

- The vertices represent the land areas and the edges represent the bridges.

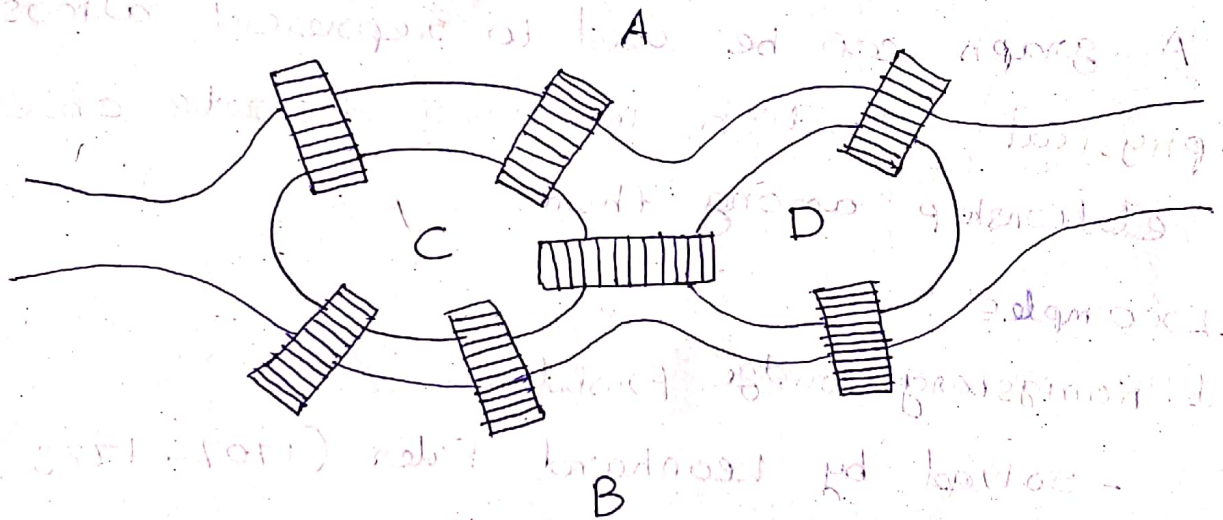


Fig: 1. Königsberg bridge Problem.

- c) The Königsberg bridge problem is the same as the problem of drawing figures without lifting the pen from the paper and without retracing a line.

2. Utilities Problem:

There are three houses, H_1 , H_2 and H_3 , each to be connected to each of the three utilities - water (W), gas (G), and electricity (E) - by means

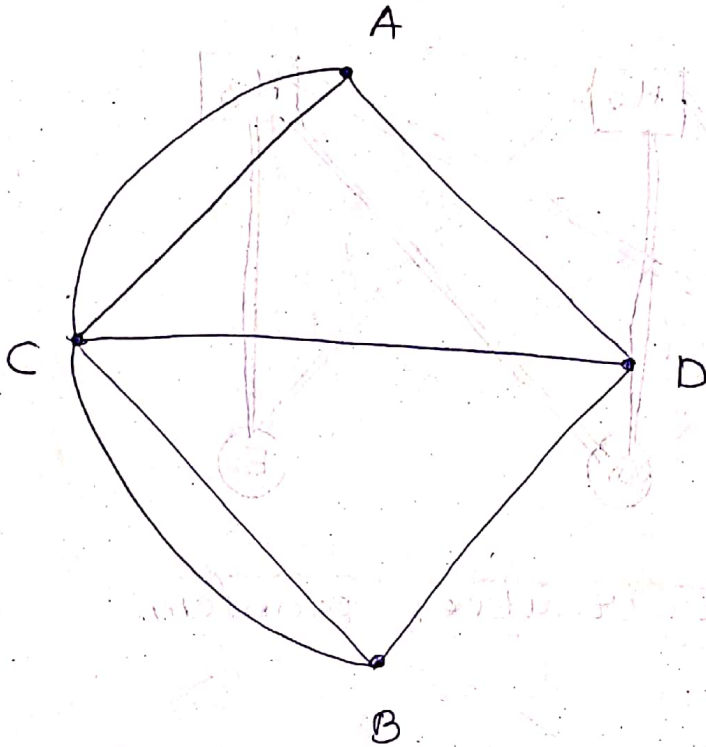


Fig. 2: Graph of Königsberg bridge problem.

of conduits. Is it possible to make such connections without any crossover of the conduits?

Fig. 3 shows the three-utilities problem.

Figure 4 shows how this problem can be represented by a graph.

- a) The conduits are shown as edges while the houses and utility supply centers are vertices.

Figure 1 cannot be drawn in the plane without edges crossing over - thus the answer to the problem is no.

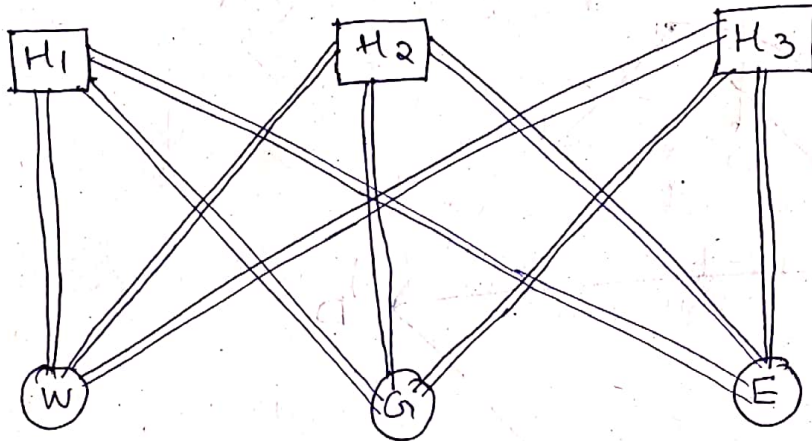


Fig. 3 Three-utilities problem.

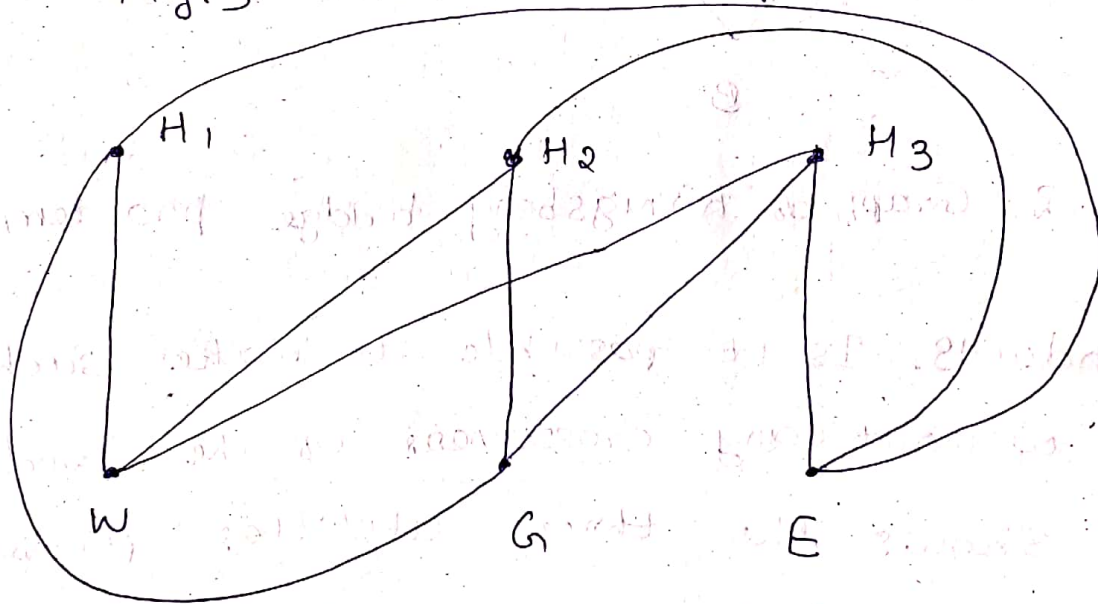


Fig. 4 Graph of three-utilities problem.

(6)

3. Electrical network problems:

a) Properties such as transfer function and input impedance of an electrical network are functions of only two factors:

1. The nature and value of the elements forming the network, such as resistors, inductors, transistors and so forth.
2. The way these elements are connected together, that is, the topology of the network.

Since there are only few different types of electrical elements

a. the variations in networks are chiefly due to the variations in topology.

b. ~~the~~ Electrical network analysis and synthesis are mainly the study of network topology.

b) The topology of a network is studied by means of its graph.

- In drawing a graph of an electrical network

- the junctions are represented by vertices, and
- branches are represented by edges.

regardless of the nature and size of the electrical elements.

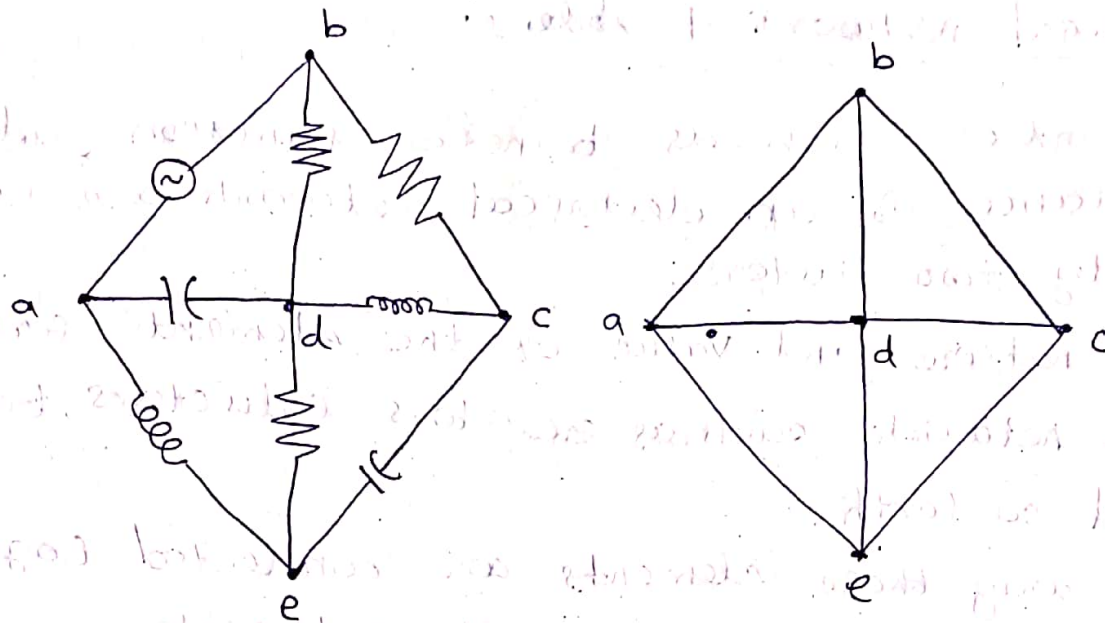


Fig. 5 Electrical network and its graph.

4. Seating Problem

- i. Nine members of a new club meet each day for lunch at a round table.
- ii. They decide to sit such that every member has different neighbors at each lunch.
- iii. How many days can this arrangement last?

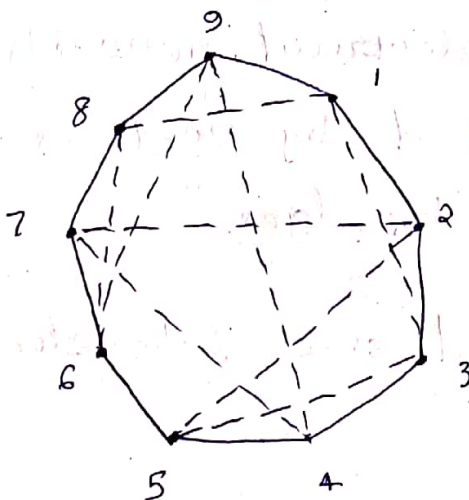


Fig. 6 Arrangements at a dinner table (8)

Seating arrangements

- 1 2 3 4 5 6 7 8 9 1
- 1 3 5 2 7 4 9 6 8 1
- 1 5 7 3 9 2 8 4 6 1
- 1 7 9 5 8 3 6 2 4 1

In general it can be shown that for n people the number of such possible arrangement is

$$\frac{n-1}{2}$$

if n is odd,

and

$$\frac{n-2}{2}$$

if n is even.

FINITE AND INFINITE GRAPHS

A graph with a finite number of vertices as well as a finite number of edges is called a finite graph; otherwise, it is an infinite graph.

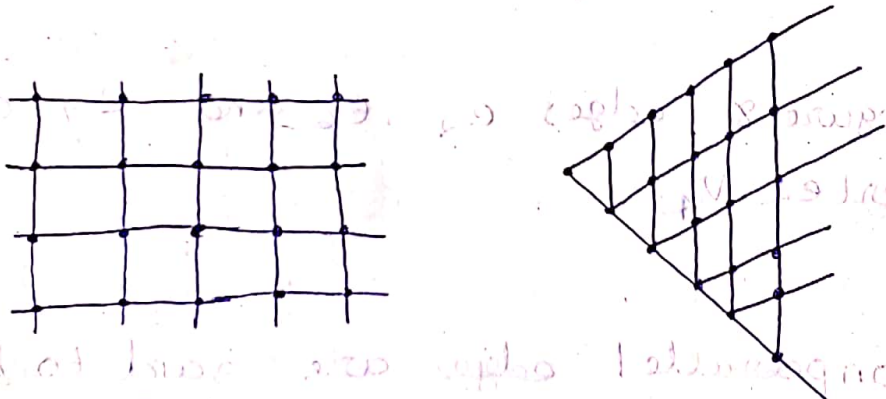


Fig. 7 Portions of two infinite graphs.

INCIDENCE AND DEGREE

i. incident with (on or to)

When a vertex v_i is an end vertex of some edge e_j , v_i and e_j are said to be incident with each other.

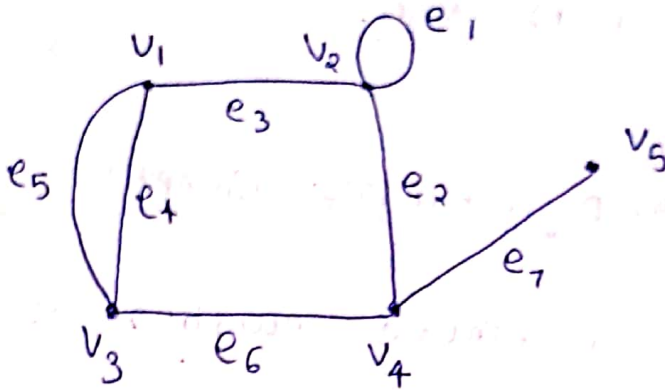


Fig. 8 A graph with five vertices and seven edges.

Eg:

In Figure 8 edges e_2 , e_6 and e_7 are incident with vertex v_4 .

ii. adjacent

Two nonparallel edges are said to be adjacent if they are incident on a common vertex.

eg:

e_2 and e_7 in Fig. 8 are adjacent.

iii. ^{rdly}

two vertices are said to be adjacent if they are the end vertices of the same edge.

eg: In Figure 8, v_4 and v_5 are adjacent but v_1 and v_4 are not.

iii. degree, $d(v_i)$ of vertex v_i .

The number of edges incident on a vertex v_i , with self-loops counted twice, is called the degree of vertex v_i .

In Figure 8, for example,

$$d(v_1) = d(v_3) = d(v_4) = 3,$$

$$d(v_4) = 3$$

$$d(v_2) = 4 \text{ and}$$

$$d(v_5) = 1.$$

1 The degree of a vertex is sometimes referred to as its valency.

2 Let G be a graph with e edges and n vertices

a) since each edge contributes two degrees

b) the sum of the degrees of all vertices in G is twice the number of edges in G .

$$\text{i.e., } \sum_{i=1}^n d(v_i) = 2e. \quad (1)$$

Theorem:

The number of vertices of odd degree in a graph is always even.

Proof:

If we consider the vertices with odd and even degrees separately, the quantity in the left side of Eq.(1) can be expressed as the sum of two sums, each taken over vertices of even and odd degrees, respectively, as follows:

$$\sum_{i=1}^n d(v_i) = \sum_{\text{even}} d(v_i) + \sum_{\text{odd}} d(v_k) \quad (2)$$

Since the left-hand side in Eq.(2) is even, and the first expression on the right-hand side is even (being a sum of even numbers), the second expression must also be even:

$$\sum_{\text{odd}} d(v_k) = \text{an even number} \quad (3)$$

Because in Eq.(3) each $d(v_k)$ is odd, the total number of terms in the sum must be even to make the sum an even number. Hence the theorem.

(12)

1.11. regular graph.

A graph in which all vertices are of equal degree is called a regular graph (or simply a regular).

Eg: The graph of three utilities is a regular of degree three.

ISOLATED VERTEX, PENDANT VERTEX, AND NULL GRAPH

1.12. isolated vertex

A vertex having no incident edge is called an isolated vertex.

Isolated vertices are vertices with zero degree.

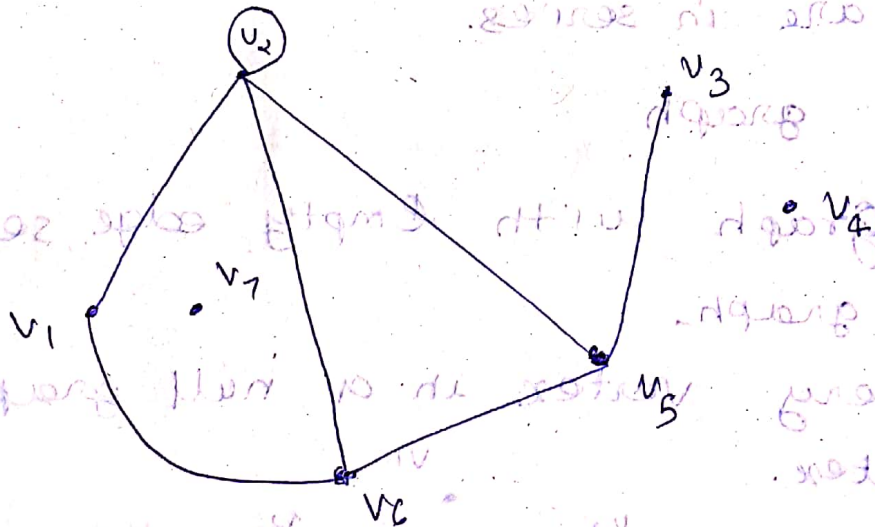


Fig. 9 Graph containing isolated vertices, series edges, and a pendant vertex.

Eg: vertices v_4 and v_7 in Figure 9.

iii. pendant vertex (end vertex)

A vertex of degree one is called a pendant vertex or an end vertex.

eg: Vertex v_3 in Fig. 9 is a pendant vertex.

iii. series

Two adjacent edges are said to be in series if their common vertex is of degree two.

eg: In Fig. 9, the two edges incident on v_1 are in series.

iv. null graph

A graph with empty edge set is called a null graph.

- Every vertex in a null graph is an isolated vertex.

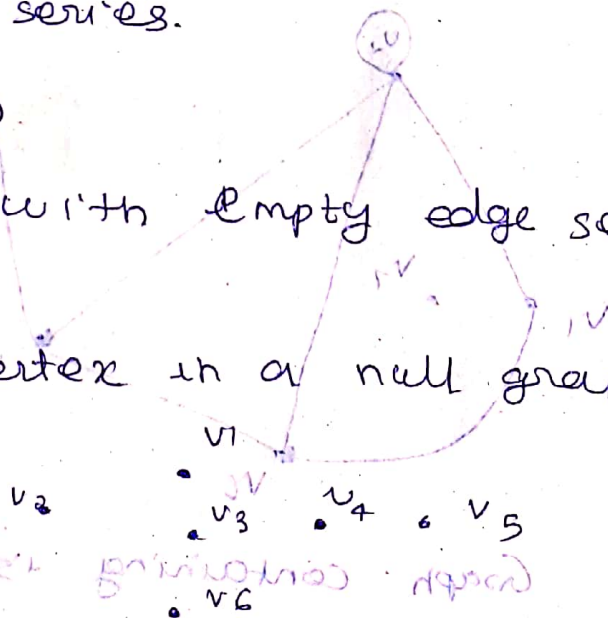


Fig. 10. Null graph of 7 vertices.

PATHS AND CIRCUITS

Deals mainly with the nature of connectivity in graphs.

i. ISOMORPHISM

a) Two graphs are thought of as equivalent if they have identical behavior in terms of graph-theoretic properties.

b) Two graphs G and G_1 are said to be isomorphic if there is a one-to-one correspondence between their vertices and between their edges such that the incidence relationship is preserved.

c) Two isomorphic graphs must have

1. The same number of vertices.

2. The same number of edges.

3. An equal number of vertices with a given degree.

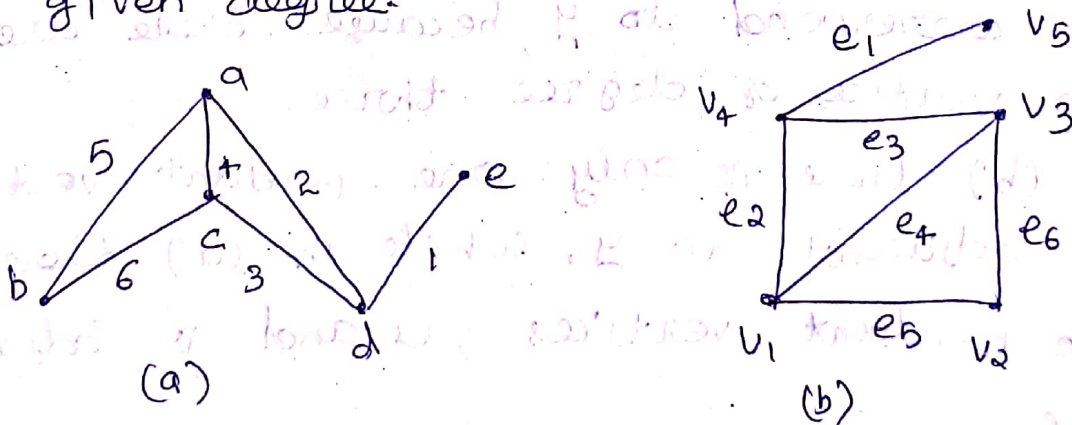


Fig. 11 Isomorphic graphs.

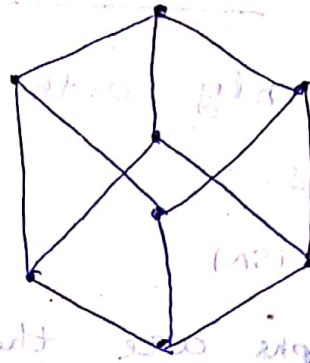
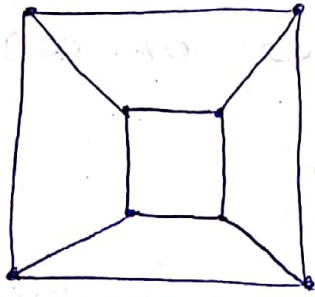


Fig. 12 Isomorphic graphs.

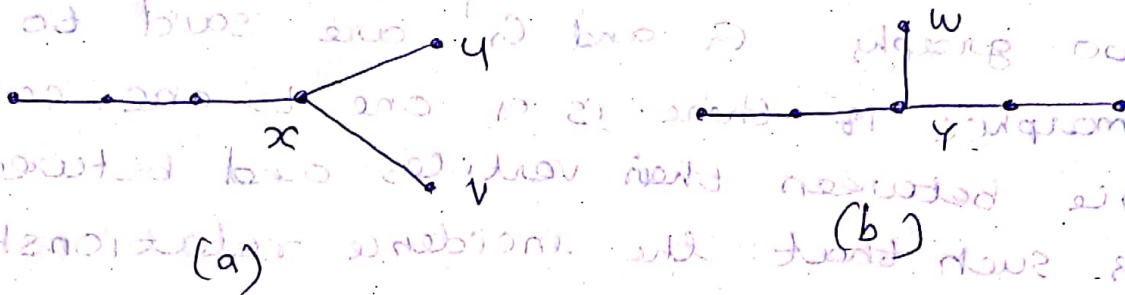


Fig. 13 Two graphs that are not isomorphic.

The two graphs shown in Figure 13 satisfy all three conditions, yet they are not isomorphic.

a. If the graph in Fig. 13(a) were to be isomorphic to the one in (b), vertex x must correspond to y , because there are no other vertices of degree three.

b. In (b) there is only one pendant vertex, w , adjacent to y , while in (a) there are two pendant vertices, u and v , adjacent to x .

ii. SUBGRAPHS

A graph g is said to be a subgraph of a graph G if all the vertices and all the edges of g are in G , and each edge of g has the same end vertices in g as in G .

1. Every graph is its own subgraph.
2. A subgraph of a subgraph of G is a subgraph of G .
3. A single vertex in a graph G is a subgraph of G .
4. A single edge in G , together with its end vertices, is also a subgraph of G .

* Ex

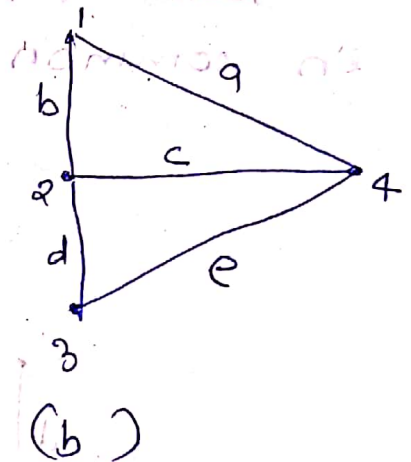
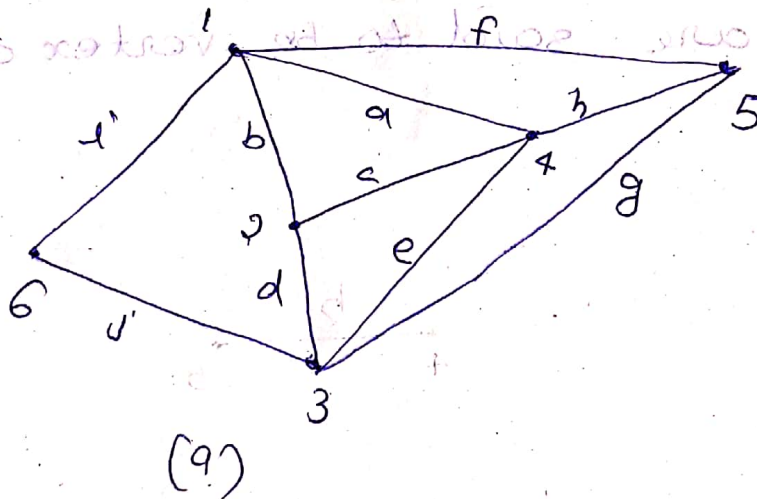
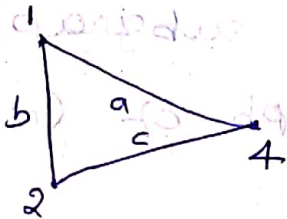


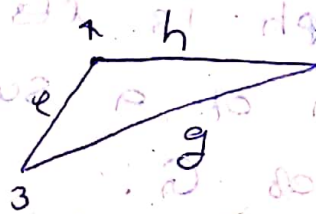
Fig. 14 Graph (a) and one of its subgraphs (b).

iii. Edge - Disjoint Subgraphs:

Two subgraphs g_1 and g_2 of a graph G are said to be edge disjoint if g_1 and g_2 do not have any edges in common.



(a)

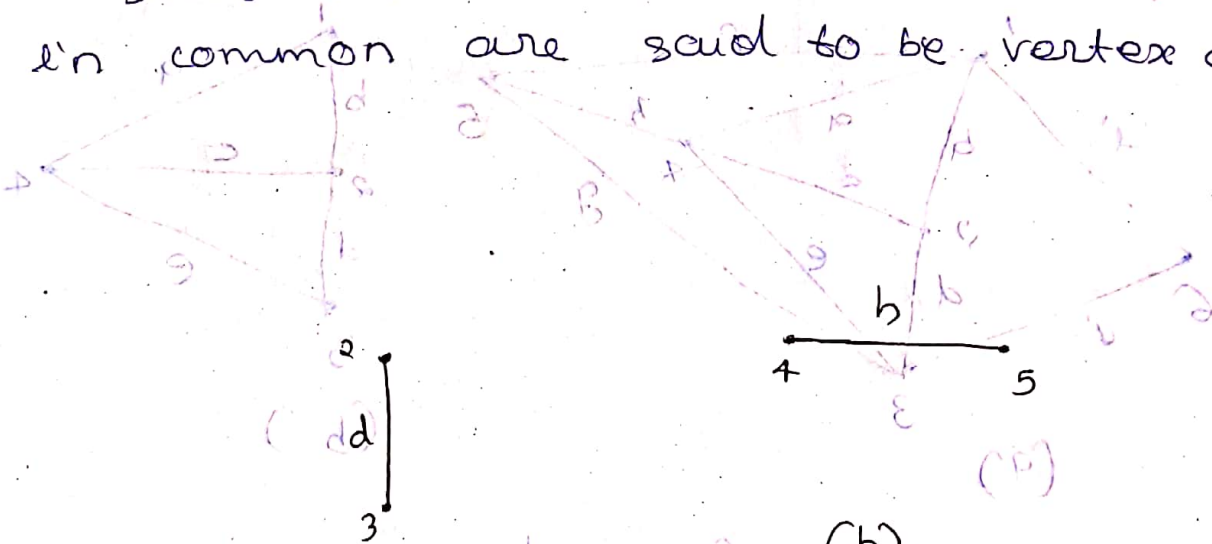


(b)

Fig. 15 Edge disjoint subgraphs of Fig. 14(a)

xiv. Vertex disjoint subgraphs

subgraphs that do not even have vertices in common are said to be vertex disjoint.



(a)



(b)

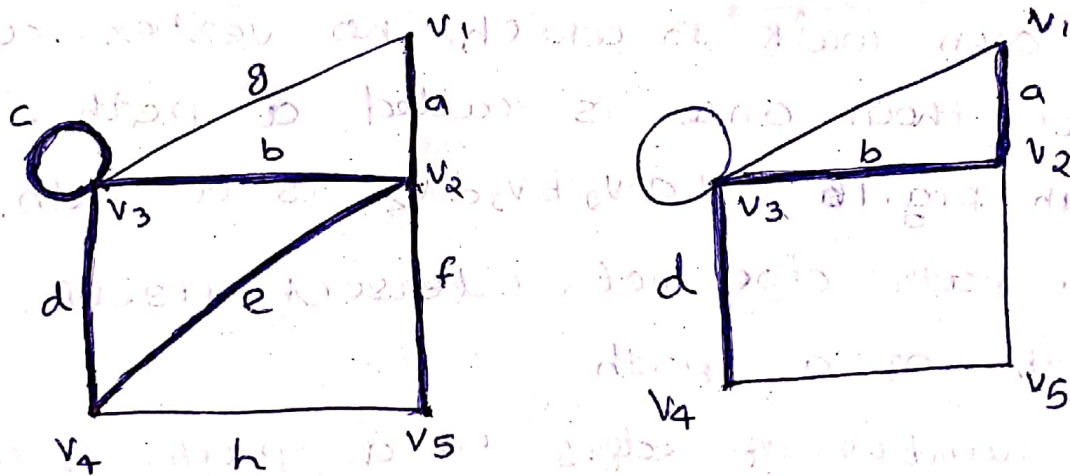
Fig. 15(a) vertex disjoint subgraphs of Fig. 14(a)

WALKS, PATHS and CIRCUITS

1. Walk (edge train or a chain)

A walk is defined as a finite alternating sequence of vertices and edges, beginning and ending with vertices, such that each edge is incident with the vertices preceding and following it.

- No edge appears more than once in a walk.
- A vertex may appear more than once.



(a) An open walk

(b) A Path of Length Three

Fig. 16. A walk and a path.

Eg: $v_1-a-v_2-b-v_3-c-v_3-d-v_4-e-v_2-f-v_5$

2. terminal vertices.

vertices with which a walk begins and ends are

(19).

called its terminal vertices. Eg: in Fig 16(a) v_1 & v_2

3 closed walk

A walk which begins and ends at the same vertex is called a closed walk.

4. Open walk

A walk that is not closed, i.e. the terminal vertices are distinct, is called an open walk.

Eg: Fig. 16(a)

5. Path (A simple path or an elementary path)

An open walk in which no vertex appears more than once is called a path.

eg. in Fig. 16, $v_1 a v_2 b v_3 d v_4$ is a path.

- a path does not intersect itself.

6 length of a path

The number of edges in a path is called the length of a path.

- An edge which is not a self-loop is a path of length one.

- a self-loop can be included in a walk but not a path.

7. intermediate Vertices

The terminal vertices of a path are of degree one, and the rest of the vertices are of degree two.

8. Circuit

A closed walk in which no vertex (except the initial and the final vertex) appears more than once is called a circuit.

eg: In fig. 16(9) $v_2 b v_3 d v_4 e v_2$.

- every vertex in a circuit is of degree two.

- also called

- a cycle

- elementary cycle

- circular path and

- polygon.

- every self-loop is a circuit, but not every circuit is a self-loop.

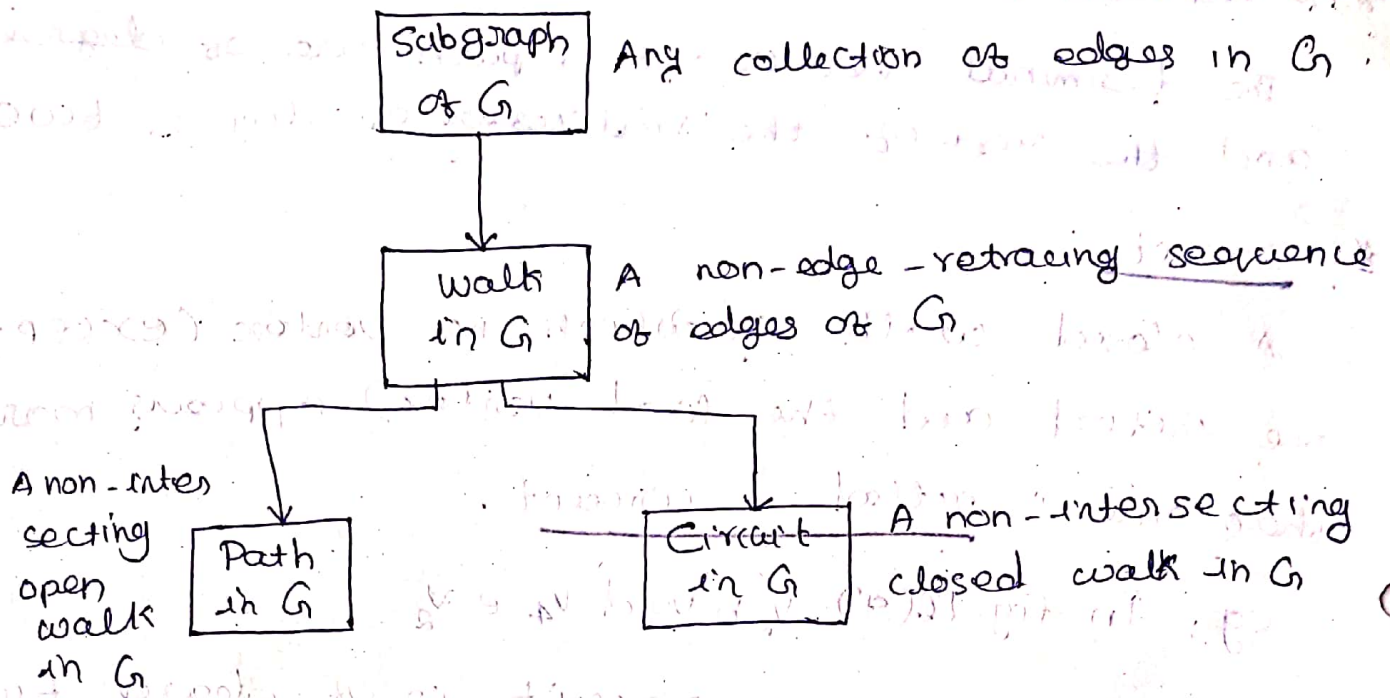


Fig. 17. Walks, paths and circuits as subgraphs.

CONNECTED GRAPHS, DISCONNECTED GRAPHS AND COMPONENTS

1. Connected Graph

A graph G is said to be connected if there is at least one path between every pair of vertices in G . eg: Fig. 16(a) is connected.

2. Disconnected graph

A graph G is said to be disconnected if there is no path between every pair of vertices in G .

(22)

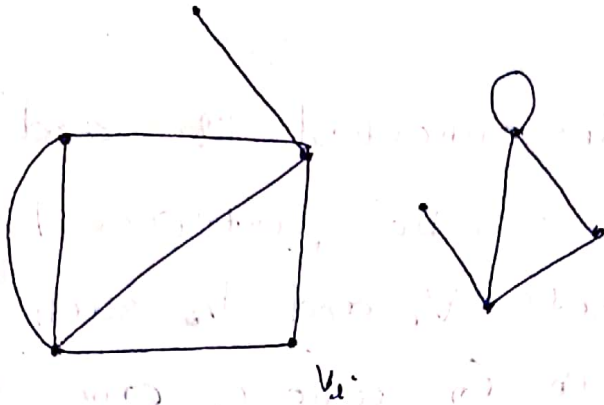


Fig: 17 A disconnected graph with two components.

- A null graph of more than one vertex is disconnected.

3. Component.

In a disconnected graph

Disconnected graph consists of two or more connected graphs. Each of these connected subgraphs is called a component:

THEOREM 2

A Graph G is disconnected if and only if its vertex set V can be partitioned into two nonempty, disjoint subsets V_1 and V_2 such that there exists no edge in G whose one end vertex is in subset V_1 and the other in subset V_2 .

Proof:

Suppose that such a partitioning exists. Consider two arbitrary vertices a and b of G , such that $a \in V_1$ and $b \in V_2$. No path can exist between vertices a and b ; otherwise, there would be at least one edge whose one end vertex would be in V_1 and the other in V_2 . Hence, if a partition exists, G is not connected.

Conversely, let G be a disconnected graph. Consider a vertex a in G . Let V_1 be the set of all vertices are joined by paths to a . Since G is disconnected, V_1 does not include all vertices of G . The remaining vertices will form a set (nonempty) V_2 . No vertex in V_1 is joined to any in V_2 by an edge. Hence the partition.

Theorem 3

If a graph (connected or disconnected) has exactly two vertices of odd degree, there must be a path joining these two vertices.

Proof:

Let G be a graph with all even vertices except vertices v_1 and v_2 , which are odd. For every graph and therefore for every component of a disconnected graph, no graph can have an odd number of odd vertices. Therefore, in graph G , v_1 and v_2 must belong to the same component, and hence must have a path between them.

Theorem 4

A simple graph (i.e., a graph without parallel edges or self-loops) with n vertices and k components can have at most $(n-k)(n-k+1)/2$ edges.

Proof:

Let the number of vertices in each of the k components of a graph G be n_1, n_2, \dots, n_k .

Thus we have

$$n_1 + n_2 + \dots + n_k = n$$

$$n_i \geq 1$$

The proof of the theorem depends on an algebraic inequality.

$$\sum_{i=1}^k n_i^2 \leq n^2 - (k-1)(2n-k). \quad \text{--- (1)}$$

Now the maximum number of edges in the i^{th} component of G (which is a simple connected graph) is $\frac{1}{2} n_i (n_i - 1)$.

Therefore, the maximum number of edges in G is

$$\begin{aligned} \frac{1}{2} \sum_{i=1}^k (n_i - 1) n_i &= \frac{1}{2} \left(\sum_{i=1}^k n_i^2 \right) - \frac{n}{2} \\ &\leq \frac{1}{2} [n^2 - (k-1)(2n-k)] - \frac{n}{2} \quad \text{from (1)} \\ &= \underline{\underline{\frac{1}{2} (n-k)(n-k+1)}}. \end{aligned}$$

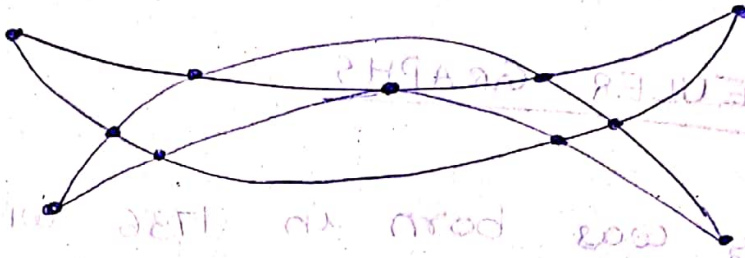
EULER GRAPHS

- * Graph theory was born in 1736 with Euler's famous paper in which he solved the Königsberg bridge problem.
- * In what type of graph G is it possible to find a closed walk running through every edge of G exactly once?
such a walk is now called an Euler line.

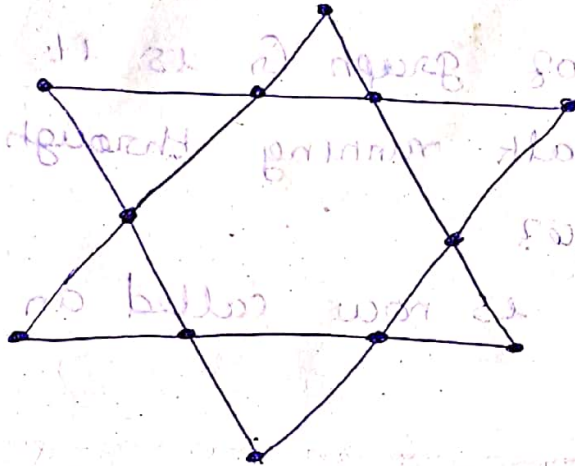
- * Euler graph

If some closed walk in a graph contains all the edges of the graph, then the walk is called an Euler line and the graph an Euler graph.

- Euler graphs do not have any isolated vertices and are therefore connected.
- since the Euler line (which is a walk) contains all the edges of the graph, an Euler graph is always connected.



(a) Mohamud's scimitars



(b) Star of David

Fig 2.1 Two Euler graphs

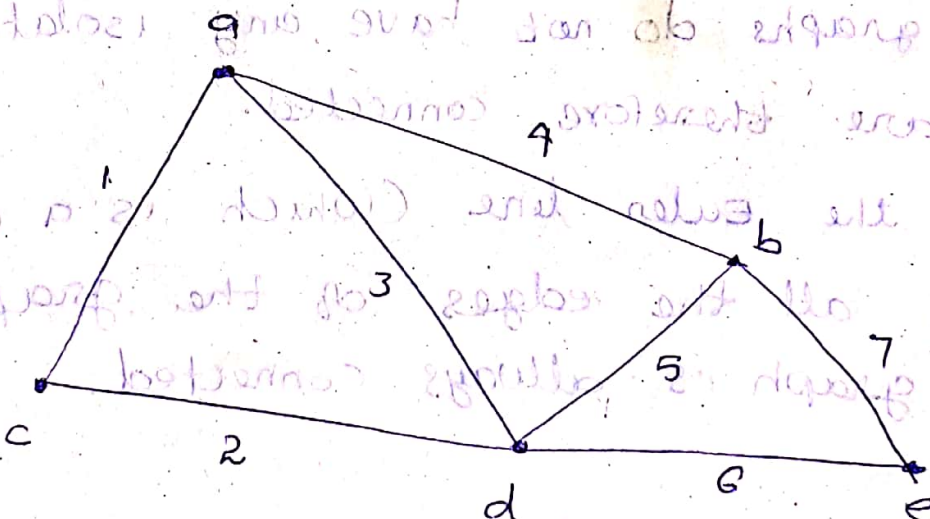


Fig 2.2 Unicausal graph

① ②

THEOREM 2-1

A given connected graph G is an Euler graph if and only if all vertices of G are of even degree.

Proof:

Suppose that G is an Euler graph. It therefore contains an Euler line (which is a closed walk). In tracing this walk we observe that every time the walk meets a vertex v it goes through two "new" edges incident on v - with one we "entered" v and with the other "exited". This is true not only of all intermediate vertices of the walk but also of the terminal vertex, because we "exited" and "entered" the same vertex at the beginning and end of the walk, respectively. Thus if G is an Euler graph, the degree of every vertex is even.

To prove the sufficiency of the condition, assume that all vertices of G are of even degree. Now we construct a walk starting at an arbitrary vertex v and going through the edges of G such that no edge is traced more than once. We continue tracing as far as possible. Since every vertex is

(3)

(+)

of even degree, we can exit from every vertex we enter; the tracing cannot stop at any vertex but v . And since v is also of even degree, we shall eventually reach v when the tracing comes to an end. If this closed walk h we just traced includes all the edges of G , G is an Euler graph.

If not, we remove from G all the edges in h and obtain a subgraph h' of G formed by the remaining edges. Since both G and h have all their vertices of even degree, the degrees of the vertices of h' are also even. Moreover, h' must touch h at least at one vertex a , because G is connected. Starting from a , we can again construct a new walk in graph h' . Since all the vertices of h' are of even degree, this walk in h' must terminate at vertex a ; but this walk in h' can be combined with h to form a new walk, which starts and ends at vertex v and has more edges than h . This process can be repeated until we obtain a closed walk that traverses all the edges of G . Thus G is an Euler graph.

* Königsberg Bridge Problem:

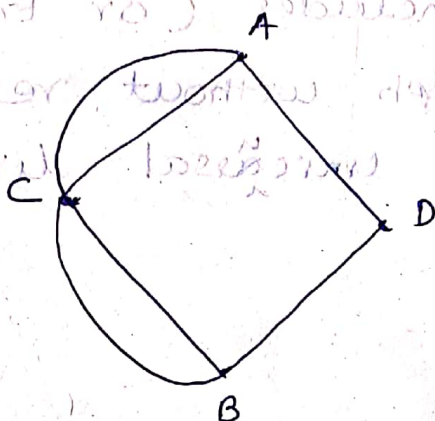


Fig: 2.3 Graph of Königsberg bridge problem.

Now looking at the graph of the Königsberg bridges we find that

- not all its vertices are of even degree,

- Hence it is not an Euler graph.

- Thus it is not possible to walk over each of the seven bridges exactly once and return to the starting point.

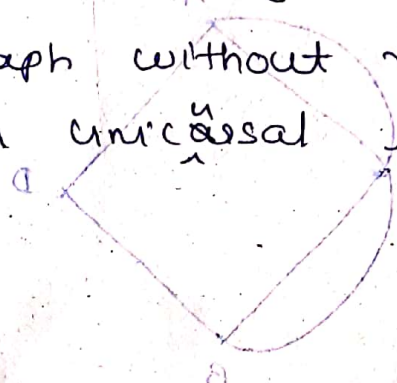
* The problem common to encounters Euler lines in various puzzles is to find

- how a given picture can be drawn in one continuous line

- without retracing and without lifting the pencil from the paper.

* Open Euler line (Unicursal line)

An open walk that includes (or traces or covers) all edges of a graph without retracing any edge is called as a unicursal line or an open Euler line.



- Unicursal graph

A (connected) graph that has a unicursal line will be called a unicursal graph.

- By adding an edge between the initial and final vertices of a unicursal line we shall get an Euler line.

- A connected graph is unicursal if and only if it has exactly two vertices of odd degree.

* Theorem 2-2

A connected graph G is an Euler graph if and only if it can be decomposed into circuits.

Proof:

Suppose graph G can be decomposed into circuits; that is, G is a union of edge-disjoint circuits. Since the degree of every vertex in

⑥

⑦

a circuit is two, the degree of every vertex in G is even. Hence G is an Euler graph.

conversely, let G be an Euler graph, consider a vertex v_1 . There are at least two edges incident at v_1 . Let one of these edges be between v_1 and v_2 . Since vertex v_2 is also of even degree, it must have at least another edge, say between v_2 and v_3 . Proceeding in this fashion, we eventually arrive at a vertex that has previously been traversed, thus forming a circuit Γ . Let us remove Γ from G . All vertices in the remaining graph (not necessarily connected) must also be of even degree. From the remaining graph remove another circuit in exactly the same way as we removed Γ from G . Continue this process until no edges are left. Hence the theorem.

* Arbitrarily Traceable Graphs:

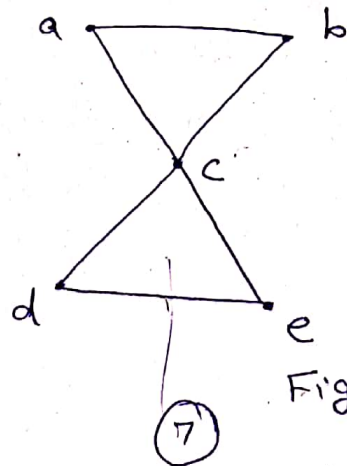


Fig. 2.4 Arbitrarily traceable graph from c .

— whenever one arrives at a vertex v one shall select any edge which has not been previously traversed, such a graph is called an arbitrarily traceable graph from vertex v .

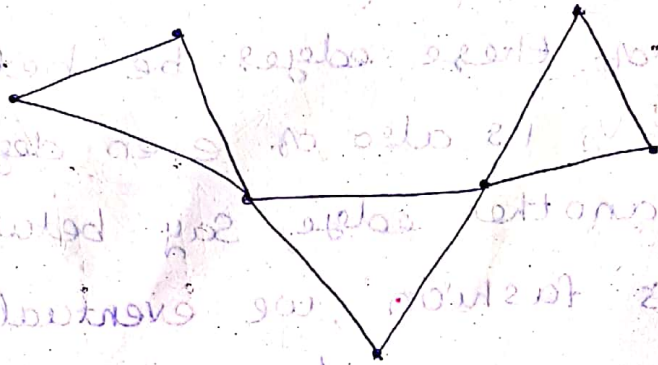


Fig. 2.5 Euler graph: not arbitrarily traceable.

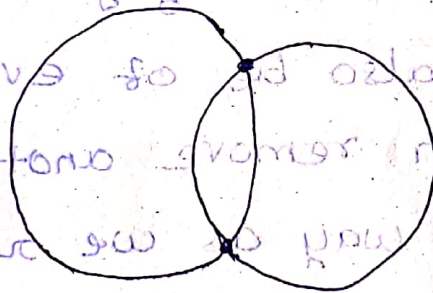


Fig. 2.6 Arbitrarily traceable graph from all vertices.



8

7

* HAMILTONIAN PATHS AND CIRCUITS

A Hamiltonian circuit in a connected graph is defined as a closed walk that traverses every vertex of G exactly once, except of course the starting vertex, at which the walk also terminates.

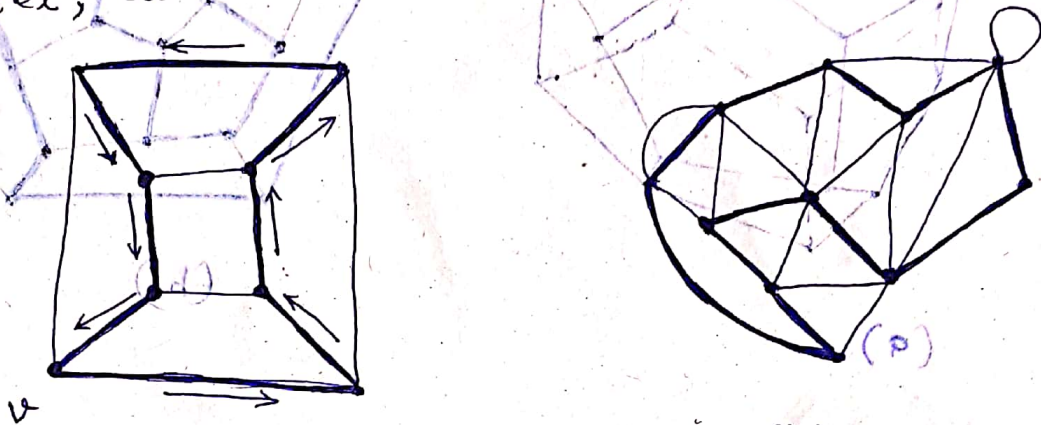


Fig. 2.7 Hamiltonian circuits.

A circuit in a connected graph G is said to be Hamiltonian if it includes every vertex of G . Hence a Hamiltonian circuit in a graph of n vertices consists of exactly n edges.

* Hamiltonian Path:

If we remove any one edge from a Hamiltonian circuit, we are left with a path. This path is called a Hamiltonian path.

- The length of a Hamiltonian path (if it exists) in a connected graph of n vertices is $n-1$.

(9)

(10)

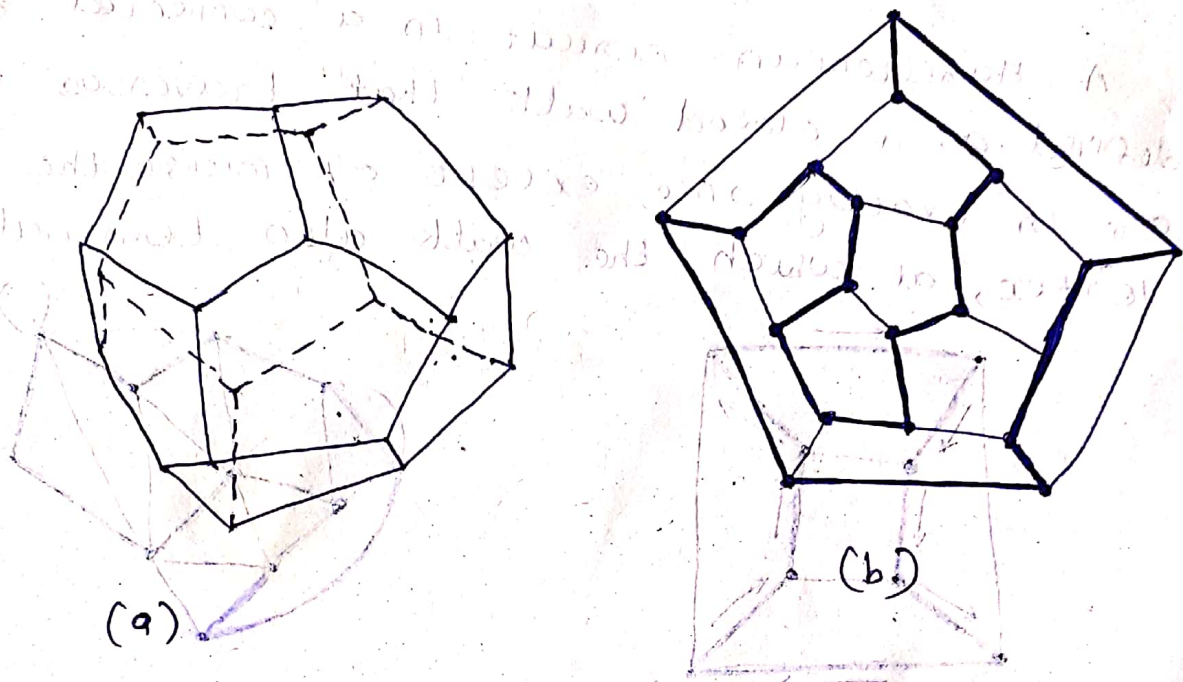


Fig. 2.8 Dodecahedron and its graph shown with a Hamiltonian.

* There is no known criterion we can apply to determine the existence of a Hamiltonian circuit in general.

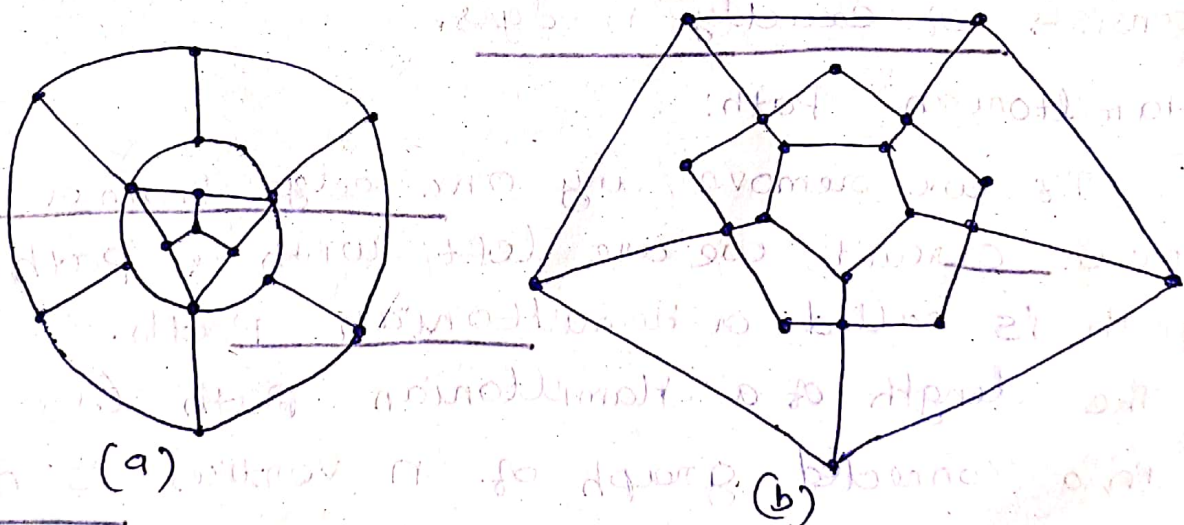


Fig. 2.9 Graphs without Hamiltonian circuits.

* What general class of graphs is guaranteed to have a Hamiltonian circuit?

Complete graphs of three or more vertices constitute one such class.

- Complete graph: (Universal graph or clique)

A simple graph in which there exist an edge between every pair of vertices is called a complete graph.

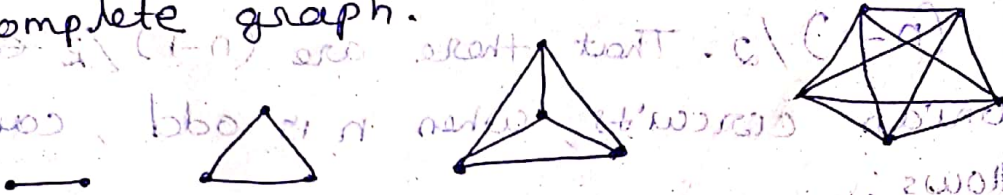


Fig. 2.10 Complete graphs of two, three, four and five vertices.

* Number of Hamiltonian Circuits in a Graph:

- A given graph may contain more than one Hamiltonian circuit.
- The determination of the exact number of edge-disjoint Hamiltonian circuits (or paths) in a graph in general is also an unsolved problem.

(11)

(15)

Theorem 2.3

In a complete graph with n vertices there are $(n-1)/2$ edge-disjoint Hamiltonian circuits, if n is an odd number ≥ 3 .

Proof: A complete graph G of n vertices has $n(n-1)/2$ edges, and a Hamiltonian circuit in G consists of n edges. Therefore, the number of edge-disjoint Hamiltonian circuits in G cannot exceed $(n-1)/2$. That there are $(n-1)/2$ edge-disjoint Hamiltonian circuits when n is odd, can be shown as follows:

The subgraph (of a complete graph of n vertices) in Fig 2.11 is a Hamiltonian circuit. Keeping the vertices fixed on a circle, rotate the polygonal pattern clockwise

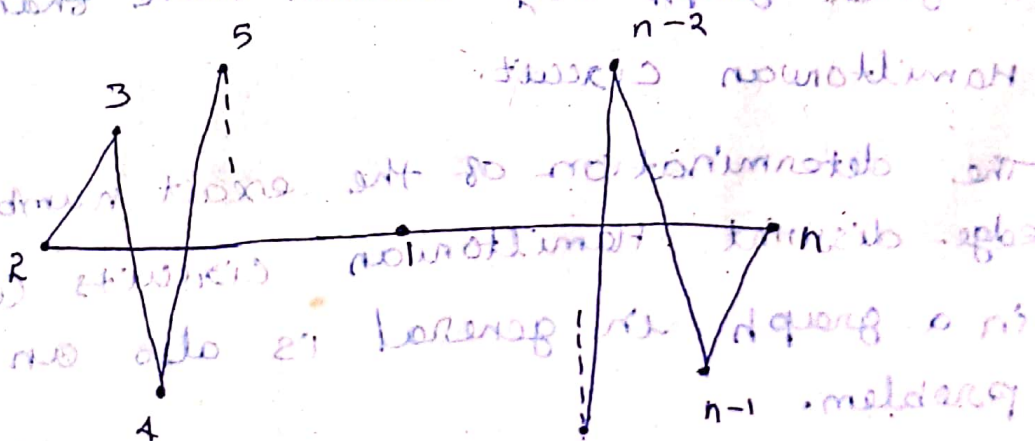


Fig. 2.11 Hamiltonian circuit; n is odd.

(12)

(11)

by $360/(n-1), 2 \cdot 360/(n-1), 3 \cdot 360/(n-1), \dots, (n-3)/2 \cdot 360/(n-1)$ degrees. Observe that each rotation produces a Hamiltonian circuit that has no edge in common with any of the previous ones. Thus we have $(n-3)/2$ new Hamiltonian circuits, all edge disjoint from the one in Fig. 2.11 and also edge disjoint among themselves. Hence the theorem.

Dirac's Theorem

Theorem 1.

Every graph G with $n \geq 3$ vertices and minimum degree $\delta(G) \geq n/2$ has a Hamilton cycle.

proof: suppose that $G = (V, E)$ satisfies the hypotheses of the theorem. Then G is connected, since otherwise the degree of any vertex in a smallest component C of G would be at most $|C| - 1 < n/2$, contradicting the hypothesis $\delta(G) \geq n/2$.

Let $P = x_0 x_1 \dots x_k$ be a longest path in G . Since P cannot be extended to a longer path, all the neighbours of x_0 and all the neighbours

of x_k lie on P . Hence, at least $n/2$ of the vertices x_0, \dots, x_{k-1} are adjacent to x_k , and at least $n/2$ of the vertices x_1, \dots, x_k are adjacent to x_0 . Another way of saying the second part of the last sentence is: at least $n/2$ of the vertices $x_i \in \{x_0, \dots, x_{k-1}\}$ are such that $x_0 x_{i+1} \in E$. Combining both statements and using the pigeonhole principle, we see that there is some x_i with $0 \leq i \leq k-1$, $x_i x_k \in E$ and $x_0 x_{i+1} \in E$.

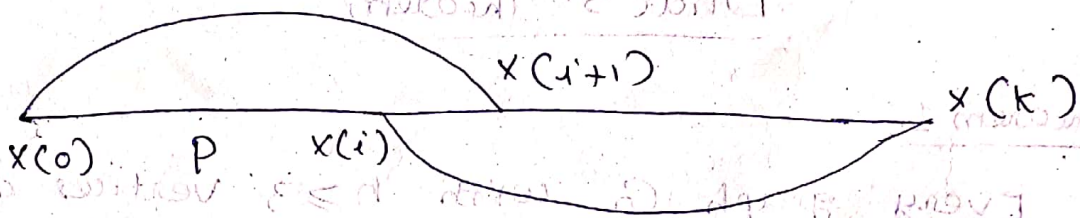


Fig. 2.12

we claim that the cycle

$$C = x_0 x_{i+1} x_{i+2} \dots x_{k-1} x_k x_i x_{i-1} \dots x_1 x_0 = x_0 x_{i+1} P x_k x_i P x_0$$

is a Hamilton cycle of G . Otherwise since G is connected, there would be some vertex x_j of C adjacent to a vertex y not in C , so that $e = x_j y \in E$. But then we could attach e to a path ending in x_j containing k edges of C , constructing a path in G longer than P .

TRAVELING - SALESMAN PROBLEM

- A salesman is required to visit a number of cities during a trip.

Given the distances between the cities, in what order should he travel so as to visit every city precisely once and return home, with the minimum mileage traveled?

vertices \rightarrow the cities

edges \rightarrow roads between them.

$w(e_i) \rightarrow$ weight of edge e_i , the distance

- If each of the cities has a road to every other city, we have a complete weighted graph.

- The total number of different (not edge disjoint) Hamiltonian circuits in a complete graph of n vertices can be shown to be $(n-1)!/2$.

from first vertex $n-1$ edges

from second vertex $n-2$ edges

from third vertex $n-3$ edges

we get $(n-1)!$ possible number of choices.

This number is divided by 2 because each Hamiltonian circuit has been counted twice.

* The problem of the traveling salesman can always be solved by enumerating all $(n-1)!/2$ Hamiltonian circuits.

* A travelling salesman wishes to visit several given cities and return to his starting point, covering the least possible total distance.

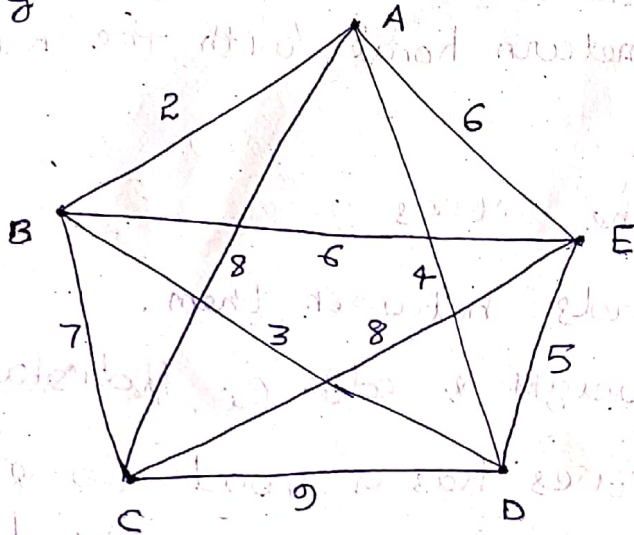


Fig. 2.13

The shortest possible route is $A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow A$, giving a total distance of 26.

* One possible algorithm is to calculate the total distance for all Hamiltonian cycles.

Ex:1 solve the travelling salesman problem for the weighted graph of Fig. 2.14

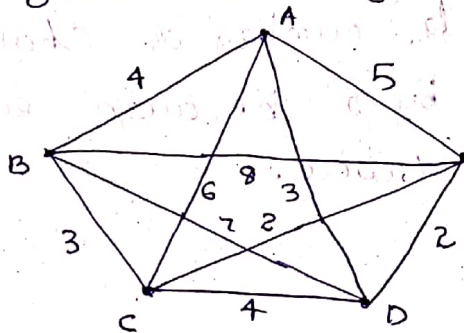


Fig. 2.14

(16)

Ex-2. Find the Hamiltonian cycle of greatest weight in the graph of Fig. 2.13

DIRECTED GRAPHS (Oriented graph).

- Graphs in which edges have directions.

Eg:

1. The street map of a city with one-way streets,
2. Flow networks with valves in the pipes and
3. Electrical networks.

4. Abstract representations of computer programs.

Vertices - program instructions

edges - the execution sequence.

* A directed graph (or a digraph for short) G consists of a set of vertices $V = \{v_1, v_2, \dots\}$, a set of edges $E = \{e_1, e_2, \dots\}$, and a mapping ψ that maps every edge onto some ordered pair of vertices (v_i, v_j) .

- edge is (arc)

- incident out of a vertex (initial vertex)

- incident into a vertex (terminal vertex)

- self-loop (initial & terminal are same)

* out-degree (or out-valence or outward demidegree)

The number of edges incident out of a vertex v_i is called the out-degree of v_i and is written $d^+(v_i)$.

- in-degree (or in-valence or inward demidegree)

The number of edges incident into v_i is called the in-degree of v_i and is written as $d^-(v_i)$.

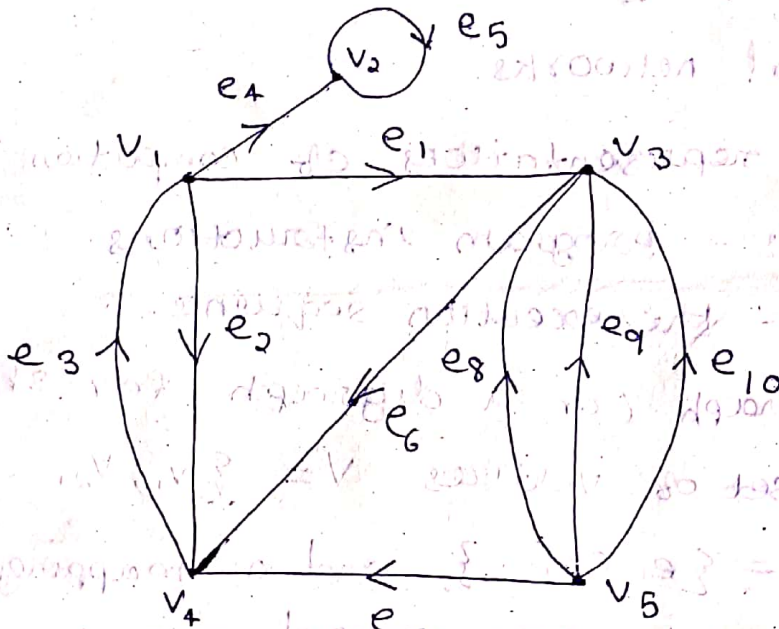


Fig. 2.15 Directed graph with 5 vertices and 10 edges.

$$d^+(v_1) = 3,$$

$$d^-(v_1) = 1,$$

$$d^+(v_2) = 1,$$

$$d^-(v_2) = 2,$$

$$d^+(v_3) = 4,$$

$$d^-(v_5) = 0.$$

* In Any digraph G the sum of all in-degrees is equal to the sum of all out-degrees, each sum being equal to the number of edges in G ; that is

$$\sum_{i=1}^n d^+(v_i) = \sum_{i=1}^n d^-(v_i)$$

* Isolated Vertex.

An Isolated vertex is a vertex in which the in-degree and the out-degree are both equal to zero.

* pendant vertices.

A vertex v in a digraph is called pendant

if it is of degree one, that is, if

$$d^+(v) + d^-(v) = 1.$$

* parallel edges.

Two directed edges are said to be parallel if they are mapped onto the same ordered pair of vertices.

eg: In Fig 2.15 e_8, e_9 & e_{10} are parallel.

whereas e_2 & e_3 are not.

* Undirected graph corresponding to G .

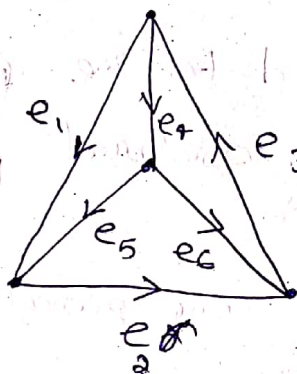
An undirected graph obtained from a directed graph G will be called the undirected graph corresponding to G .

* Digraph associated with H .

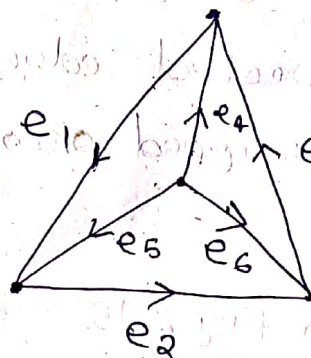
Given an undirected graph H , we can assign each edge of H some arbitrary direction. The resulting digraph, designated by \vec{H} , is called an orientation of H .

* Isomorphic Digraphs:

For two digraphs to be isomorphic not only must their corresponding undirected graphs be isomorphic, but the directions of the corresponding edges must also agree.



(a)



(b)

Fig 2.16 Two nonisomorphic digraphs.

Types of Graphs

1. Simple Digraphs
2. Asymmetric Digraphs
3. Symmetric Digraphs
4. Simple symmetric digraph
5. Complete Digraphs
6. Complete symmetric digraphs
7. Complete asymmetric digraph
8. Tournament or a complete tournament
9. Balanced Digraph or Pseudosymmetric, isograph.

Simple Digraphs:

A digraph that has no self-loop or parallel edges is called a simple digraph.

Asymmetric Digraphs:

Digraph that have at most one directed edge between a pair of vertices, but are allowed to have self-loops, are called asymmetric or anti-symmetric.

Symmetric Digraphs:

Digraphs in which for every edge (a, b) (i.e., from vertex a to b) there is also an edge (b, a) .

simple symmetric digraph

A digraph that is both simple and symmetric is called a simple symmetric digraph.

simple asymmetric digraph

A digraph that is both simple and asymmetric is simple asymmetric.

complete Digraphs

A complete undirected graph was defined as a simple graph in which every vertex is joined to every other vertex exactly by one edge.

- A complete symmetric digraph is a simple digraph in which there is exactly one edge directed from every vertex to every other vertex.
- A complete asymmetric digraph is an asymmetric digraph in which there is exactly one edge between every pair of vertices.
- A complete asymmetric digraph of n vertices contains $n(n-1)/2$ edges.
- A complete symmetric digraph of n vertices contains $n(n-1)$ edges.

Tournament

A complete asymmetric digraph is also called a tournament or a complete tournament.

Balanced Digraph (Pseudosymmetric)

A digraph is said to be balanced if for every vertex v_i , the in-degree equals the out-degree;

$$d^+(v_i) = d^-(v_i).$$

— regular graph

A balanced digraph is said to be regular if every vertex has the same in-degree and out-degree as every other vertex.

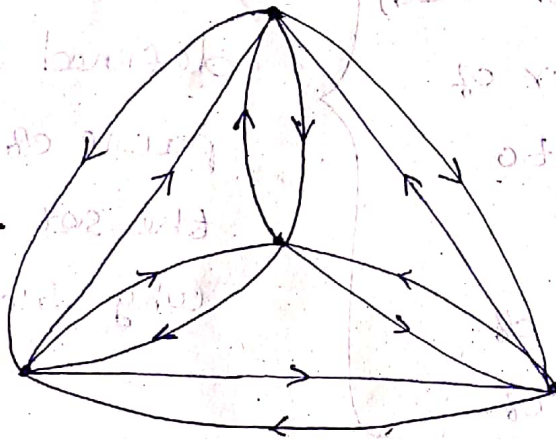


Fig 2.17 Complete symmetric digraph of four vertices.

DIGRAPHS AND BINARY RELATIONS

In a set of objects, X , where

$$X = \{x_1, x_2, \dots\},$$

a binary relation R between pairs (x_i, x_j) may exist.

$$x_i R x_j$$

and say that x_i has relation R to x_j .

Relation R

- is parallel to
- is orthogonal to
- is congruent to
- is greater than
- is a factor of
- is equal to
- is son of
- is spouse of
- is friend of

each of relations is defined only on pairs of numbers of the set, - that is why binary relation

* A digraph is the most natural way of representing a binary relation on a set X .

- Each $x_i \in X$ is represented by a vertex x_i .
- If x_i is related to x_j by the specified relation R , a directed edge is drawn from vertex x_i to x_j , for every pair (x_i, x_j) .

* Every binary relation on a finite set can be represented by a digraph without parallel edges.

* Every digraph without parallel edges defines a binary relation on the set of its vertices.

Types of Relations

1. Reflexive Relation
2. Symmetric Relation
3. Transitive Relation
4. Equivalence Relation

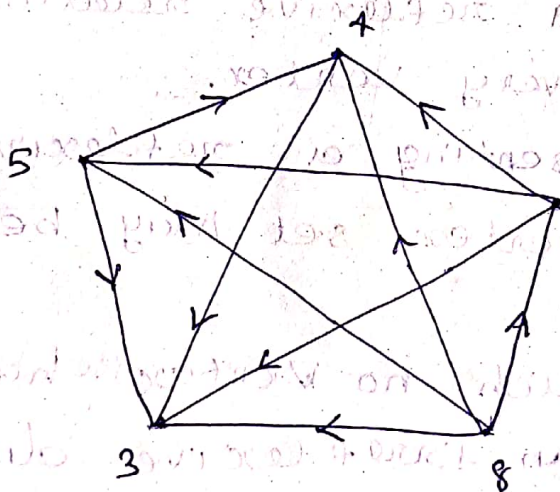


Fig. 2.18 Digraph of a binary relation.

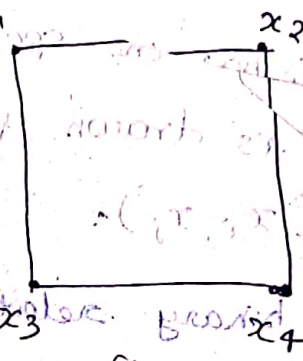
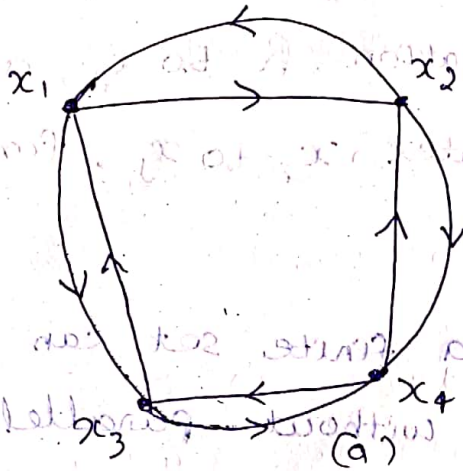


Fig. 2.19 Graphs of symmetric binary relation.

Reflexive Relation

eg: a number is always equal to itself.

A relation R on set X that satisfies

$$x_i R x_i$$

for every $x_i \in X$ is called a reflexive relation.

- The digraph of a reflexive relation will have a self-loop at every vertex.
- A digraph representing a reflexive binary relation on its vertex set may be called a reflexive digraph.
- A digraph in which no vertex has a self-loop is called an irreflexive digraph.

Symmetric Relation

For all x_i and x_j if

$x_i R x_j$ holds, then $x_j R x_i$ also holds.

Such a relation is called a symmetric relation.

eg: "Is spouse of" \Rightarrow symmetric but irreflexive

"Is equal to" \Rightarrow both symmetric and reflexive

- The digraph of a symmetric relation is a symmetric digraph.

- For every directed edge from vertex x_i to x_j , there is a directed edge from x_j to x_i .

- Every undirected graph with e edges can be thought of as a symmetric digraph with $2e$ directed edges.

eg: A two-way street is equivalent to two one-way streets pointed in opposite directions.

Transitive Relation

A relation R is said to be transitive if for

any three elements x_i , x_j , and x_k in the set,

$x_i R x_j$ and $x_j R x_k$

always imply

$x_i R x_k$.

(27)

(85)

eg: 1. Is greater than

$$x_i > x_j \text{ and } x_j > x_k$$

$$\text{clearly } x_i > x_k$$

2. Is descendent of

* A digraph representing a transitive relation (on its vertex set) is called a transitive directed graph.

Equivalence Relation:

A binary relation is called an equivalence relation if it is reflexive, symmetric and transitive.

eg: 1. Is parallel to

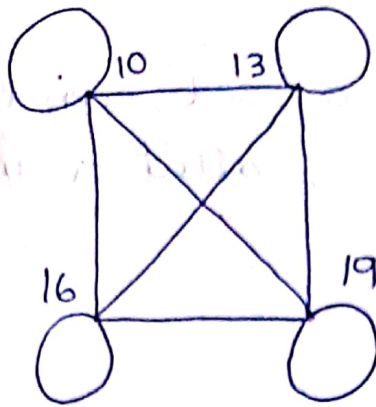
2. Is equal to

3. Is congruent to

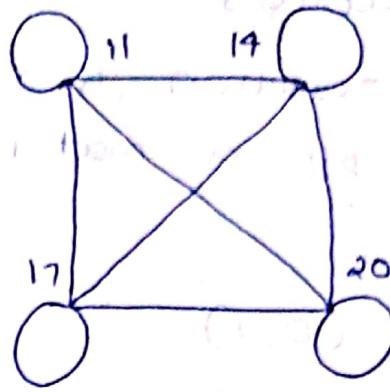
4. Is equal to modulo m

5. Is isomorphic to

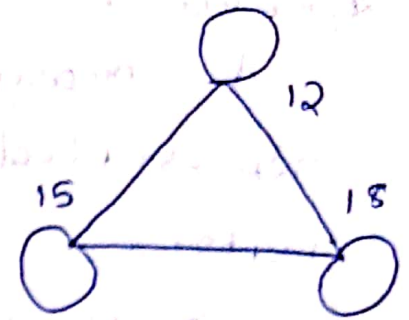
- The graph representing an equivalence relation may be called an equivalence graph.



$$\equiv 1 \pmod{3}$$



$$\equiv 2 \pmod{3}$$



$$\equiv 0 \pmod{3}$$

Fig 2.20 Equivalence graph.

1. An equivalence relation on a set partitions the elements of the set into classes
 - called equivalence classes.
 - two elements are in the same class if and only if they are related.
2. Symmetry ensures that there is no ambiguity regarding membership in the equivalence class.
 - otherwise, x_i may have been related to x_j but not vice versa.
3. Transitivity ensures that in each component every vertex is joined to every other vertex.
 - because if a is related to b and b is related to c , a is also related to c .
 - no element can be in more than one class.

4. Relation matrices.

A binary relation R on a set can also be represented by a matrix, called a relation matrix:

→ it is $(0, 1)$

→ n by n matrix,

n is the number of elements in the set.

Eg:

The relation matrix of the relation "greater than" on the set of integers

$\{3, 4, 7, 5, 8\}$ is

	3	4	7	5	8
3	0	0	0	0	0
4	1	0	0	0	0
7	1	1	0	0	0
5	1	1	1	0	0
8	1	1	1	1	0

Module III

3.1 TREES

- Application of graph
- A tree is a connected graph without any circuits.
- a graph must have at least one vertex.
- a tree has to be a simple graph.
- the term tree comes from family tree.

egs:

1. A river with its tributaries and subtributaries can be represented by a tree.
2. The sorting of mail according to zip code and
3. the sorting of punched cards are done according to a tree called
 - decision tree or sorting tree.



Fig. 3.1 Trees with one, two, three and four vertices.

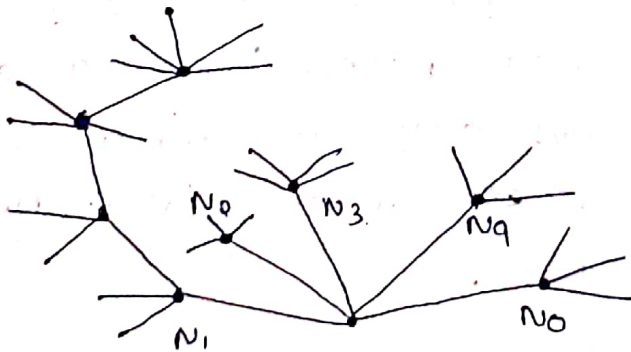


Fig. 3.2 Decision tree.

①

3.2. PROPERTIES OF TREES

* Theorem 3-1

There is one and only one path between every pair of vertices in a tree, T .

Proof:

Since T is a connected graph, there must exist at least one path between every pair of vertices in T . Now suppose that between two vertices a and b of T there are two distinct paths. The union of these two paths will contain a circuit and T cannot be a tree.

* Theorem 3-2

If in a graph G there is one and only one path between every pair of vertices, G is a tree.

Proof:

Existence of a path between every pair of vertices assures that G is connected. A circuit in a graph (with two or more vertices) implies that there is at least one pair of vertices a, b such that there are two distinct paths between a and b . Since G has one and only one path between every pair of vertices, G can have no circuit. Therefore, G is a tree.

* Theorem 3.3

A tree with n vertices has $n-1$ edges.

Proof:

The theorem will be proved by induction on the number of vertices. It's easy to see that the theorem is true for $n=1, 2$, and 3 . Assume that the theorem holds for all trees with fewer than n vertices.

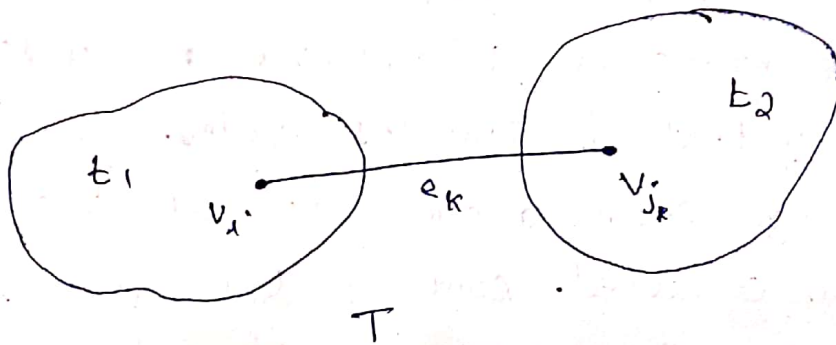


Fig. 3-3 Tree T with n vertices.

Let us now consider a tree T with n vertices. In T let e_k be an edge with end vertices v_i and v_j . According to theorem "There is one and only one path between every pair of vertices in a tree, T ," there is no other path between v_i and v_j except e_k . Therefore, deletion of e_k from T will disconnect the graph, as shown in Fig. 3-3. Furthermore, $T - e_k$ consists of exactly two components, and since there were no circuits in T to begin with, each of these components is a tree. Both these trees, t_1 and t_2 , have fewer than n vertices each, and therefore, by the induction hypothesis, each contains one less edge than the number of vertices in it. Thus

(3)

$T-e_k$ consists of $n-2$ edges (and n vertices). Hence T has exactly $n-1$ edges.

* Theorem 3-4

Any connected graph with n vertices and $n-1$ edges is a tree.

Proof:

Let G be a connected graph with n vertices and $n-1$ edges. We show that G contains no cycles. Assume to the contrary that G contains cycles.

Remove an edge from a cycle so that the resulting graph is again connected. Continue this process of removing one edge from one cycle at a time till the resulting graph H is a tree. As H has n vertices, so number of edges in H is $n-1$. Now, the number of edges in G is greater than the number of edges in H . So $n-1 > n-1$, which is not possible. Hence G has no cycles and therefore is a tree.

- Definition: minimally connected

A graph is said to be minimally connected if removal of any one edge from it disconnects the graph. Clearly, a minimally connected graph has no cycles.

* Theorem 3.5

A graph is a tree if and only if it is minimally connected.

Proof:

Let the graph G be minimally connected. Then G has no cycles and therefore is a tree.

conversely, let G be a tree. Then G contains no cycles and deletion of any edge from G disconnects the graph. Hence G is minimally connected.

* Theorem 3-6

A graph G with n vertices, $n-1$ edges, and no circuits is connected.

Proof:

Suppose there exists a circuitless graph G with n vertices and $n-1$ edges which is disconnected. In that case G will consist of two or more circuitless components. Without loss of generality, let G consist of two components, g_1 and g_2 . Add an edge e between a vertex v_1 in g_1 and v_2 in g_2 . (Fig. 3-4).

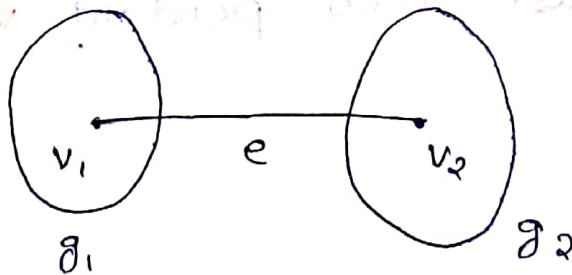


Fig. 3-4 Edge e added to $G = g_1 \cup g_2$.

Since there was no path between v_1 and v_2 in G , adding e did not create a circuit. Thus $G \cup e$ is a circuitless, connected graph (i.e., a tree) of n vertices and n edges, which is not possible, because of theorem "A tree with n vertices has $n-1$ edges".

- * A graph G with n vertices is called a tree if
1. G is connected and is circuitless, or
 2. G is connected and has $n-1$ edges, or
 3. G is circuitless and has $n-1$ edges, or
 4. There is exactly one path between every pair of vertices in G , or
 5. G is a minimally connected graph.

3-3

PENDANT VERTICES IN A TREE

- In a tree of n vertices we have $n-1$ edges
 - $2(n-1)$ degrees to be divided among n vertices.
 - no vertex can be of zero degree.
 - at least two vertices of degree one in a tree.
- * In any tree (with two or more vertices) there are at least two pendant vertices.

An Application

Given a sequence of integers, no two of which are the same, find the largest monotonically increasing subsequence in it.

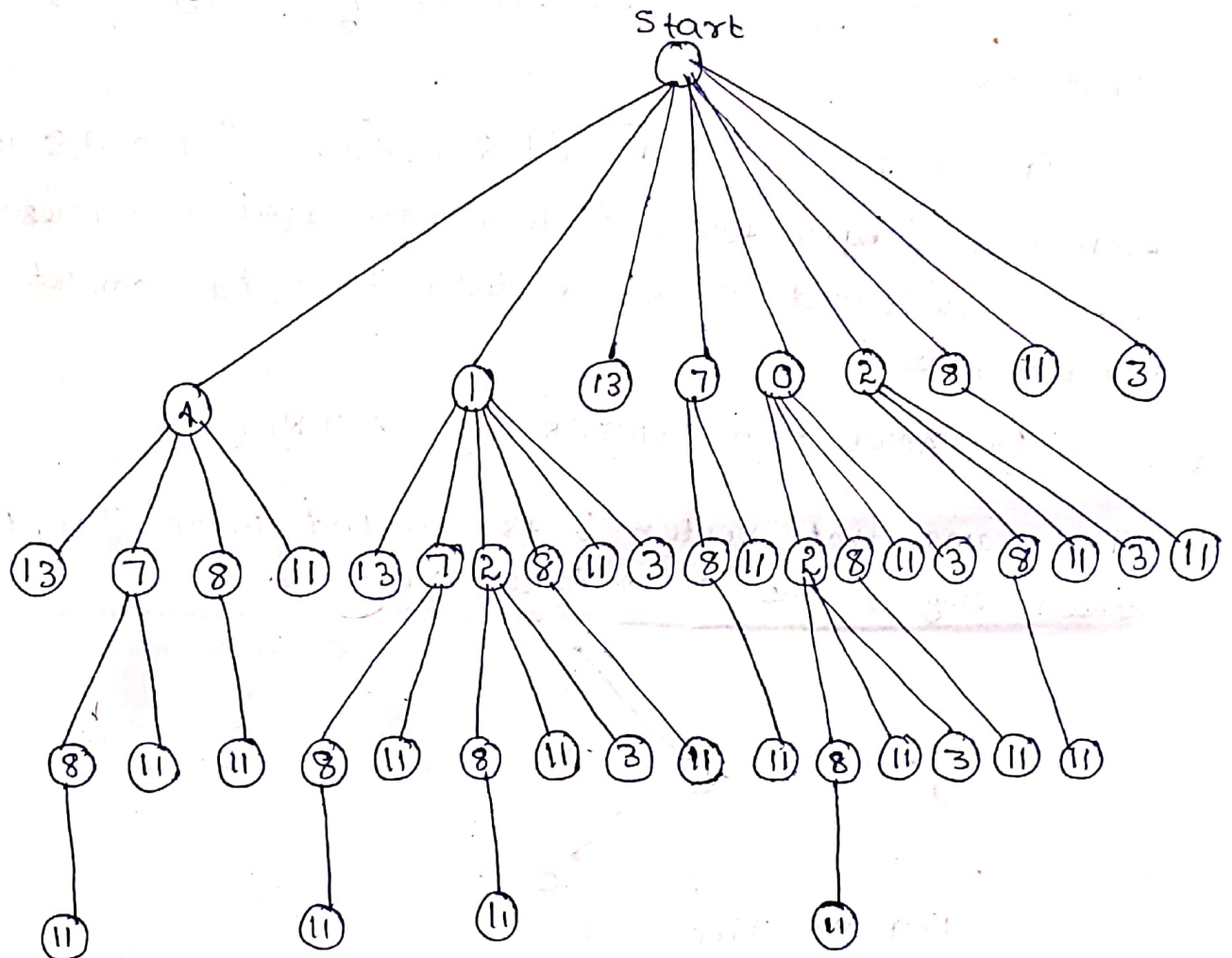


Fig. 3-5 Tree of the monotonically increasing sequences in 4, 1, 13, 7, 0, 2, 8, 11, 3.

suppose that the sequence given to us is

1, 1, 13, 7, 0, 2, 8, 11, 3

it can be represented by a tree in which the vertices (except the start vertex) represent

individual numbers in the sequence, and the path from the start vertex to a particular vertex v describes the monotonically increasing subsequence terminating in v . As shown in Fig. 3-5, this sequence contains four longest monotonically increasing subsequences that is,

$(4, 7, 8, 11)$, $(1, 7, 8, 11)$, $(1, 2, 8, 11)$, and $(0, 2, 8, 11)$.

Each is of length four. Such a tree used in representing data is referred to as a data tree by computer programmers.

3.4. DISTANCE AND CENTERS IN A TREE

It seems that vertex b is located more "centrally" than any of the other three vertices.

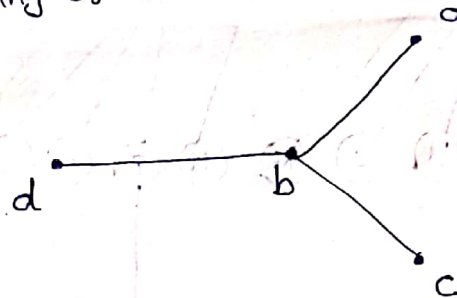


Fig. 3.6 Tree.

distance $d(v_i, v_j)$

In a connected graph G , the distance $d(v_i, v_j)$ between two of its vertices v_i and v_j is the length of the shortest path (i.e., the number of edges in the shortest path) between them.

Metric

$f(x, y)$ — A function of two variables, a "distance" between them.

1. Nonnegativity : $f(x, y) \geq 0$, and $f(x, y) = 0$ if and only if $x = y$.
2. Symmetry : $f(x, y) = f(y, x)$.
3. Triangle inequality : $f(x, y) \leq f(x, z) + f(z, y)$
for any z .

A function that satisfies these three conditions is called a metric.

Theorem 8

The distance between vertices of a connected graph is a metric.

Proof:

A function that satisfies the following three conditions is called a metric.

1. Nonnegativity : $f(x, y) \geq 0$, and $f(x, y) = 0$
if and only if $x = y$.
2. Symmetry : $f(x, y) = f(y, x)$.
3. Triangle inequality : $f(x, y) \leq f(x, z) + f(z, y)$
for any z .

That the distance $d(v_i, v_j)$ between two vertices

of a connected graph satisfies conditions 1 and 2 is immediately evident. Since $d(v_i, v_j)$ is the length of the shortest path between vertices v_i and v_j , this path cannot be longer than another path between v_i and v_j , which goes through a specified vertex v_k . Hence

$$d(v_i, v_j) \leq d(v_i, v_k) + d(v_k, v_j).$$

Eccentricity (Associated number or separation)

The eccentricity $E(v)$ of a vertex v in a graph G is the distance from v to the vertex farthest from v in G ; that is

$$E(v) = \max_{v_i \in G} d(v, v_i).$$

Center of G

A vertex with minimum eccentricity in a graph G is called a center of G .

The eccentricities of the four vertices in Fig 3.6 are $E(a)=2$, $E(b)=1$, $E(c)=2$ and $E(d)=2$. Hence vertex b is the center of that tree.

THEOREM 9

Every tree has either one or two centers.

Proof:

The maximum distance, $\max d(v, v_i)$, from a

given vertex v to any other vertex v_i occurs only when v_i is a pendant vertex. With this observation, let us start with a tree T having more than two vertices. Tree T must have two or more pendant vertices.

Delete all the pendant vertices from T . The resulting graph T' is still a tree. What about the eccentricities of the vertices in T' ? A little deliberation will reveal that removal of all pendant vertices from T uniformly reduced the eccentricities of the remaining vertices (i.e., vertices in T') by one. Therefore, all vertices that T had as centers will still remain centers in T' . From T' we can again remove all pendant vertices and get another tree T'' . We continue this process until there is left either a vertex (which is the center of T) or an edge (whose end vertices are

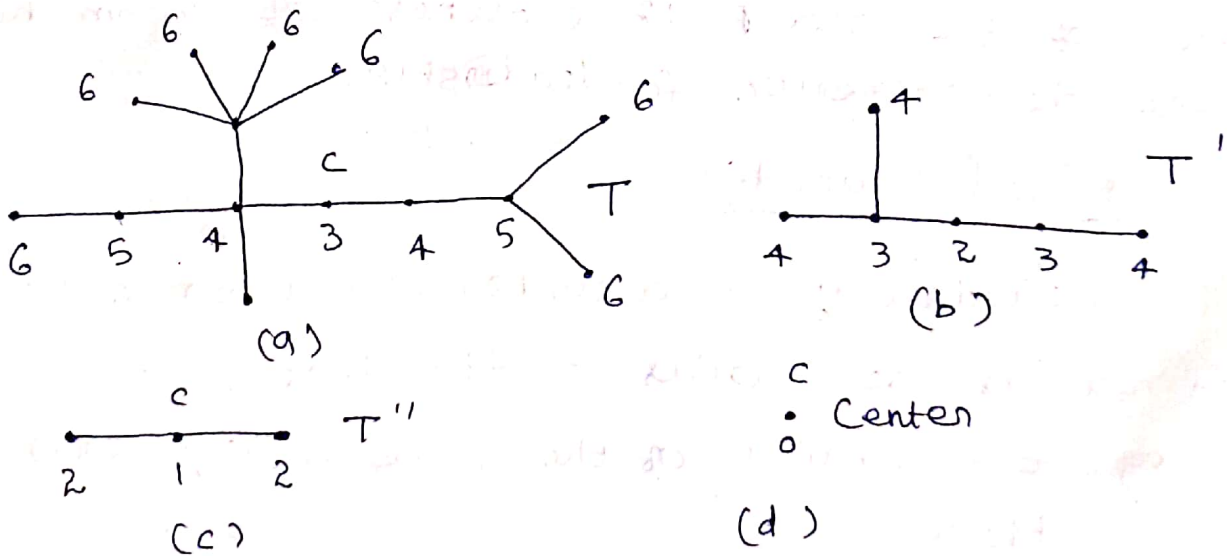


Fig. 3-7 Finding a center of a tree.

the two centers of T). Thus the theorem.

Application of eccentricity and center.

Suppose that the communication among a group of 14 persons in a society is represented by the graph in Fig. 3-7(a), where the vertices represent the persons and an edge represents the communication link between its two end vertices.

Since the graph is connected, we know that all the members can be reached by any member, either directly or through some other members. But it is also important to note that the graph is a tree - minimally connected. The group cannot afford to ~~lose~~ lose any of the communication links.

The eccentricity of each vertex, $E(v)$, represents how close v is to the farthest member of the group. In Fig. 3-7(a), vertex c should be the leader of the group, if closeness of communication were the criterion for leadership.

Radius and Diameter

The eccentricity of a center in a tree is defined as the radius of the tree.

eg: the radius of the tree in Fig. 3-7(a) is three.

The diameter of a tree T , on the other hand, is defined as the length of the longest path in T .

3.5 ROOTED AND BINARY TREES

Rooted tree

A tree in which one vertex (called the root) is distinguished from all the others is called a rooted tree.

In a diagram of a rooted tree, the root is generally marked distinctly.

— the root is enclosed in a small triangle.

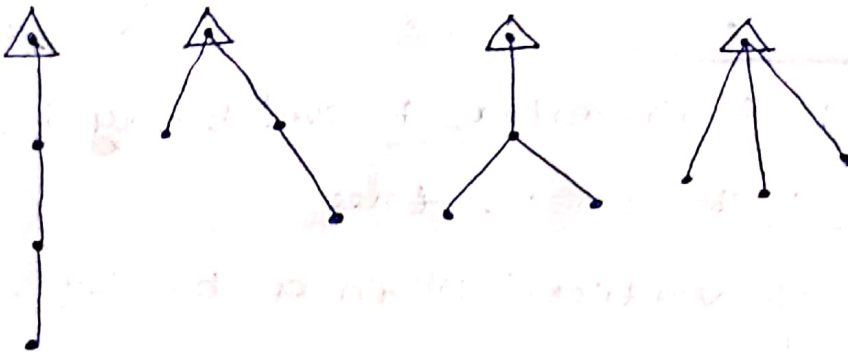


Fig. 3-8 Rooted trees with four vertices.

Binary tree

1. A special class of rooted trees,
2. Extensively used in the study of computer search methods, binary identification problems, and variable-length binary codes.

A binary tree is defined as a tree in which there is exactly one vertex of degree two, and each of the remaining vertices is of degree one or three.

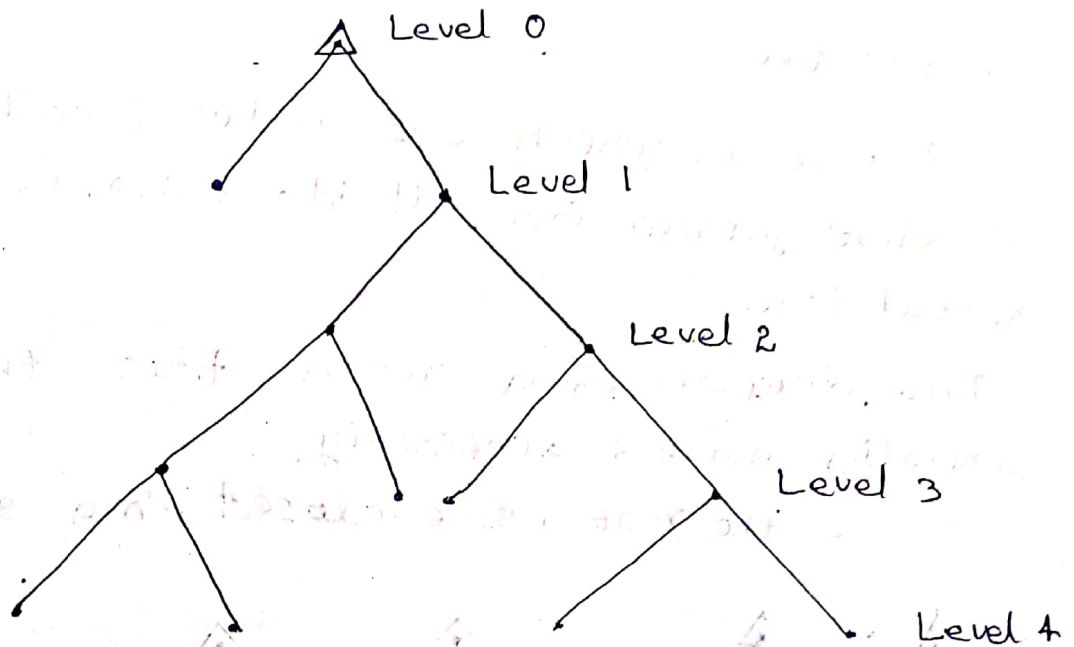


Fig. 3-9 A 13-vertex, 4-level binary tree.

Two properties of binary tree

1. The number of vertices n in a binary tree is always odd.

This is because there is exactly one vertex of even degree, and the remaining $n-1$ vertices are of odd degrees. Since the number of vertices of odd degrees is even, $n-1$ is even. Hence n is odd.

2. Let p be the number of pendant vertices in a binary tree T . Then $n-p-1$ is the number of vertices of degree three.

Therefore, the number of edges in T equals

$$\frac{1}{2} [P + 3(n - P - 1) + 2] = n - 1;$$

hence

$$P = \frac{n+1}{2}$$

-(1)

Internal vertex.

A nonpendant vertex in a tree is called an internal vertex.

The number of internal vertices in a binary tree is one less than the number of pendant vertices.

Level

In a binary tree a vertex v_i is said to be at level l_i if v_i is at a distance of l_i from the root. Thus the root is at level 0.

Application of binary trees

1. Search procedures.

Each vertex of a binary tree represents a test with two possible outcomes.

We start at the root, and the outcome of the test at the root sends us to one of the two vertices at the next level, where further tests are made,

and so on. 1

Reaching a specified pendant vertex (the goal of the search) terminates the search.

For a given number of vertices n ,

1. There can be only one vertex (the root) at level 0,
2. At most two vertices at level 1,
3. At most four vertices at level 2, and so on.

Therefore,

the maximum number of vertices possible in a k -level binary tree is

$$2^0 + 2^1 + 2^2 + \dots + 2^k \geq n.$$

Height of the tree.

The maximum level, l_{\max} , of any vertex in a binary tree is called the height of the tree.

The minimum possible height of an n -vertex binary tree is

$$\min l_{\max} = \lceil \log_2(n+1) - 1 \rceil, \quad (2)$$

where

$\lceil n \rceil$ denotes the smallest integer greater than or equal to n .

The maximum possible height of an n -vertex binary tree must have exactly two vertices at each level, except at the 0 level.

Therefore,

$$\max l_{\max} = \frac{n-1}{2}.$$

— (3)

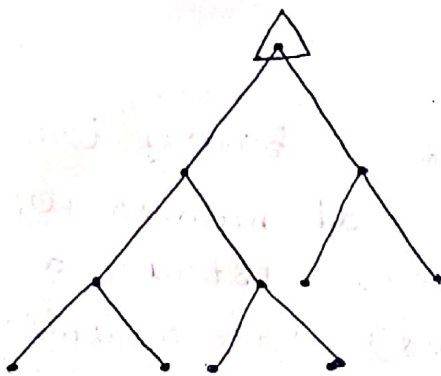
Level

0

1

2

3



Level

0

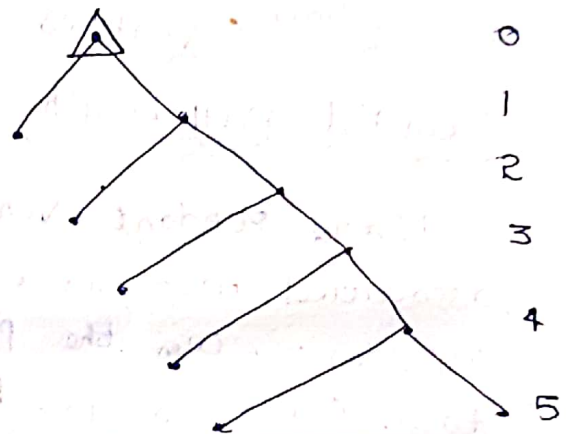
1

2

3

4

5



$$\min l_{\max} = \lceil (\log_2 12) - 1 \rceil$$

(a)

$$\max l_{\max} = \frac{11-1}{2} = 5$$

(b)

Fig. 3-10 Two 11-vertex binary trees.

The path length (external path length)

The sum of the levels of all pendant vertices. This quantity, known as the path length of a tree. The sum of the path lengths from the root to all pendant vertices.

For eg:

The path length of the binary tree in Fig. 3-9 is

$$1+3+3+4+4+4+4=23.$$

The path lengths of trees in Figs. 3-10(a) and (b) are 16 and 20, respectively.

- The path length is often directly related to the execution time of an algorithm.

- The minimum path length for a given n is

$$2^{l_{\max}-1} \text{ vertices at level } l_{\max}-1$$

Weighted Path Length

Every pendant vertex v_j of a binary tree has associated with it a positive real number w_j . Given w_1, w_2, \dots, w_m the problem is to construct a binary tree (with m pendant vertices) that minimizes

$$\sum w_j \cdot l_j,$$

where

l_j is the level of pendant vertex v_j , and the sum is taken over all pendant vertices.

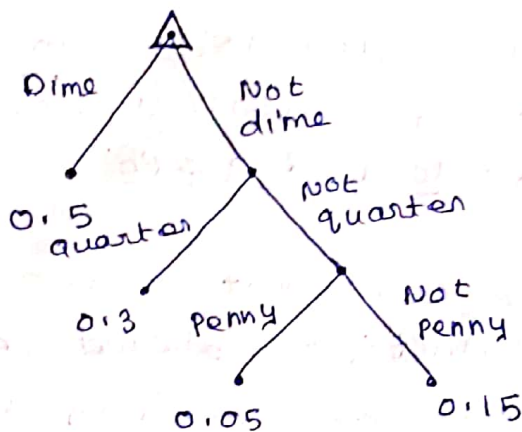
Eg: 1.

A coke machine is to identify, by a sequence of tests, the coin that is put into the machine. only pennies, nickels, dimes and quarters can go through the slot.

coin	probability
a Penny	0.05
a nickel	0.15
a dime	0.5
a quarter	0.3

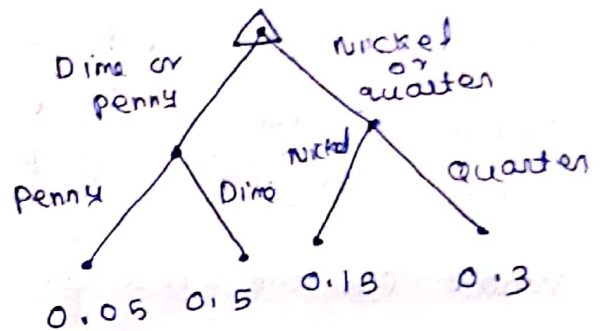
Each test has the effect of partitioning the four types of coins into two complementary sets and asserting the unknown coin to be in one of the two sets. If the time taken for each test is the same, what sequence of tests will minimize the expected time taken by the Coke machine to identify the coin?

Ans:



$$\sum w_i \cdot l_i = 1.7$$

(a)



$$\sum w_i \cdot l_i = 2$$

(b)

Fig. 3-11 Two binary trees with weighted pendant vertices

The expected time is $1.7t$, where t is the time taken for each test.

Fig:2

Variable-length binary codes

Binary trees with minimum weighted path length have also been used in constructing variable-length binary codes, where the letters of the alphabet

(A, B, C, \dots, Z) are represented by binary digits. Since different letters have different frequencies of occurrence (frequencies are interpreted as weights w_1, w_2, \dots, w_{26}), a binary tree with minimum weighted path length corresponds to a binary code of minimum cost.

ON COUNTING TREES

A graph in which each vertex is assigned a unique name or label (i.e. no two vertices have the same label) is called a labeled graph.

What is the number of different trees that one can construct with n distinct (or labeled) vertices? If $n=4$ we have 16 trees as shown in Fig. 3-12.

Theorem 3-10

There are n^{n-2} labeled trees with n vertices ($n \geq 2$).

proof:

Let the n vertices of a tree T be labeled $1, 2, 3, \dots, n$. Remove the pendant vertex (and the edge incident on it) having the smallest label, which is, say, a_1 . Suppose that b_1 was the vertex adjacent to a_1 . Among the remaining $n-1$ vertices let a_2 be the pendant vertex with the smallest label, and b_2 be the vertex adjacent to a_2 . Remove the edge (a_2, b_2) . This operation is repeated the remaining $n-2$ vertices, and then, on $n-3$ vertices, and so on. The process is terminated after $n-2$ steps, when only two

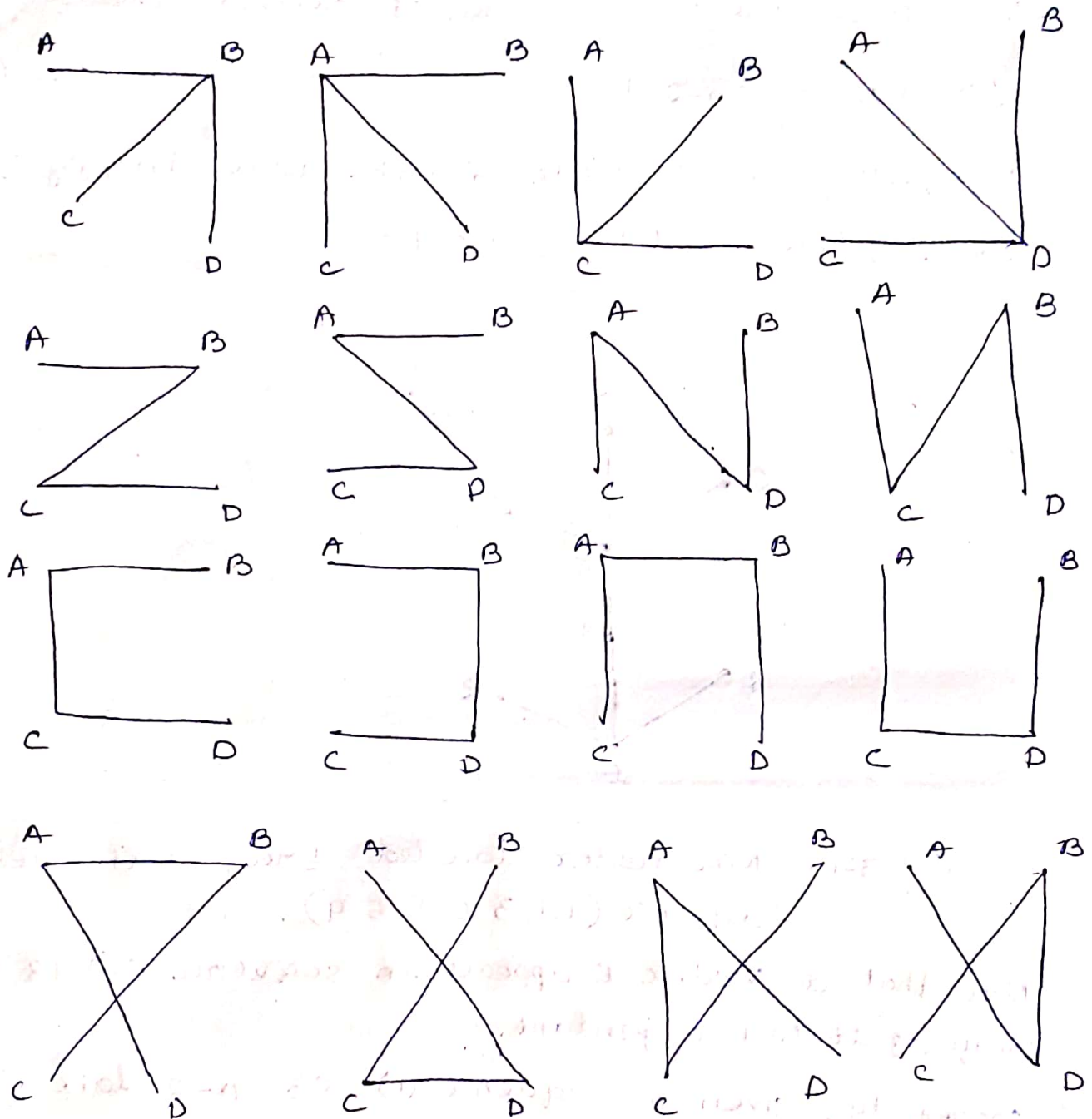


Fig. 3-12 All 16 trees of four labeled vertices.

Vertices are left. The tree T defines the sequence $(b_1, b_2, \dots, b_{n-2})$ — (1)

uniquely. For example for the tree in Fig. 3-13 the sequence is $(1, 1, 3, 5, 5, 5, 9)$.

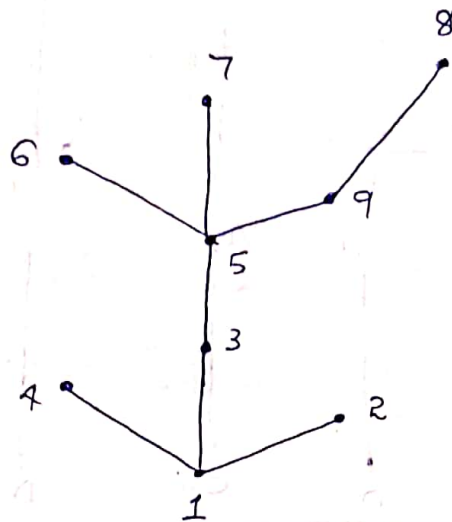


Fig. 3-13 Nine-vertex labeled tree, which yields sequence $(1, 1, 3, 5, 5, 5, 9)$.

Note that a vertex i appears in sequence (1) if and only if it is not pendant. In

conversely, given a sequence (1) of $n-2$ labels, an n -vertex tree can be constructed uniquely, as follows: Determine the first number in the sequence

$$1, 2, 3, \dots, n$$

(2)

that does not appear in sequence (1). This number clearly is a_1 . And thus the edge (a_1, b_1) is defined. Remove b_1 from sequence (1) and a_1 from (2). In the remaining

sequence of (2) find the first number that does not appear in the remainder of (1). This would be a_2 , and thus the edge (a_2, b_2) is defined. The construction is continued till the sequence (1) has no element left. Finally, the last two vertices remaining in (2) are joined. For example, given a sequence

$$(4, 4, 3, 1, 1),$$

we can construct a seven-vertex tree as follows:

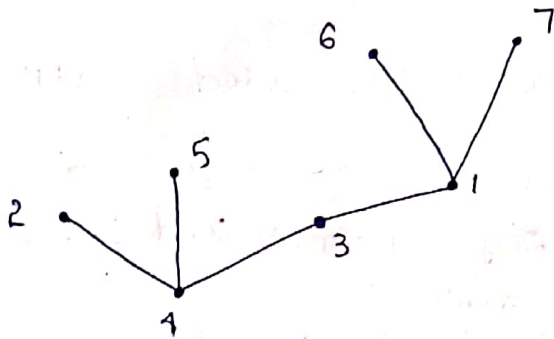


Fig. 3-14 Tree constructed from sequence $(4, 4, 3, 1, 1)$.

$(2, 4)$ is the first edge. The second is $(5, 4)$. Next, $(4, 3)$. Then $(3, 1)$, $(6, 1)$, and finally $(7, 1)$ as shown in Fig. 3-14

For each of the $n-2$ elements in sequence (1) we can choose any one of n numbers, thus forming

$$n^{n-2} \quad (3)$$

$(n-2)$ -tuples, each defining a distinct labeled tree of n vertices. And since each tree defines one of these sequences uniquely, there is a one-to-one correspondence between

the trees and the n^{n-2} sequences. Hence the theorem.

Unlabeled Trees

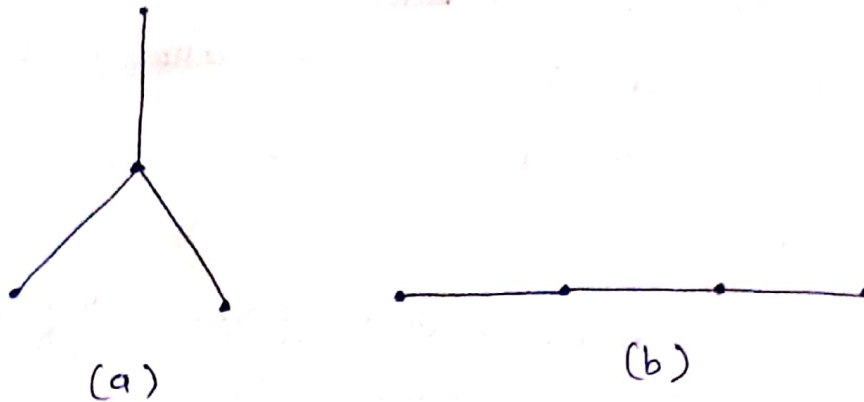


Fig. 3-15 All trees of four unlabeled vertices.

The four trees in the first row of Fig. 3-14 are isomorphic to the one in Fig. 3-15(a); and the other 12 are isomorphic to Fig. 3-15(b).

SPANNING TREES

A given graph has numerous subgraphs - from e edges, 2^e distinct combinations are possible. Some of these subgraphs will be trees.

A tree T is said to be a spanning tree of a connected graph G if T is a subgraph of G and T contains all vertices of G .

For instance, the subgraph in heavy lines in Fig. 3-16 is a spanning tree of the graph shown.

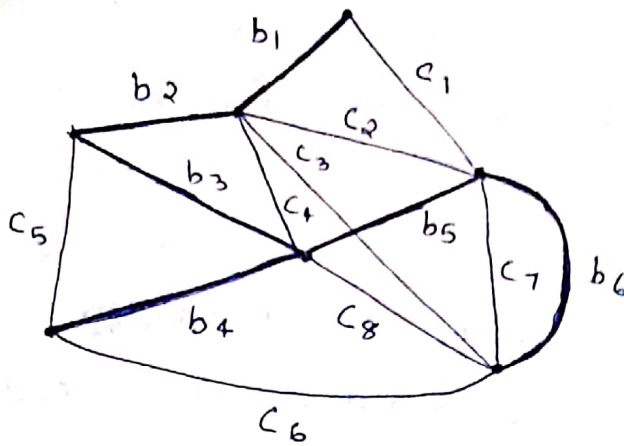


Fig. 3-16 spanning tree.

Since spanning trees are the largest (with maximum number of edges) trees among all trees in G , it is also quite appropriate to call a spanning tree a maximal tree subgraph or maximal tree of G .

Finding a spanning tree of a 'connected' graph G is simple.

If G has no circuit, it is its own spanning tree.

If G has a circuit, delete an edge from the circuit.

If there are more circuits, repeat the operation till an edge from the last circuit is deleted - leaving a connected, circuit-free graph that contains all the vertices of G .

Theorem 3-11

Every connected graph has at least one spanning tree.

Branch

An edge in a spanning tree T is called a branch of T .

chord

An edge of G that is not in a given spanning tree T , is called a chord.

In Fig. 3-16

edges b_1, b_2, b_3, b_4, b_5 and b_6 are branches

edges $c_1, c_2, c_3, c_4, c_5, c_6, c_7$ and c_8 are chords.

An edge that is a branch of one spanning tree T_1 (in a graph G) may be a chord with respect to another spanning tree T_2 .

$$T \cup \bar{T} = G$$

T : spanning tree

\bar{T} : complement of T in G .

Theorem 3-12

With respect to any of its spanning trees, a connected graph of n vertices and e edges has $n-1$ tree branches and $e-n+1$ chords.

Ex: If we have a farm consisting of six walled plots of land as shown in Fig. 3-17, and these plots are full of water, how many walls will have to be broken so that all the water can be drained out?

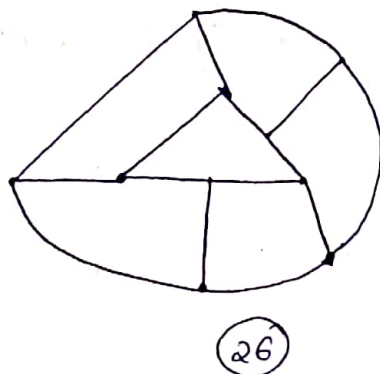


Fig. 3-17 Farm with walled plots of land.

Here $n=10$ and $e=15$.

We shall have to select a set of size $(15-10+1=6)$ walls such that the remaining nine constitute a spanning tree.

Breaking these six walls will drain the water out.

Rank and Nullity.

$$\text{Rank } r = n - k,$$

where n : no. of vertices in G

k : " components " "

$$\text{nullity } u = e - n + k,$$

The rank of a connected graph is $n-1$, and the nullity, $e - n + 1$.

rank of G = number of branches in any spanning tree (or forest) of G ,

nullity of G = number of chords in G ,

rank + nullity = number of edges in G .

The nullity of a graph is also referred to as its cyclomatic number, or first Betti number.

FINDING ALL SPANNING TREES OF A GRAPH

In a given connected graph there are a large number of spanning trees.

Cyclic interchange or elementary tree transformation

Generation of one spanning tree from another through addition of a chord and deletion of

an appropriate branch.

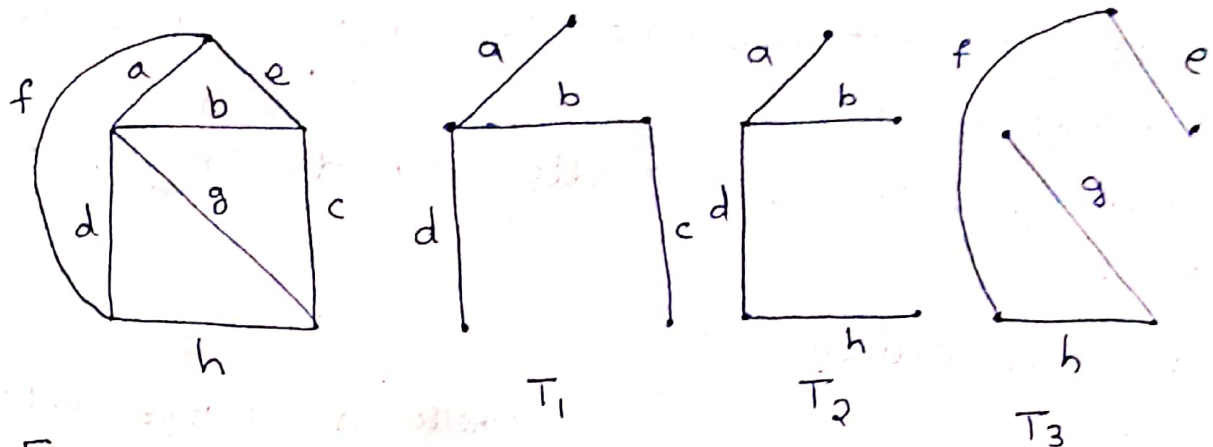


Fig. 3-18 Graph and three of its spanning trees.

The distance between two spanning trees ($d(T_i, T_j)$)

The number of edges of G present in one tree but not in the other. In Fig. 3-18 $d(T_2, T_3) = 3$.

The ring sum of two spanning trees. ($T_i \oplus T_j$)

The subgraph of G containing all edges of G that are either in T_i or in T_j but not in both.

$$d(T_i, T_j) = \frac{1}{2} N(T_i \oplus T_j).$$

The number $d(T_i, T_j)$ is the minimum number of cyclic interchanges involved in going from T_i to T_j .

Theorem 3-14.

The distance between the spanning trees of a graph is a metric. That is, it satisfies

$$d(T_i, T_j) \geq 0 \text{ and } d(T_i, T_j) = 0 \text{ if and only if } T_i = T_j,$$

$$d(T_i, T_j) = d(T_j, T_i),$$

$$d(T_i, T_j) \leq d(T_i, T_k) + d(T_k, T_j).$$

Theorem 3-15

Starting from any spanning tree of a graph G , we can obtain every spanning tree of G by successive cyclic exchanges.

The maximum distance between any two spanning trees in G is

$$\max d(T_i, T_j) = \frac{1}{2} \max n(T_i \oplus T_j)$$

$\leq r$, the rank of G .

μ is the nullity of G , no more than μ edges of a spanning tree T_i can be replaced to get another tree T_j .

$$\max d(T_i, T_j) \leq \mu;$$

Combining the two;

$$\max d(T_i, T_j) \leq \min(\mu, r)$$

central Tree

T_0 is called a central tree of G if

$$\max d(T_0, T_i) \leq \max_j d(T, T_j)$$

for every tree T of G .

SPANNING TREES IN A WEIGHTED GRAPH.

If graph G is weighted graph (i.e., if there is a real number associated with each edge of G), then the weight of a spanning tree T of G is defined as the sum of the weights of all the branches in T .

Theorem 3-16

A spanning tree T (of a given weighted connected graph G) is a shortest spanning tree (of G) if and only if there exists no other spanning tree (of G) at a distance of one from T whose weight is smaller than that of T .

Proof:

The necessary or the "only if" condition is obvious; otherwise, we shall get another tree shorter than T by a cyclic interchange. The fact that this condition is also sufficient is remarkable and is not obvious. It can be proved as follows:

Let T_1 be a spanning tree in G satisfying the hypothesis of the theorem (i.e., there is no spanning tree at a distance of one from T_1 which is shorter than T_1).

The proof will be completed by showing that if T_2 is a shortest spanning tree (different from T_1)

in G , the weight of T_1 will also be equal to that of T_2 . Let T_2 be a shortest spanning tree in G . Clearly, T_2 must also satisfy the hypothesis of the theorem (otherwise there will be a spanning tree shorter than T_2 at a distance of one from T_2 , violating the assumption that T_2 is shortest).

Consider an edge e in T_2 which is not in T_1 . Adding e to T_1 forms a fundamental circuit with branches in T_1 . Some but not all of the branches in T_1 that form the fundamental circuit with e may also be in T_2 ; each of these branches in T_1 has a weight smaller than or equal to that of e , because of the assumption on T_1 . Amongst all those edges in this circuit which are not in T_2 at least one, say b_j , must form some fundamental circuit (with respect to T_2) containing e . Because of the minimality assumption on T_2 weight of b_j cannot be less than that of e . Therefore b_j must have the same weight as e . Hence the spanning tree $T_1' = (T_1 \cup e - b_j)$, obtained from T_1 through one cycle exchange, has the same weight as T_1 . But T_1 has one edge more in common with T_2 , and it satisfies the condition of Theorem 3-16. This argument can be repeated, producing a series of trees of equal weight,

T_1, T_2, T_3, \dots , each a unit distance closer to T_2 , until we get T_2 itself.

This proves that if none of the spanning trees at a unit distance from T is shorter than T , no spanning tree shorter than T exists in the graph.

Algorithm for Shortest Spanning Tree:

Finding a shortest spanning tree in a given graph.

1) Kruskal's Algorithm.

1. List all edges of the graph G in order of nondecreasing ~~tree~~ weight.
2. Select a smallest edge of G .
For each successive step select (from all remaining edges of G) another smallest edge that makes no circuit with the previously selected edges.
3. Continue step 2 until $n-1$ edges have been selected, and these edges will constitute the desired shortest spanning tree.

2) Prim's algorithm

1. Draw n isolated vertices and label them v_1, v_2, \dots, v_n .
2. Tabulate the given weights of the edges of G in an n by n table.

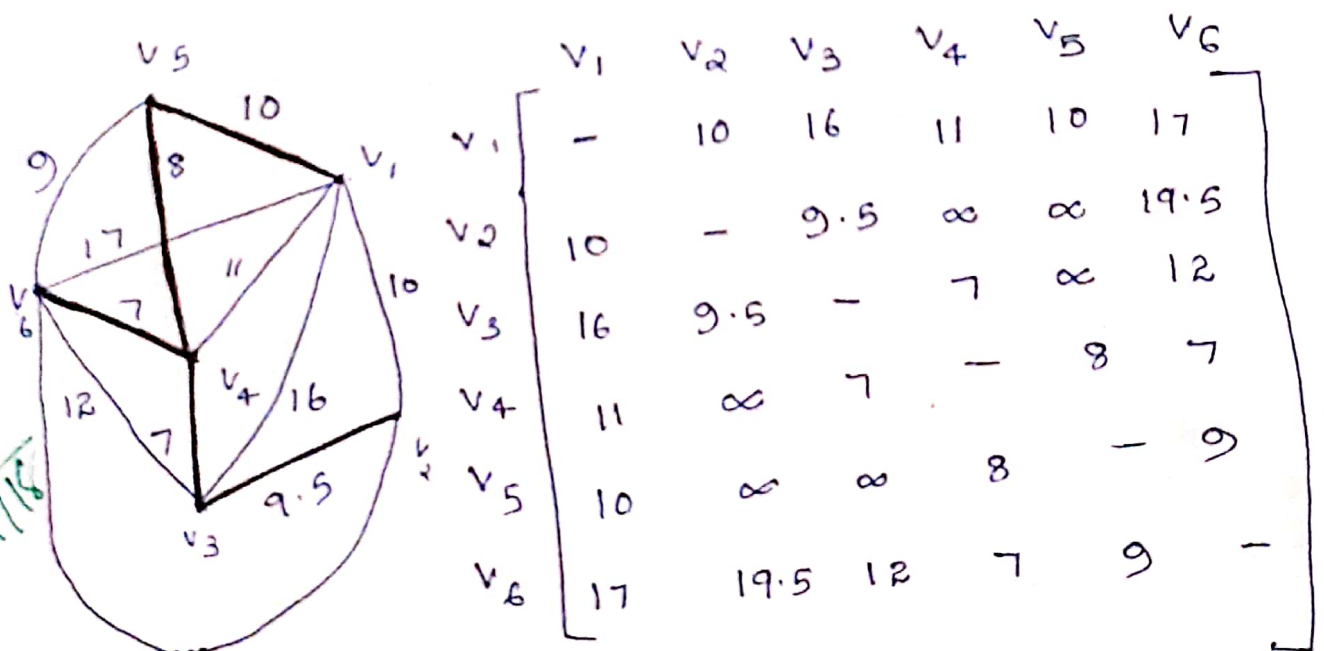
a) The entries in the table are symmetric with respect to the diagonal, and the diagonal is empty.

b) Set the weights of non-existent edges as very large.

3. start from vertex v_1 and connect it to its nearest neighbor (i.e., to the vertex which has the smallest entry in row 1 of the table), say v_k .

Now consider v_1 and v_k as one subgraph, and connect this subgraph to its closest neighbors (i.e., to a vertex other than v_1 and v_k that has the smallest entry among all entries in row 1 and k).

4. Continue step 3 until all n vertices have been connected by $n-1$ edges.



19.5

Fig. 3-19 shortest spanning tree in a weighted graph.

Module 4

CUT-SETS

In a connected graph G , a cut-set is a set of edges whose removal from G leaves G disconnected.

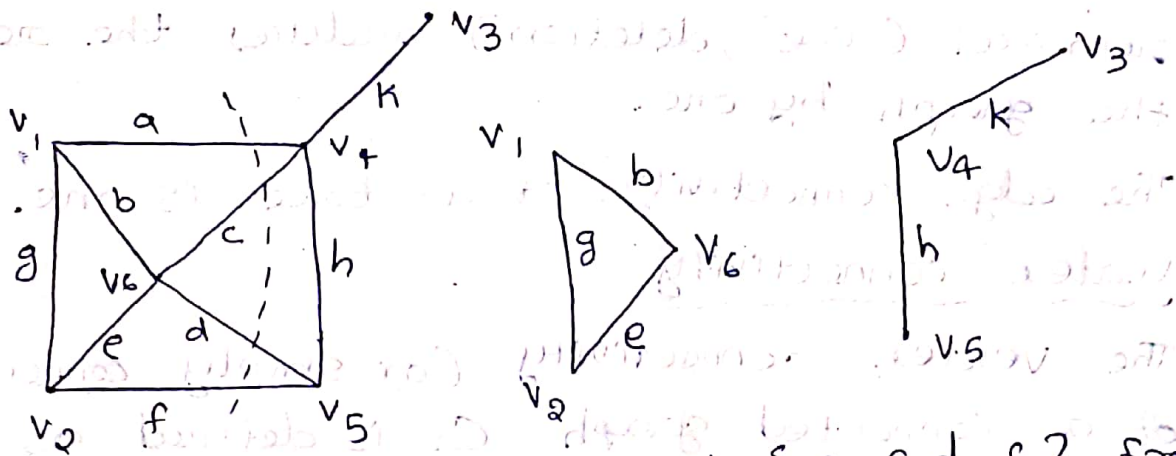


Fig. 4-1 Removal of a cut-set $\{a, c, d, f\}$ from a graph "cuts" it into two.

1. The set of edges $\{a, c, h, d\}$, on the other hand is not a cut-set, because one of its proper subsets $\{a, c, h\}$ is a cut-set.
2. No proper subset of a cut-set can be a cut-set.
3. A cut-set always "cuts" a graph into two.
4. Removal of any edge from a tree breaks the tree into two parts, every edge of a tree is a cut-set.

Applications of cut-sets include

1. studying properties of communication networks &
2. studying properties of transportation networks.

CONNECTIVITY AND SEPERABILITY

Edge connectivity

The number of edges in the smallest cut-set is defined as the edge connectivity of G .

The edge connectivity of a connected graph can be defined as the number of edges, whose removal (i.e., deletion) reduces the rank of the graph by one.

The edge connectivity of a tree is one.

Vertex connectivity,

The vertex connectivity (or simply connectivity) of a connected graph G is defined as the minimum number of vertices whose removal from G leaves the remaining graph disconnected.

The vertex connectivity of a tree is one.

The vertex connectivities of the graphs in Fig 4-1 (a) and

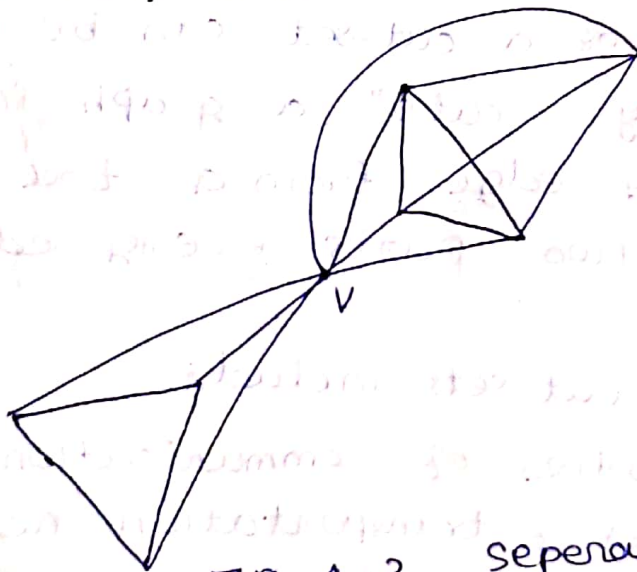


Fig 4.2

seperable graph

1-2 are one and one, respectively.

Seperable Graph:

A connected graph is said to be seperable if its vertex connectivity is one. All other connected graphs are called nonseperable.

cut-vertex

In a seperable graph a vertex whose removal disconnects the graph is called a cut-vertex, a cut-node, or an articulation point.

In a tree every vertex with degree greater than one is a cut-vertex.

Theorem 1-1

A vertex v in a connected graph G is a cut-vertex if and only if there exist two vertices x and y in G such that every path between x and y passes through v .

An Application

suppose we are given n stations that are to be connected by means of e lines (telephone lines, bridges, railroads, tunnels or highways) where $e \geq n-1$. what is the best way of connecting?

Construct a graph with n vertices and e edges that has the maximum possible edge connectivity and vertex connectivity.

Eg:

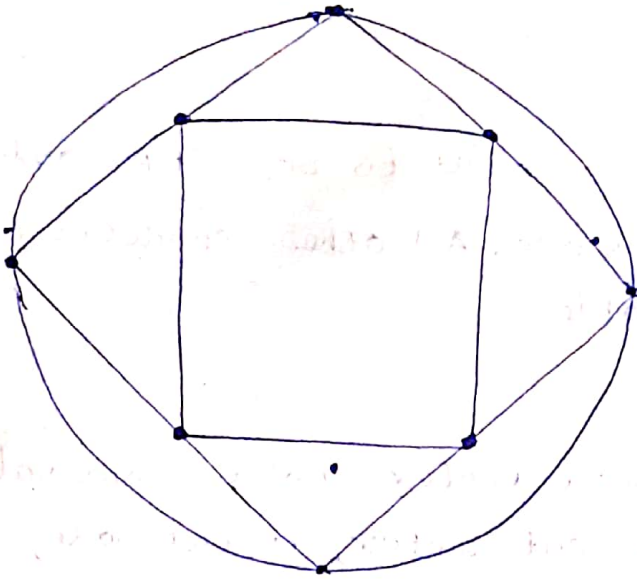


Fig-4-3 Graph with 8 vertices and 16 edges.

In Fig-4-3 the edge connectivity as well as the vertex connectivity of the graph is four.

- Even after any three stations are bombed or
 - any three lines destroyed,
- the remaining stations can still continue to "communicate" with each other.

Thus, the network of Fig. 4-3 is better connected than that of Fig. 4-2.

Theorem 1-2

The edge connectivity of a graph G cannot exceed the degree of the vertex with the smallest degree in G .

Proof:

Let vertex v_i be the vertex with the smallest degree in G . Let $d(v_i)$ be the degree of v_i . Vertex v_i can be separated from G by removing the $d(v_i)$ edges incident on vertex v_i . Hence the theorem.

Theorem 4-3

The vertex connectivity of any graph G can never exceed the edge connectivity of G .

Proof:

Let κ denote the edge connectivity of G . Therefore, there exists a cut-set S in G with κ edges.

Let S partition the vertices of G into subsets V_1 and V_2 . By removing at most κ vertices from V_1 (or V_2) on which the edges in S are incident, we can effect the removal of S (together with all other edges incident on these vertices) from G . Hence the theorem.

COROLLARY

Every cut-set in a nonseparable graph with more than two vertices contains at least two edges.

Theorem 4-4

The maximum vertex connectivity one can achieve with a graph G of n vertices and e

(5)

edges $(e, n-1)$ is the integral part of the number $2e/n$; that is, $\lfloor 2e/n \rfloor$.

Proof:

Every edge in G contributes two degrees. The total ($2e$ degrees) is divided among n vertices. Therefore, there must be at least one vertex in G whose degree is equal to or less than the number $2e/n$. The vertex connectivity of G cannot exceed this number, in light of Theorems -

The edge connectivity of a graph G cannot exceed the degree of the vertex with the smallest degree in G .

and

The vertex connectivity of any graph G can never exceed the edge connectivity of G .

To show that this value can actually be achieved, one can first construct an n -vertex regular graph of degree $\lfloor 2e/n \rfloor$ and then add the remaining $e - (n/2) \cdot \lfloor 2e/n \rfloor$ edges arbitrarily.

k -connected graph

A graph G is said to be k -connected if the vertex connectivity of G is k ; therefore, a 1-connected graph is the same as a separable graph.

PROPERTIES OF A CUT-SET

Consider a spanning tree T in a connected graph G and an arbitrary cut-set S in G .

Is it possible for S not to have any edge in common with T ? The answer is no. Otherwise, removal of the cut-set S from G would not disconnect the graph.

Theorem 4-5

Every cut-set in a connected graph G must contain at least one branch of every spanning tree of G .

Proof:

Let S be a cut-set of G . Let T be a spanning tree of G . Suppose S does not contain any branch of T . Then all the edges of T present in $G-S$. It means $G-S$ is a connected graph. It means S is not a cut-set. Hence a cut-set must contain at least one branch of a spanning tree of G .

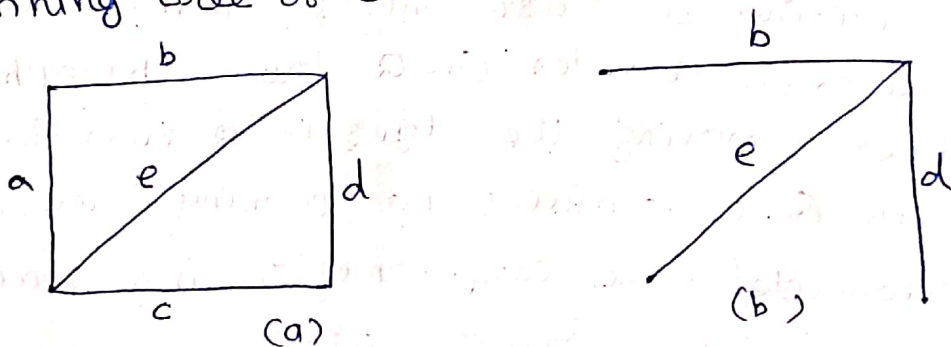


Fig. 1-1 (a) connected graph G (b) spanning tree T .

$[a, c]$ is not a cut-set so it should contain one branch of T to become a cut-set.

Theorem 4-6

In a connected graph G , any minimal set of edges containing at least one branch of every spanning tree of G is a cut-set.

Proof:

Let S be subset of edges of G such that S contains at least one branch of every spanning tree, having minimum number of edges. If $G-S$ is connected it means $G-S$ has spanning tree and no edge of spanning tree present in S . So it must contain an edge of spanning, however it may contain more than one branch of spanning tree. S is a minimal cut-set containing branch of the tree.

OR

In a given connected graph G , let Q be a minimal set of edges containing at least one branch of every spanning tree of G . Consider $G-Q$, the subgraph that remains after removing the edges in Q from G . Since the subgraph $G-Q$ contains no spanning tree of G , $G-Q$ is disconnected (one component of which may just consist of an isolated vertex). Also, since Q is a minimal set of edges with this property, any edge e from Q returned to $G-Q$ will create at least one spanning tree. Thus the subgraph $G-Q+e$ will be a connected graph. Therefore, Q is a minimal set of edges whose

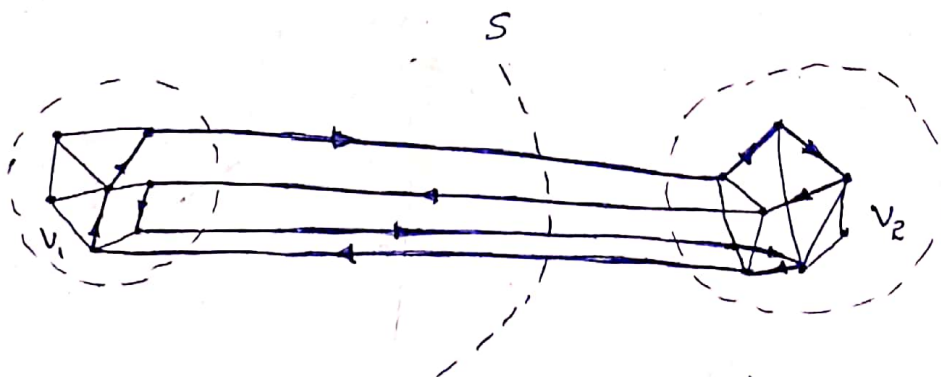
removal from G disconnects G . This, by definition, is a cut-set. Hence the theorem.

Theorem 4-7

Every circuit has an even number of edges in common with any cut-set.

Proof:

consider a cut-set S in graph G (Fig. 4-5). Let the removal of S partition the vertices of G into two (mutually exclusive or disjoint) subsets V_1 and V_2 . Consider a circuit Γ in G . If all the vertices in Γ are entirely within vertex set V_1 (or V_2), the number of edges common to S and Γ is zero; that is, $n(S \cap \Gamma) = 0$, an even number. If, on the other hand, some vertices in Γ are in V_1 and some in V_2 , we traverse back and forth between the sets V_1 and V_2 as we traverse the circuit (see Fig. 4-5).



circuit Γ shown in heavy lines, and its traverse along the direction of the arrows.

Fig. 4-5 Circuit and a cut-set in G .

Because of the closed nature of a circuit, the number of edges we traverse between V_1 and V_2 must be even. And since every edge in S has one end in V_1 and the other in V_2 , and no other edge in G has this property (of separating sets V_1 and V_2), the number of edges common to S and T is even.

ALL CUT-SETS IN A GRAPH

Cut-sets are used to identify weak spots in a communication net.

A spanning tree is essential for defining a set of fundamental circuits, so is a spanning tree essential for a set of fundamental cut-sets.

Fundamental Cut-Sets: (basic cut-set)

Consider a spanning tree T of a connected graph G . A cut-set S containing exactly one branch of a tree T is called a fundamental cut-set with respect to T .

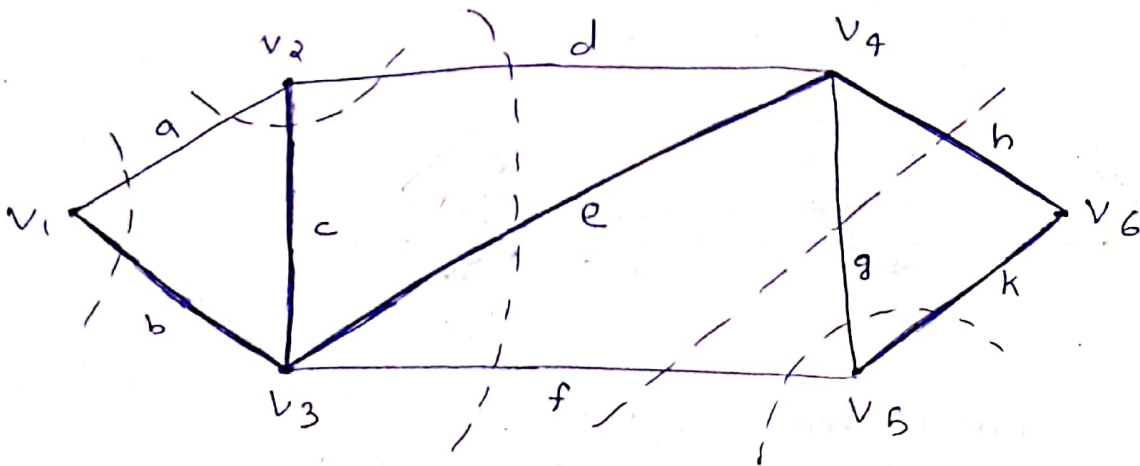


Fig. 4-6 Fundamental cut-sets of a graph.

Just as every chord of a spanning tree defines a unique fundamental ~~set~~ circuit, every branch of a spanning tree defines a unique fundamental cut-set.

Operations ON Graphs.

1. Union

The union of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is another graph G_3 (written as $G_3 = G_1 \cup G_2$) whose vertex set $V_3 = V_1 \cup V_2$ and the edge set $E_3 = E_1 \cup E_2$.

2. Intersection

The intersection $G_1 \cap G_2$ of graphs G_1 and G_2 is a graph G_4 consisting only of those vertices and edges that are in both G_1 and G_2 .

3. Ring sum.

The ring sum of two graphs G_1 and G_2 (written as $G_1 \oplus G_2$) is a graph consisting of the vertices set $V_1 \cup V_2$ and of edges that are either in G_1 or G_2 , but not both.

4. Decomposition

A graph G is said to have been decomposed into two subgraphs g_1 and g_2 if

$$g_1 \cup g_2 = G,$$

and

$$g_1 \cap g_2 = \text{a null graph.}$$

Every edge of G occurs either in g_1 or in g_2 , but not in both.

Some of the vertices occur in both g_1 and g_2 .

A graph containing m edges $\{e_1, e_2, \dots, e_m\}$ can be decomposed in 2^{m-1} different ways into pairs of subgraphs g_1, g_2 .

5. Deletion :

If v_i is a vertex in graph G , then $G - v_i$ denotes a subgraph of G obtained by deleting (i.e., removing) v_i from G .

Deletion of a vertex always implies the deletion of all edges incident on that vertex.

Deletion of an edge e_j :

If e_j is an edge in G , then $G - e_j$ is a subgraph of G obtained by deleting e_j from G .

Deletion of an edge does not imply deletion of its end vertices. Therefore $G - e_j = G \oplus e_j$.

6. Fusion :

A pair of vertices a, b in a graph are said to be fused (merged or identified) if the two vertices are replaced by a single new vertex such that every edge that was incident on either a or b or on both is incident on the new vertex.

Fusion of two vertices does not alter the number of edges, but it reduces the number of vertices by one.

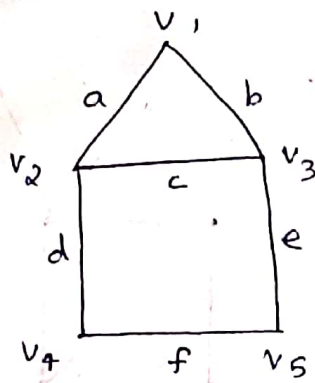
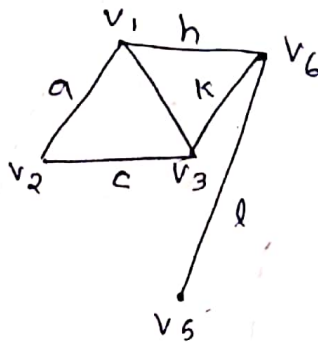
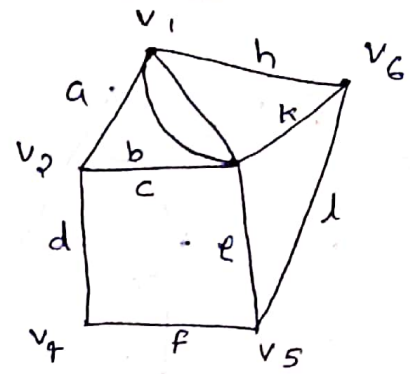
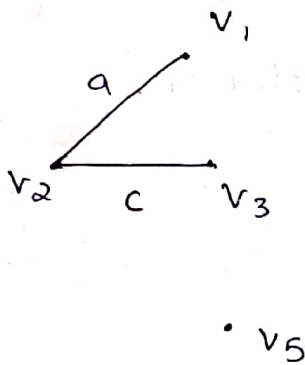
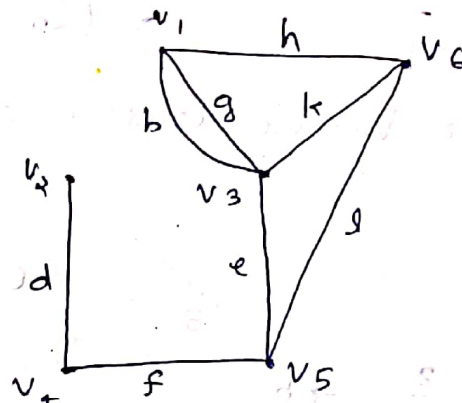
 G_1  G_2  $G_1 \cup G_2$  $G_1 \cap G_2$  $G_1 \oplus G_2$

Fig. 4-7 Union, intersection and ring sum of two graphs.

Commutativity

$$G_1 \cup G_2 = G_2 \cup G_1,$$

$$G_1 \cap G_2 = G_2 \cap G_1,$$

$$G_1 \oplus G_2 = G_2 \oplus G_1.$$

For any graph G

$$G \cup G = G \cap G = G,$$

and $G \oplus G = \text{a null graph}.$

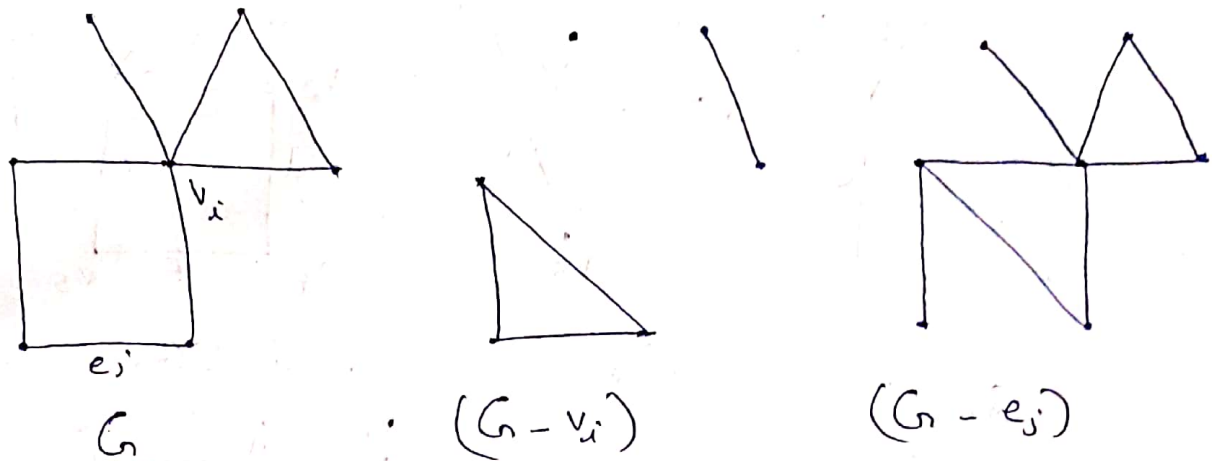


Fig. 4-8 Vertex deletion and edge deletion.

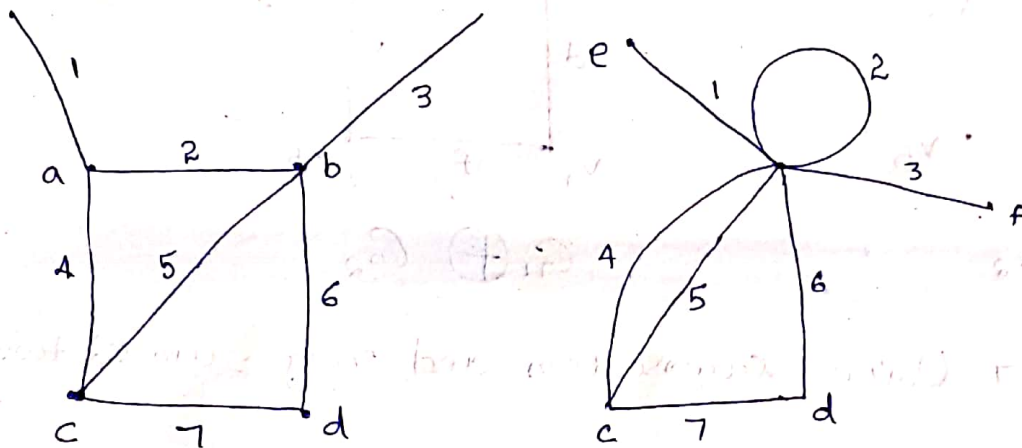


Fig. 4-9 Fusion of vertices a and b.

Theorem 4-7

The ring sum of any two cut-sets in a graph is either a third cut-set or an edge disjoint union of cut-sets.

Proof

Let S_1 and S_2 be two cut-sets in a given connected graph G . Let V_1 and V_2 be the (unique and disjoint) partitioning of the vertex set V of G corresponding to S_1 . Let V_3 and V_4 be the partitioning corresponding

to S_2 , clearly [see Figs. 4-10(a) and (b)],

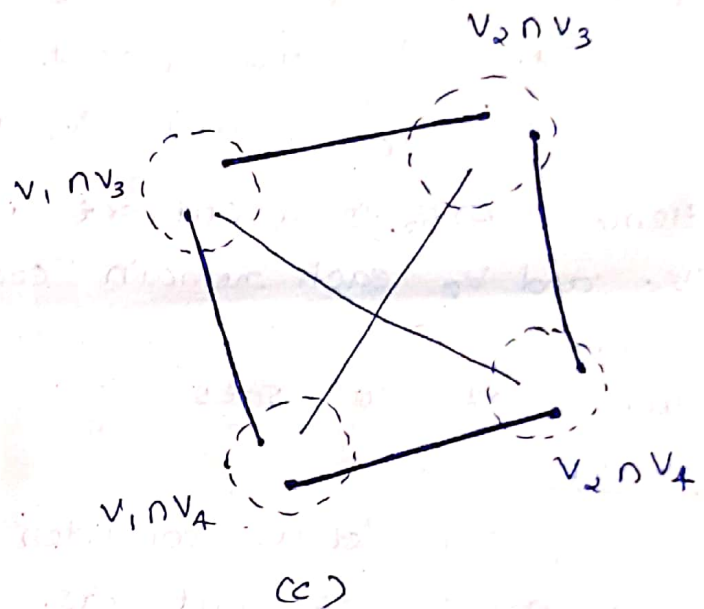
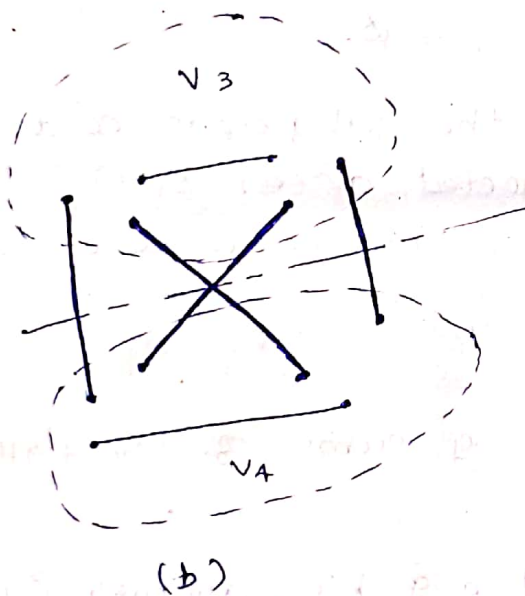
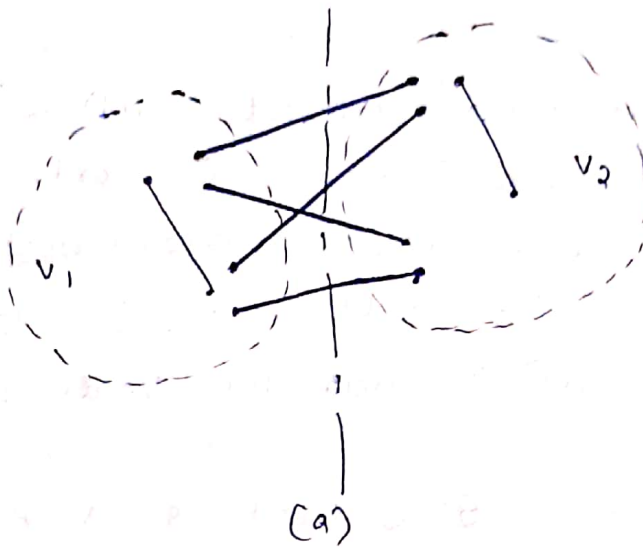


Fig. 4-10 Two cut-sets and their partitionings.

$$V_1 \cup V_2 = V \quad \text{and} \quad V_1 \cap V_2 = \emptyset,$$

$$V_3 \cup V_4 = V \quad \text{and} \quad V_3 \cap V_4 = \emptyset.$$

Now let the subset $(V_1 \cap V_4) \cup (V_2 \cap V_3)$ be called V_5 , and this by definition is the same as the ring sum $V_1 \oplus V_3$. Similarly, let the subset $(V_1 \cap V_3) \cup (V_2 \cap V_4)$

be called V_6 , which is the same as $V_2 \oplus V_3$.
See Fig. 4-10(c).

The ring sum of the two cut-sets $S_1 \oplus S_2$ can be seen to consist only of edges that join vertices in V_5 to those in V_6 . Also, there are no edges outside $S_1 \oplus S_2$ that join vertices in V_5 to those in V_6 . Also, there are no edges outside $S_1 \oplus S_2$ that join vertices in V_5 to those in V_6 .

Thus the set of edges $S_1 \oplus S_2$ produces a partitioning of V into V_5 and V_6 such that

$$V_5 \cup V_6 = V \text{ and } V_5 \cap V_6 = \emptyset.$$

Hence $S_1 \oplus S_2$ is a cut-set if the subgraphs containing V_5 and V_6 each remain connected after $S_1 \oplus S_2$ is removed from G . Otherwise, $S_1 \oplus S_2$ is an edge-disjoint union of cut-sets.

Example

In Fig. 4-6 let us consider ring sums of the following three pairs of cut-sets.

$$\{d, e, f\} \oplus \{f, g, h\} = \{d, e, g, h\}, \text{ another cut-set,}$$

$$\{a, b\} \oplus \{b, c, e, f\} = \{a, c, e, f\}, \text{ another cut-set,}$$

$$\{d, e, g, h\} \oplus \{f, g, k\} = \{d, e, f, h, k\}$$

$$= \{d, e, f\} \cup \{h, k\}, \text{ an edge-disjoint union of cut-sets.}$$

FUNDAMENTAL CIRCUITS AND CUT-SETS

consider a spanning tree T in a given connected graph G . Let C_i be a chord with respect to T , and let the fundamental circuit made by C_i be called Γ , consisting of k branches b_1, b_2, \dots, b_k in addition to the chord C_i ; that is,

$\Gamma = \{C_i, b_1, b_2, \dots, b_k\}$ is a fundamental circuit with respect to T .

Every branch of any spanning tree has a fundamental cut-set associated with it. Let S_i be the fundamental cut-set associated with b_i , consisting of q chords in addition to the branch b_i ; that is,

$S_i = \{b_i, C_1, C_2, \dots, C_q\}$ is a fundamental cut-set with respect to T .

Theorem 4-8

With respect to a given spanning tree T , a chord C_i that determines a fundamental circuit Γ occurs in every fundamental cut-set associated with the branches in Γ and in no other.

Proof:

Every circuit has an even number of edges in common with any cut-set. Because of this, there must be an even number of edges common to Γ and S_i , where S_i be the fundamental cut-set associated with b_i , consisting of q chords in addition to the branch b_i ; that is, $S_i = \{b_i, C_1, C_2, \dots, C_q\}$ is a fundamental cut-set with respect to T .

Edge b_1 is in both Γ and S_1 , and there is only one other edge in Γ (which is c_1) that can possibly also be in S_1 . Therefore, we must have two edges b_1 and c_1 common to S_1 and Γ . Thus the chord c_1 is one of the chords c_1, c_2, \dots, c_q .

Exactly the same argument holds for fundamental cut-sets associated with b_2, b_3, \dots , and b_k . Therefore, the chord c_i is contained in every fundamental cut-set associated with branches in Γ .

Let S' be another fundamental cut-set for the chord c_i with respect to T . It is not possible for the chord c_i to be in any other S' besides those associated with b_1, b_2, \dots and b_k . Otherwise (since none of the branches in Γ are in S'), there would be only one edge c_i common to S' and Γ , a contradiction to the theorem that "Every circuit has an even number of edges in common with any cut-set." Hence the theorem.

Theorem 4-9

With respect to a given spanning tree T , a branch b_i that determines a fundamental cut-set S is contained in every fundamental circuit associated with the chords in S , and in no others.

Proof

The proof consists of arguments.

Let the fundamental cut-set S determined by a branch b_i be $S = \{b_i, c_1, c_2, \dots, c_p\}$,

and let Γ_1 be the fundamental circuit determined by chord C_1 .

$$\Gamma_1 = \{C_1, b_1, b_2, \dots, b_q\}.$$

Since the number of edges common to S and Γ_1 must be even, b_1 must be in Γ_1 . The same is true for the fundamental circuits made by chords C_2, C_3, \dots, C_p . On the other hand, suppose that b_1 occurs in a fundamental circuit

Γ_{p+1} made by a chord other than C_1, C_2, \dots, C_p . Since none of the chords C_1, C_2, \dots, C_p is in Γ_{p+1} , there is one and only one edge b_1 common to a circuit Γ_{p+1} and the cut-set S , which is not possible. Hence the theorem.

Example

Consider a spanning tree $\{b, c, e, h, k\}$, shown in heavy lines in Fig 4-6. The fundamental circuit made by chord f is

$$\{f, e, h, k\}.$$

The three fundamental cut-sets determined by the three branches e, h and k are determined by branch e : $\{d, e, f\}$, determined by branch h : $\{f, g, h\}$, determined by branch k : $\{f, g, k\}$.

Chord f occurs in each of these three fundamental cut-sets, and there is no

other fundamental cut-set that contains f .

consider branch e of spanning tree $\{b, c, e, h, k\}$. The fundamental cut-set determined by e is $\{e, d, f\}$.

The two fundamental circuits determined by chords d and f are determined by chord d : $\{d, c, e\}$, determined by chord f : $\{f, e, h, k\}$.

Branch e is contained in both these fundamental circuits, and none of the remaining three fundamental circuits contains branch e .

PLANAR GRAPHS

Is it possible to draw G in a plane without its edges crossing over?

Applications

1. In the design of a printed-circuit board, the electrical engineer must know if he can make the required connections without an extra layer of insulation.
2. The solution to the puzzle of three utilities, requires the knowledge of whether or not the corresponding graph can be drawn in a plane.

Combinatorial Versus Geometric Graphs

A graph exists as an abstract object.

Euclidean space

A space in any finite number of dimensions, in which points are designated by coordinates (one for each dimension).

Abstract graph (Combinatorial)

An abstract graph G_1 can be defined as

$$G_1 = (V, E, \psi),$$

where

$$V = \{a, b, c, d, e\}$$

$$E = \{1, 2, 3, 4, 5, 6, 7\}$$

$\psi \Rightarrow$ the relationship between the two sets

(21)

$$\Psi = \begin{cases} 1 \rightarrow (a, c) \\ 2 \rightarrow (c, d) \\ 3 \rightarrow (a, d) \\ 4 \rightarrow (a, b) \\ 5 \rightarrow (b, d) \\ 6 \rightarrow (d, e) \\ 7 \rightarrow (b, e) \end{cases}$$

The symbol $1 \rightarrow (a, c)$ says that object 1 from set E is mapped onto the (unordered) pair (a, c) of objects from set V .

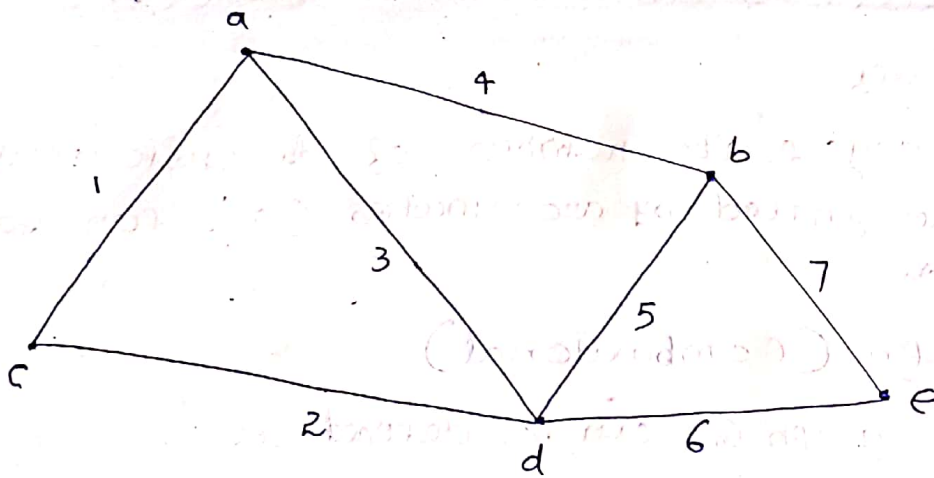


Fig. 4-11 Geometric representation of G_1

PLANAR GRAPHS

A graph G is said to be planar if there exists some geometric representation of G which can be drawn on a plane such that no two of its edges intersect.

NONPLANAR

A graph that cannot be drawn on a plane without a crossover between its edges is called a nonplanar.

Embedding

A drawing of a geometric representation of a graph on any surface such that no edges intersect is called embedding.

Plane representation of G

An embedding of a planar graph G on a plane is called a plane representation of G .

Eg:

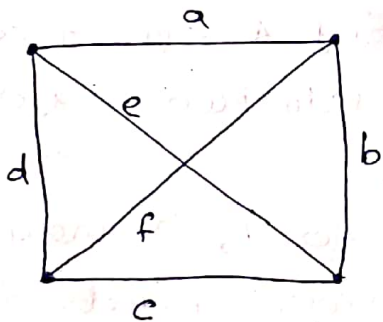


Fig. 4-12 Geometric representation

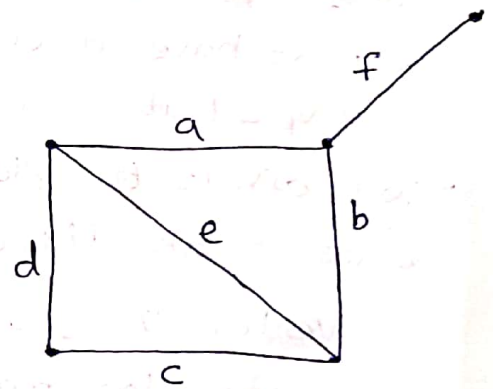


Fig. 4-13 Embedded the new geometric graph.

The geometric representation shown in Fig. 4-12 clearly is not embedded in a plane, because the edges e and f are intersecting.

But if we redraw edge f outside the quadrilateral, leaving the other edges unchanged as in Fig. 4-13, thus showing that the graph which is being represented by Fig. 4-12 is planar.

KURATOWSKI'S TWO GRAPHS

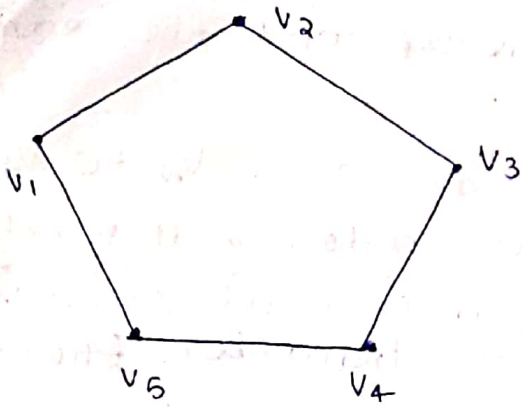
Theorem 4-10

The complete graph of five vertices is nonplanar.

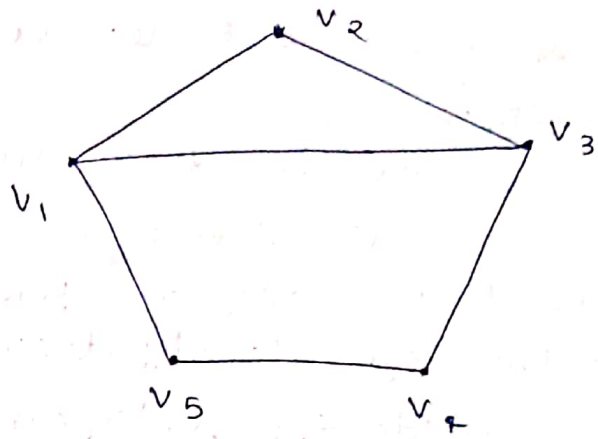
Proof:

Let the five vertices in the complete graph be named v_1, v_2, v_3, v_4 and v_5 . A complete graph, as you may recall, is a simple graph in which every vertex is joined to every other vertex by means of an edge. This being the case, we must have a circuit going from v_1 to v_2 to v_3 to v_4 to v_5 to v_1 - that is, a pentagon. See Fig. 4-14(a). This pentagon must divide the plane of the paper into two regions, one inside and the other outside.

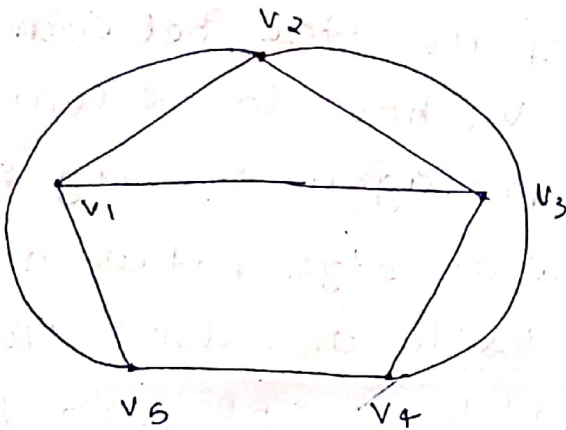
Since vertex v_1 is to be connected to v_3 by means of an edge, this edge may be drawn inside or outside the pentagon (without intersecting the five edges drawn previously). Suppose that we choose to draw a line from v_1 to v_3 inside the pentagon. See Fig. 4-14(b).



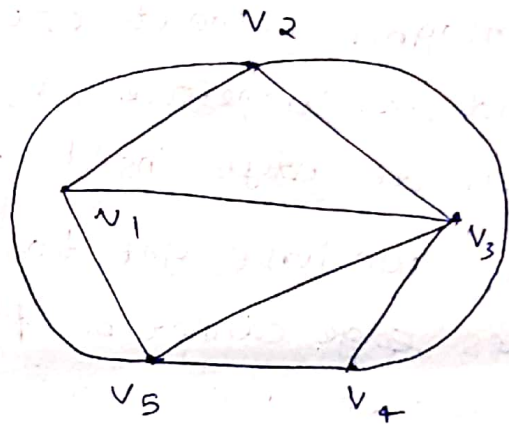
(a)



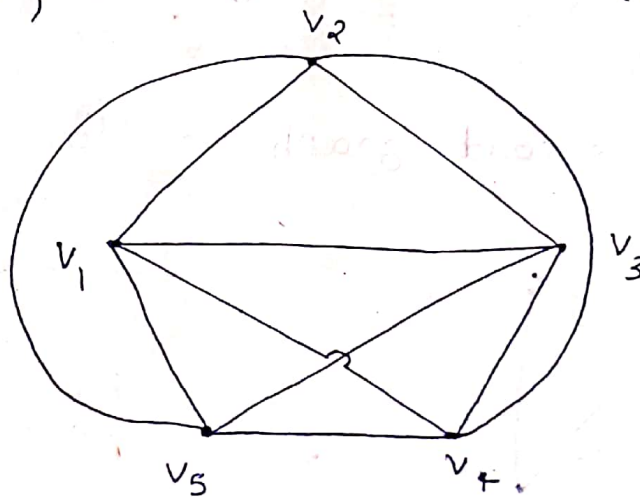
(b)



(c)



(d)



(e)

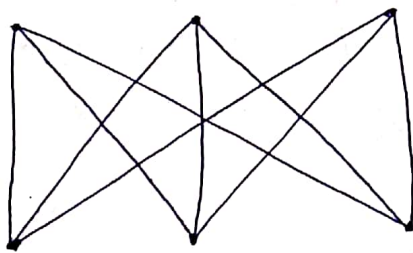
Fig. 4-14 Building up of the five-vertex complete graph.

(If we choose outside, we end up with the same argument.)

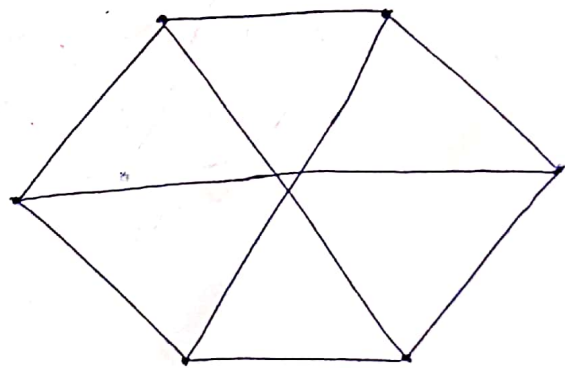
Now we have to draw an edge from V_2 to V_4 and another one from V_2 to V_5 . Since neither of these edges can be drawn inside the pentagon without crossing over the edge already drawn, we draw both these edges outside the pentagon. See Fig. 4-14(c). The edge connecting V_3 and V_5 cannot be drawn outside the pentagon without crossing the edge between V_2 and V_4 . Therefore, V_3 and V_5 have to be connected with an edge inside the pentagon. See Fig. 4-14(d). Now we have yet to draw an edge between V_1 and V_4 . This edge cannot be placed inside or outside the pentagon without a crossover. Thus the graph cannot be embedded in a plane. See Fig. 4-14(e).

Theorem 4-11

Kuratowski's second graph is also nonplanar.



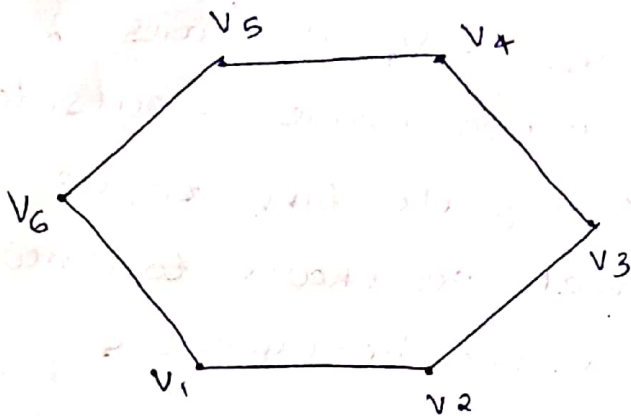
(a)



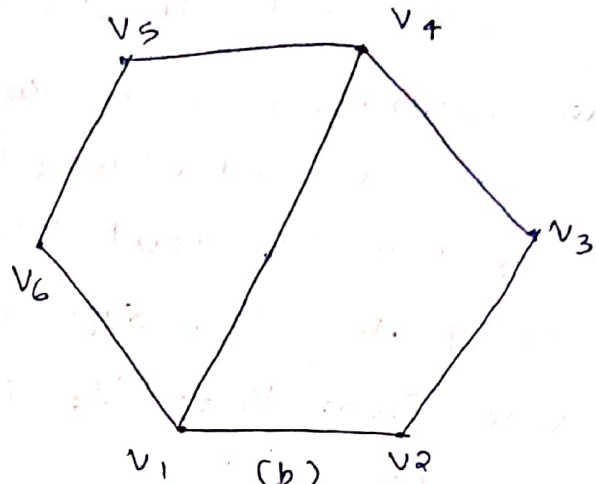
(b)

Fig. 4-15 Kuratowski's second graph.

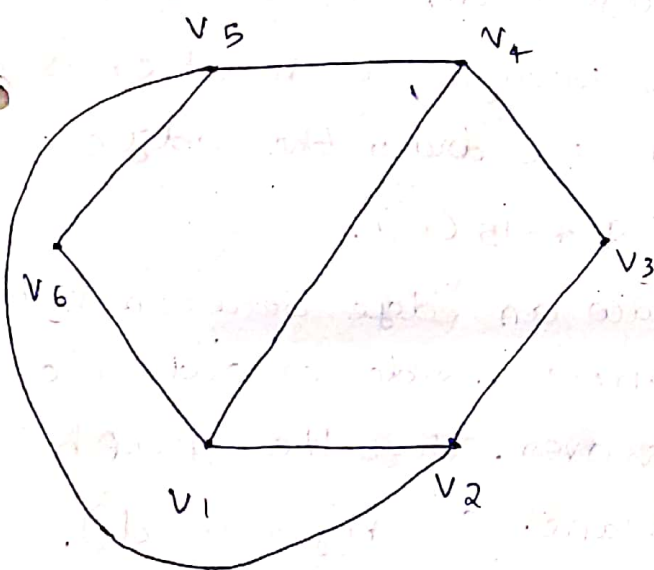
Proof:



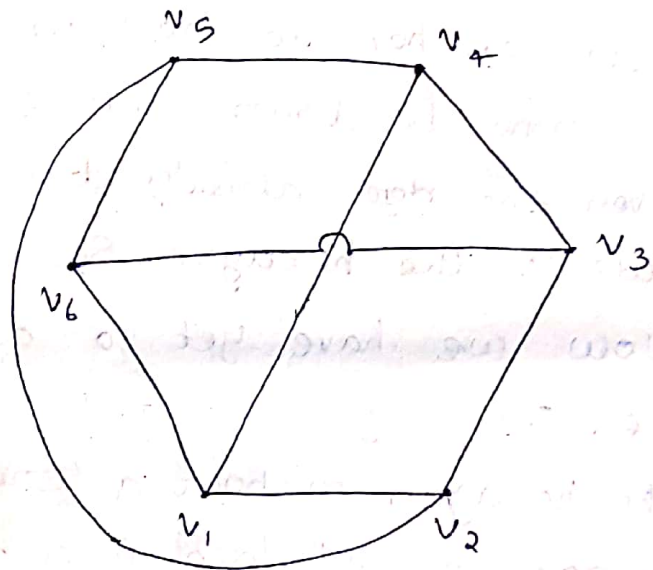
(a)



(b)



(c)



(d)

Fig. 4-16 Building up of Kuratowski's second graph.

Let the six vertices in the Kuratowski's second graph be named V_1, V_2, V_3, V_4, V_5 and V_6 . We must have a circuit going from V_1 to V_2 to V_3 to V_4 to V_5 to V_6 to V_1 , that is a hexagon. See Fig. 4-16(a). This hexagon must divide the plane of the paper into two regions,

one inside and the other outside.

Since vertex V_1 is to be connected to V_4 by means of an edge, this edge may be drawn inside or outside the hexagon (without intersecting the five edges drawn previously). Suppose that we choose to draw a line from V_1 to V_4 inside the hexagon. See Fig. 4-15(b).

Now we have to draw an edge from V_2 to V_5 . Since it cannot be drawn inside the hexagon without crossing over the edge already drawn, we draw the edge outside the hexagon. See Fig. 4-15(c).

Now we have yet to draw an edge between V_3 and V_6 . This edge cannot be placed inside or outside the hexagon without a crossover. Thus the graph cannot be embedded in a plane. See Fig. 4-15(d).

Properties common to the Kuratowski's two graphs.

1. Both are regular graphs.
2. Both are nonplanar.
3. Removal of one edge or a vertex makes each a planar graph.
4. Kuratowski's first graph is the nonplanar graph with the smallest number of vertices, and Kuratowski's second graph is the nonplanar graph with the smallest number of edges. Thus both are the simplest non planar graphs.

DIFFERENT REPRESENTATIONS OF A PLANAR GRAPH.

Theorem 4-12:

Any simple planar graph can be embedded in a plane such that every edge is drawn as a straight line segment.

Proof:

As an illustration, the graph in Fig. 4-14 (d) can be redrawn using straight line segments to look like Fig. 4-17. In this theorem, it is necessary for the graph to be simple because a self-loop or one or two parallel edges cannot be drawn by a straight line segment.

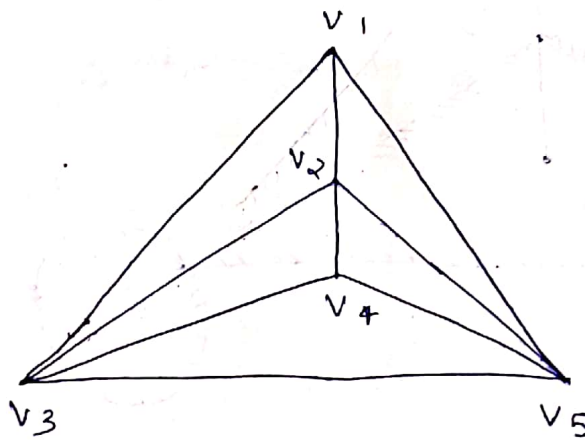


Fig. 4-17 Straight-line representation of the graph in Fig. 4-14 (d).

Region:

A plane representation of a graph divides the plane into regions (also called windows, faces, or meshes), as shown in Fig. 4-18.

A region is characterized by the set of edges (or the set of vertices) forming its boundary.

Region is not defined in a nonplanar graph or even in a planar graph not embedded in a plane.

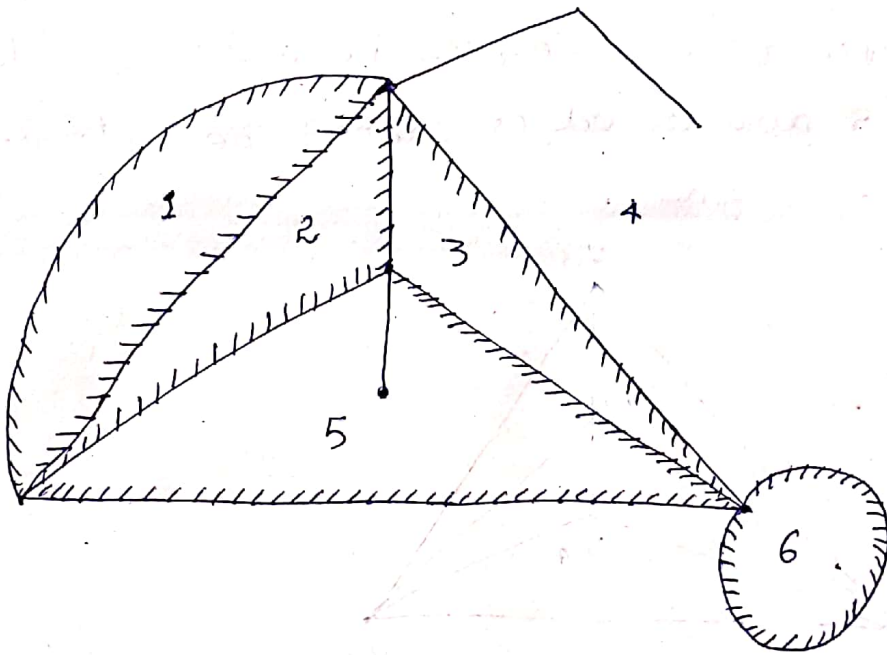


Fig. 4-18 Plane representation (the numbers stand for regions).

Infinite Region: (unbounded, outer or exterior)

The portion of the plane lying outside a graph embedded in a plane is infinite in its extent.

Eg.:

A region 4 in Fig. 4-18.

Like other regions, the infinite region is also characterized by a set of edges (or vertices), by changing the embedding of a given planar graph, we can change the infinite region.

Eg: Fig. 4-14(d) and 4-17 are two different embeddings of the same graph.

- The finite region $V_1 V_3 V_5$ in Fig. 4-14(d) becomes the infinite region in 4-17.

Embedding on a sphere:

- Put the sphere on the plane and call the point of contact SP (south pole).
- At point SP, draw a straight line perpendicular to the plane upto NP (north pole).
- For any point P on the plane, there exists a unique point P' on the sphere and vice versa.
 - P' is the point at which the straight line from point P to point NP intersects the surface of the sphere.
- * Any graph that can be embedded in a plane can also be embedded in the surface of the sphere, and vice versa.

Theorem 4-13

A graph can be embedded in the surface of a sphere if and only if it can be embedded in a plane.

Theorem 4-14

A planar graph may be embedded in a plane such that any specified region (i.e., specified by the edges forming it) can be made the infinite region.

A planar graph embedded in the surface of a sphere if and only if it can be added in a plane. Sphere divides the surface into different regions. Each region on the sphere is finite, the finite region on the plane having been mapped onto the region containing the point ∞ . Now it is clear that by suitably rotating the sphere we can make any specified region map onto the infinite region on the plane.

Euler's Formula:

Since a planar graph may have different plane representations.

Is the number of regions resulting from each embedding is the same.

The answer is yes.

Theorem 1-15

A connected planar graph with n vertices and e edges has $e - n + 2$ regions.

Proof:

It will suffice to prove the theorem for a simple graph, because adding a self-loop or a parallel edge simply adds one region to the graph and simultaneously increases the value of e by one. We can also disregard (i.e., remove) all edges that do not form boundaries of any region. Three such edges are shown in Fig. 4-18. Addition (or removal) of any such edge increases (or decreases) e by one and increases (or decreases) n by one, keeping the quantity $e - n$ unaltered.

Since any simple planar graph can have a plane representation such that each edge is a straight line, any planar graph can be drawn such that each region is a polygon (a polygonal net). Let the polygonal net representing the given graph consist of f regions or faces, and let k_p be the number of p -sided regions.

Since each edge is on the boundary of exactly two regions,

$$3 \cdot k_3 + 4 \cdot k_4 + 5 \cdot k_5 + \dots + r \cdot k_r = 2 \cdot e, \quad (1)$$

where k_r is the number of polygons, with maximum edges.

Also,

$$k_3 + k_4 + k_5 + \dots + k_r = f \quad (2)$$

The sum of all angles subtended at each vertex in the polygonal net is

$$2\pi n. \quad (3)$$

Recalling that the sum of all interior angles of a p -sided polygon is $\pi(p-2)$, and the sum of the exterior angles is $\pi(p+2)$, let us compute the expression in (3) as the grand sum of all interior angles of $f-1$ finite regions plus the sum of the exterior angles of the polygon defining the infinite region. This sum is:

$$\begin{aligned} & \pi(3-2) \cdot k_3 + \pi(4-2) \cdot k_4 + \dots + \pi(r-2) \cdot k_r + 4\pi \\ & = \pi(2e - 2f) + 4\pi. \end{aligned} \quad (4)$$

Equating (4) to (3), we get

$$2\pi(e-f) + 4\pi = 2\pi n,$$

$$e - f + 2 = n.$$

or

Therefore, the number of regions is

$$f = e - n + 2.$$

COROLLARY

In any simple, connected planar graph with f regions, n vertices, and e edges ($e > 2$), the following inequalities must hold:

$$e \geq \frac{3}{2}f, \quad (5)$$

$$e \leq 3n - 6. \quad (6)$$

Proof:

Since each region is bounded by at least three edges and each edge belongs to exactly two regions,

$$2e \geq 3f$$

or

$$e \geq \frac{3}{2}f.$$

substituting for f from Euler's formula in inequality (5),

$$e \geq \frac{3}{2}(e - n + 2)$$

$$\text{or } e \leq 3n - 6.$$

∴

Plane Representation and Connectivity:

A disconnected graph is planar if and only if each of its components is planar.

If of all possible embeddings on a sphere no two are distinct, the graph is said to have a unique embedding on a sphere (or a unique plane representation).

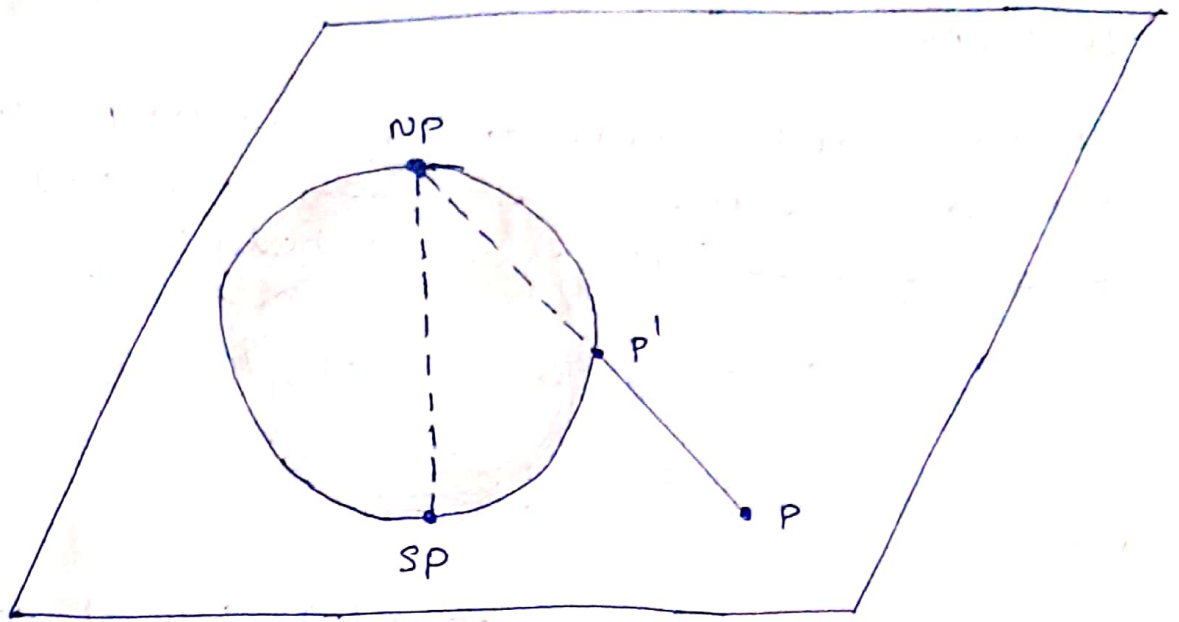


Fig. 4-19 Stereographic projection.

The two embeddings are distinct, and the ~~two~~ graph has no unique plane representation.

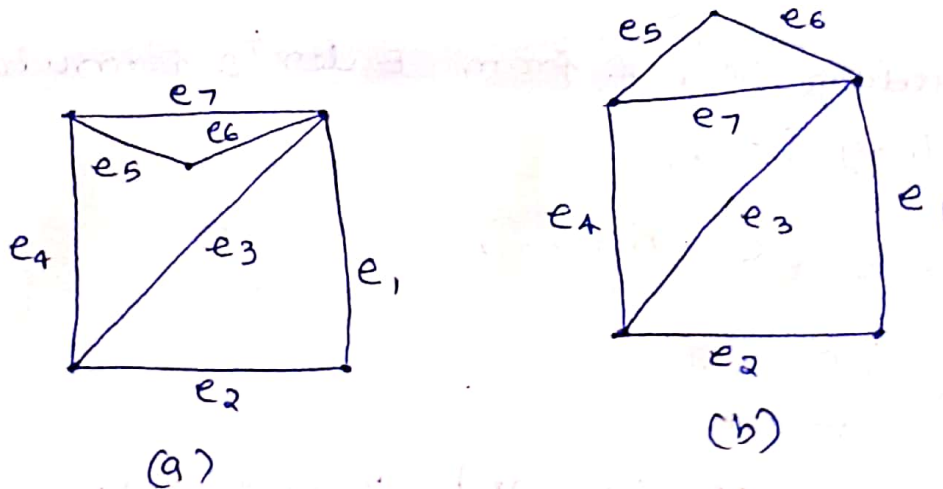


Fig. 4-20 Two distinct plane representation of the same graph.

Theorem 4-16

The spherical embedding of every planar 3-connected graph is unique.

(36)

GEOMETRIC DUAL

1-isomorphic

Two separable graphs G_1 and G_2 are said to be 1-isomorphic if they become isomorphic to each other under repeated application of the following operation.

* Operation 1:

"split" a cut-vertex into two vertices to produce two disjoint subgraphs.

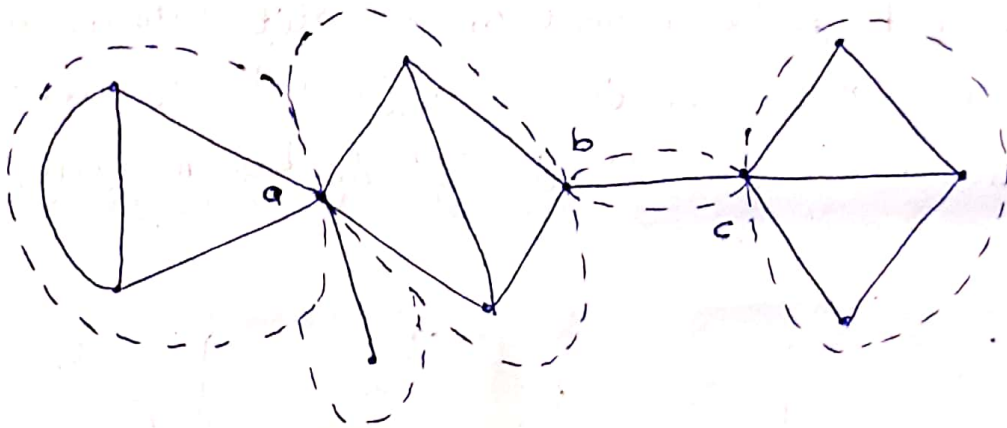


Fig. 4-21 Separable graph with three cut-vertices and five blocks.

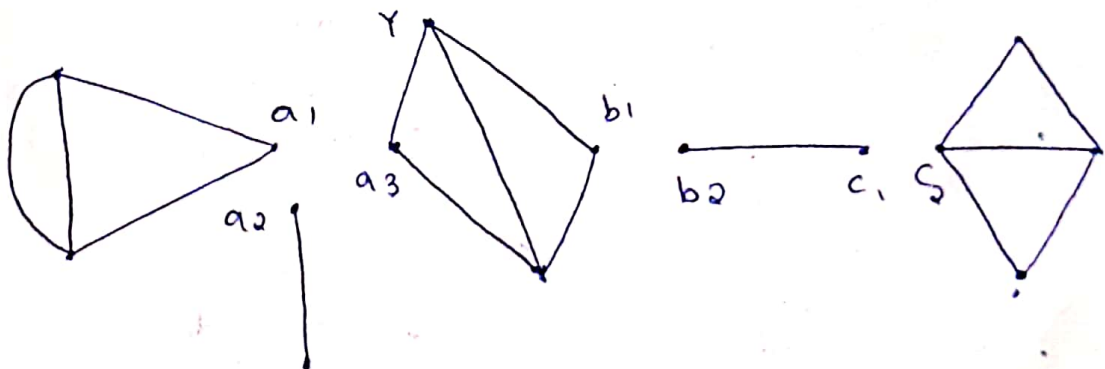


Fig. 4-22 Disconnected graph 1-isomorphic to Fig. 4-21.

2- ISOMORPHISM

Graphs with vertex connectivity of two.

Two graphs are said to be 2-isomorphic if they become isomorphic after undergoing

1. operation 1 or
2. Operation 2 or
3. Both operations any number of times.

* Operation 2 :

"Split" the vertex x into x_1 and x_2 and the vertex y into y_1 and y_2 such that G is split into g_1 and \bar{g}_1 .

Let vertices x_1 and y_1 go with g_1 and x_2 and y_2 with \bar{g}_1 .
Now rejoin the graphs g_1 and \bar{g}_1 by merging x_1 with y_2 and x_2 with y_1 .

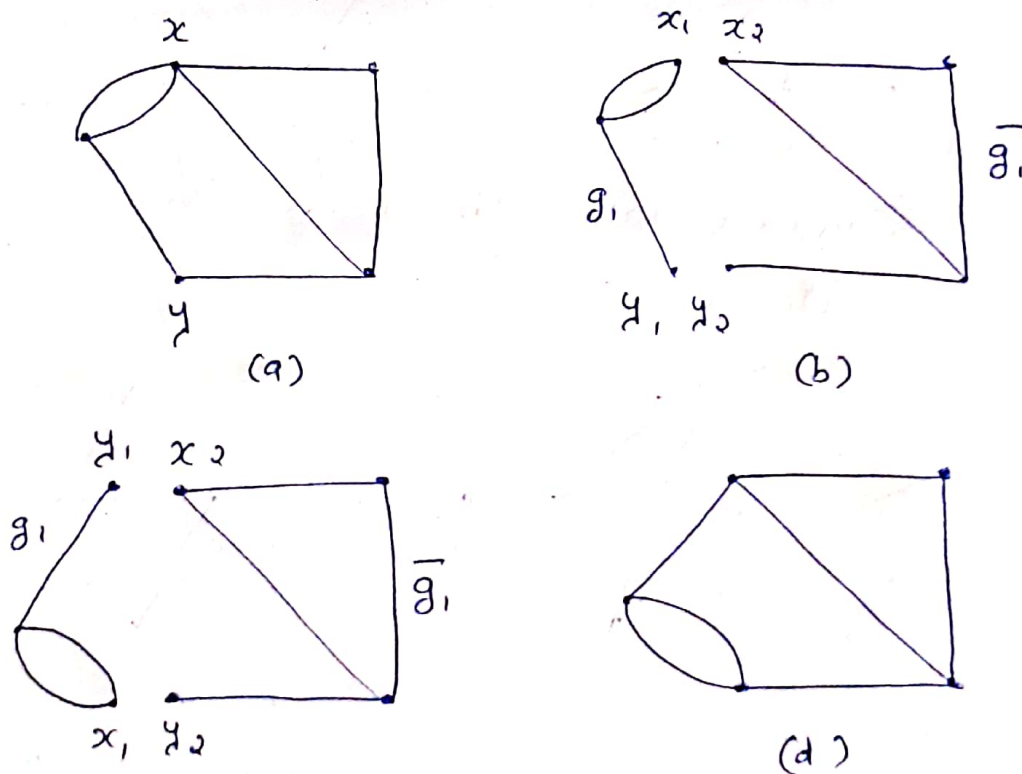


Fig. 4-23 2-isomorphic graphs (a) and (d).

Construction of Dual of G (G^*)

1. Place n points p_1, p_2, \dots, p_n one in each of the regions F_1, F_2, \dots, F_n .

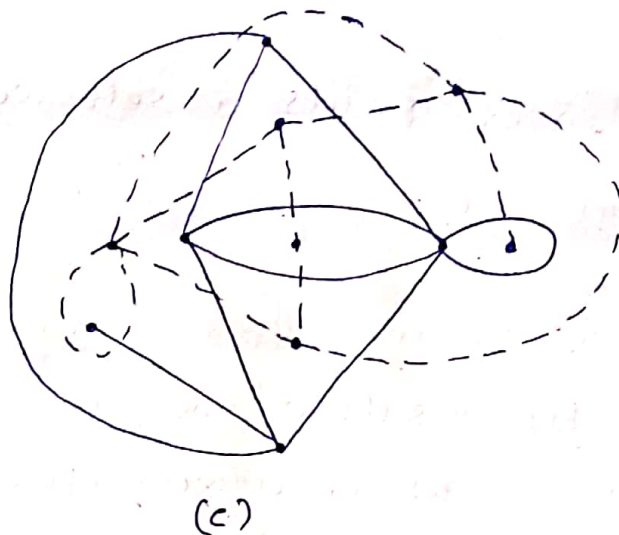
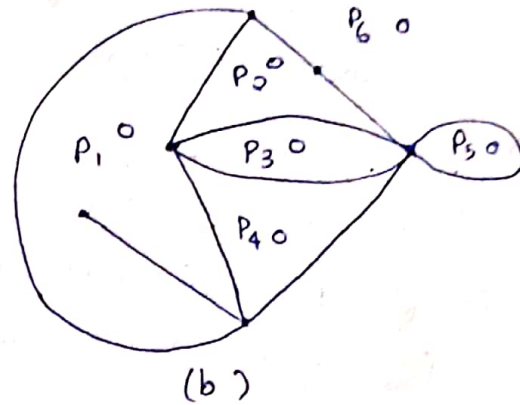
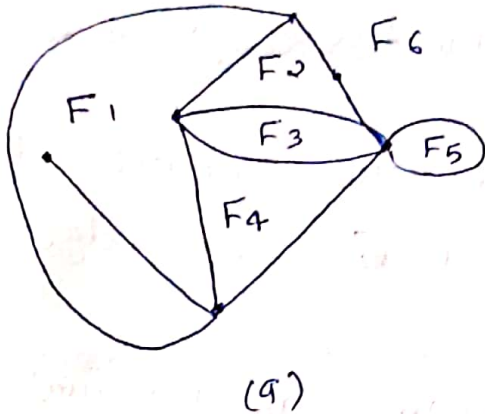


Fig. 4-24 Construction of a dual graph.

2. Join the n points according to the following.
- (a) If two regions F_i and F_j are adjacent (i.e., have a common edge), draw a line joining points p_i and p_j that intersects the common edge between F_i and F_j exactly once.

(b) If there is more than one edge common between F_i and F_j , draw one line between points P_i and P_j for each of the common edges.

(b') For an edge e lying entirely in one region, say F_k , draw a self-loop at point P_k intersecting e exactly once.

There is a one-to-one correspondence between the edges of graph G and its dual G^* - one edge of G^* intersecting one edge of G .

Relationship between a planar graph G and its dual G^*

1. An edge forming a self-loop in G yields a pendant edge in G^* .
2. A pendant edge in G yields a self-loop in G^* .
3. Edges that are in series in G produce parallel edges in G^* .
4. Parallel edges in G produce edges in series in G^* .
5. Remarks 1-4 are the result of the general observation that the number of edges constituting the boundary of a region F_i in G is equal to the degree of the corresponding vertex P_i in G^* , and vice versa.
6. Graph G^* is also embedded in the plane and is therefore planar.
7. Considering the process of drawing a dual G^* from G , it is evident that G is a dual of G^* . Therefore, instead of calling G^* a dual of G , we usually say that G and G^* are dual graphs.

8. If n, e, f, r and u denote as usual the numbers of vertices, edges, regions, rank, and nullity of a connected planar graph G , and if n^*, e^*, f^*, r^* and u^* are the corresponding numbers in dual graph G^* , then

$$n^* = f,$$

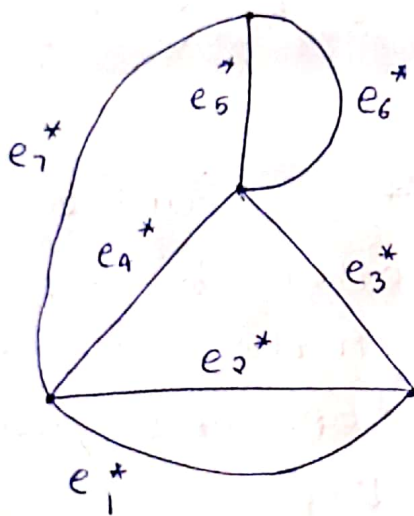
$$e^* = e,$$

$$f^* = n.$$

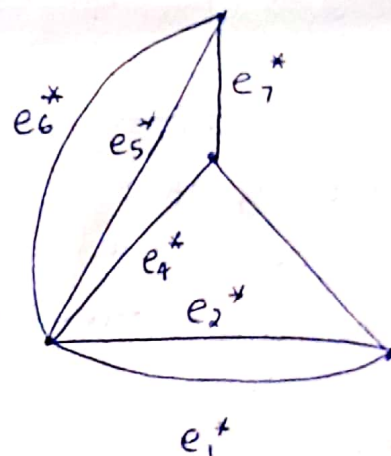
Using the above relationship,

$$r^* = u,$$

$$u^* = r.$$



(a) Dual of 4-20(a)



(b) Dual of 4-20(b)

Fig. 4-25 Duals of graphs in Fig. 4-20.

Uniqueness of Dual Graphs:

1. Is a geometric dual of a graph unique?
2. Are all duals of a given graph isomorphic?

A planar graph G will have a unique dual if and only if it has a unique plane representation or unique embedding on a sphere.

The graphs in Fig. 4-25 are 2-isomorphic.

Theorem 4-17

All duals of a planar graph G are 2-isomorphic; and every graph 2-isomorphic to a dual of G is also a dual of G .

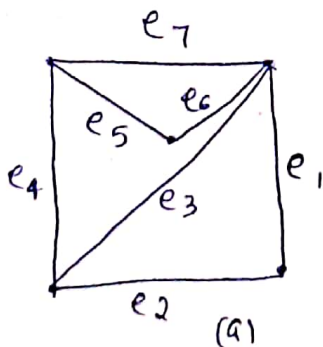
- * All duals of a 3-connected graph are isomorphic.

COMBINATORIAL DUAL

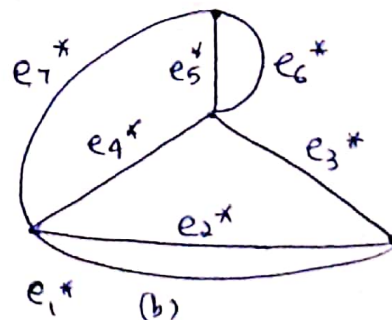
Two planar graphs G and G^* are said to be duals or combinatorial duals of each other if there is a one-to-one correspondence between the edges of G and G^* such that if g is any subgraph of G and h is the corresponding subgraph of G^* , then

$$\text{rank of } (G^* - h) = \text{rank of } G^* - \text{nullity of } g.$$

Eg:



(12)



Consider the above graph (a) and its dual in (b). Take the subgraph $\{e_4, e_5, e_6, e_7\}$ in (a) and the corresponding subgraph $\{e_4^*, e_5^*, e_6^*, e_7^*\}$ in Fig (b).

$$\text{rank of } (G^* - \{e_4^*, e_5^*, e_6^*, e_7^*\}) = \text{rank of } \{e_1^*, e_2^*, e_3^*\} \\ = 2,$$

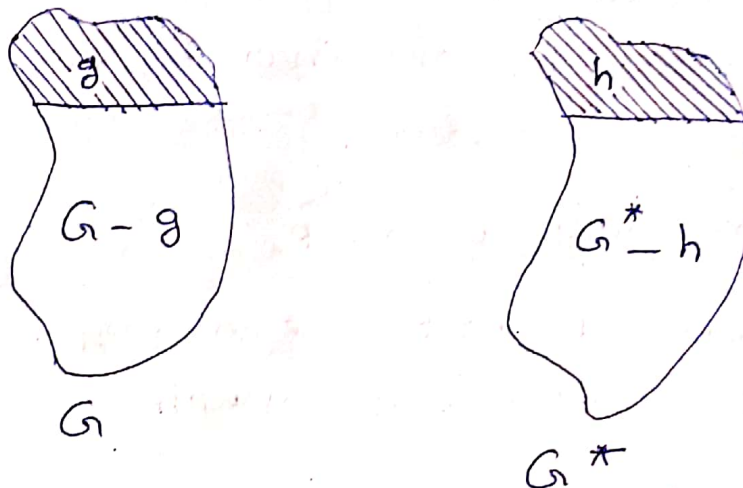
$$\text{rank of } G^* = 3,$$

$$\text{nullity of } \{e_4, e_5, e_6, e_7\} = 1,$$

$$\text{and } 2 = 3 - 1.$$

Self-Dual Graphs:

If a planar graph G is isomorphic to its own dual, it is called a self-dual graph.



$$\text{Rank of } (G^* - h) = \text{Rank of } G^* - \text{nullity of } g$$

Fig. 4-26 Combinatorial duals.

Dual of a Subgraph:

1. Let G be a planar graph and G^* be its dual.
 2. Let a be an edge in G , a^* in G^* .
 3. Delete a from G , find dual of $G-a$.
 - a) If edge a was on the boundary of two regions,
 - merge two regions into one.
 - b) $(G-a)^*$ can be obtained from G^* by deleting a^* and then fusing end vertices of a^* in $G^* - a^*$.
 - c) If edge a is not on the boundary, a^* forms a self-loop.
- $G^* - a^*$ is the same as $(G-a)^*$.

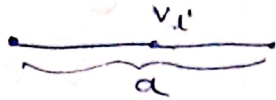
Thus if a graph G has a dual G^* , the dual of any subgraph of G can be obtained by successive application of this procedure.

Dual of a Homeomorphic Graph:

Two graphs are said to be homeomorphic if both can be obtained from the same graph by subdivisions of edges.

1. Let G be a planar graph and G^* be its dual.
2. Let a be an edge in G , and the corresponding edge in G^* be a^* .

3. Add a vertex of degree two in edge a .



How will it affect the dual?

4. It will add an edge parallel to a^* in G^* .
 5. Merging of two edges in series will eliminate one of the corresponding parallel edges in G^* .

Thus if a graph G has a dual G^* , the dual of any graph homeomorphic to G can be obtained from G^* by the above procedure.

Theorem 4-18

A necessary and sufficient condition for two planar graphs G_1 and G_2 to be duals of each other is as follows:

There is a one-to-one correspondence between the edges in G_1 and the edges in G_2 such that a set of edges in G_1 forms a circuit if and only if the corresponding set in G_2 forms a cut-set.

Proof:

Let us consider a plane representation of a planar graph G . Let us also draw (geometrically) a dual G^* of G . Then consider an arbitrary circuit Γ in G . Clearly, Γ will form some closed simple curve in

the plane representation of G - dividing the plane into two areas. Thus the vertices of G^* are partitioned into two nonempty, mutually exclusive subsets - one inside Γ and the other outside. In other words, the set of edges Γ^* in G^* corresponding to the set Γ in G is a cut-set in G^* . (No proper subset of Γ^* will be a cut-set in G^* ; why?). Likewise it is apparent that corresponding to a cut-set S^* in G^* there is a unique circuit consisting of the corresponding edge-set S in G such that S is a circuit. This proves necessity.

To prove the sufficiency, let G be a planar graph and let G' be a graph for which there is a one-to-one correspondence between the cut-sets of G and circuits of G' , and vice versa. Let G^* be a dual graph of G . There is a one-to-one correspondence between the circuits of G' and cut-sets of G , and also between the cut-sets of G and circuits of G^* . Therefore there is a one-to-one correspondence between the circuits of G' and G^* , implying that G' and G^* are 2-isomorphic. So G' must be a dual of G because "All duals of a planar graph G are 2-isomorphic, and every graph 2-isomorphic to a dual of G is also a dual of G ."

Theorem 4-19

A graph has a dual if and only if it is planar.

Proof:

We need to prove just the "only if" part. That is, we have only to prove that a nonplanar graph does not have a dual. Let G be a nonplanar graph. Then according to Kuratowski's theorem, G contains K_5 or $K_{3,3}$ or a graph homeomorphic to either of these. We have already seen that a graph G can have a dual only if every subgraph g of G and every graph homeomorphic to g has a dual. Thus if we can show that neither K_5 nor $K_{3,3}$ has a dual, we have proved the theorem. This we shall prove by contradiction as follows:

- (a) suppose that $K_{3,3}$ has a dual D ; observe that the cut-sets in $K_{3,3}$ correspond to circuits in D and vice versa. Since $K_{3,3}$ has no cut-set consisting of two edges, D has no circuit consisting of two edges. That is, D contains no pair of parallel edges. Since every circuit in $K_{3,3}$ is of length four or six, D has no cut-set with less than four edges. Therefore, the degree of every vertex in D is at least four. As D has no parallel edges and the degree of every vertex is at least four, D must have at least

five vertices each of degree four or more. That is D must have at least $(5 \times 4)/2 = 10$ edges. This is a contradiction, because $K_{3,3}$ has nine edges and so must its dual. Thus $K_{3,3}$ cannot have a dual. Likewise,

b) Suppose that the graph K_5 has a dual H . Note that K_5 has

- (1) 10 edges,
- (2) no pair of parallel edges,
- (3) no cut-set with two edges, and
- (4) cut-sets with only four or six edges.

consequently, graph H must have

- (1) 10 edges,
- (2) no vertex with degree less than three,
- (3) no pair of parallel edges, and
- (4) circuits of length four and six only.

Now graph H contains a hexagon and no more than three edges can be added to a hexagon without creating a circuit of length three or a pair of parallel edges. Since both of these are forbidden in H and H has 10 edges, there must be at least seven vertices in H . The degree of each of these vertices is at least three. This leads to H having at least 11 edges. A contradiction.

Module V

Matrix Representation of graphs

Pictorial representation of a graph is very convenient for a visual study.

A matrix is a convenient and useful way of representing a graph to a computer.

1. ADJACENCY MATRIX (or CONNECTION MATRIX).

The adjacency matrix of a graph G with n vertices and no parallel edges is an $n \times n$ symmetric binary matrix $X = [x_{ij}]$ defined over the ring of integers such that

$x_{ij} = 1$, if there is an edge between i th and j th vertices, and

$= 0$, if there is no edge between them.

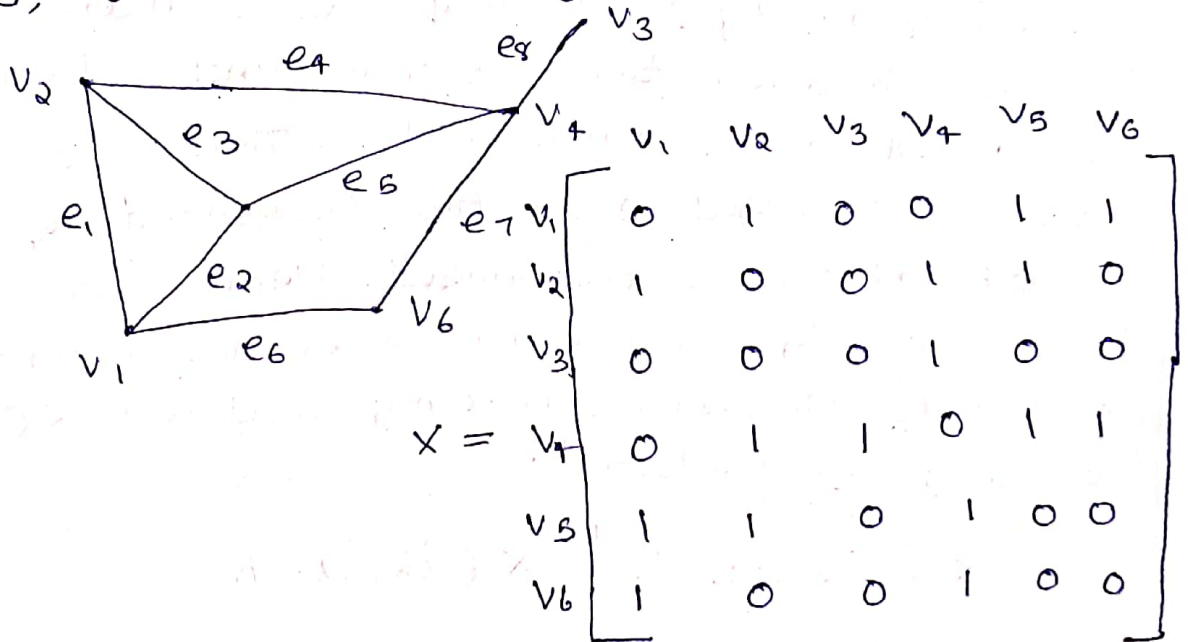


Fig. 5-1 simple graph and its adjacency matrix.

Observations about the adjacency matrix X of G are

1. The entries along the principal diagonal of X are all 0's if and only if the graph has no self-loops. A self-loop at the i th vertex corresponds to $x_{ii} = 1$.
2. The definition of adjacency matrix makes no provision for parallel edges. This is why the adjacency matrix X was defined for graphs without parallel edges.
3. If the graph has no self-loops (and no parallel edges, of course), the degree of a vertex equals the number of 1's in the corresponding row or column of X .
4. Permutations of rows and of the corresponding columns imply reordering the vertices. It must be noted, however, that the rows and columns must be arranged in the same order. Thus, if two rows are interchanged in X , the corresponding columns must also be interchanged. Hence two graphs G_1 and G_2 with no parallel edges are isomorphic if and only if their adjacency matrices $X(G_1)$ and $X(G_2)$ are related:

$$X(G_2) = R^{-1} \cdot X(G_1) \cdot R,$$

where R is a permutation matrix.

②

5. A graph G is disconnected and is in two components g_1 and g_2 and only if its adjacency matrix $X(G)$ can be partitioned as

$$X(G) = \begin{bmatrix} X(g_1) & 0 \\ 0 & X(g_2) \end{bmatrix}$$

where $X(g_1)$ is the adjacency matrix of the component g_1 and $X(g_2)$ is that of the component g_2 .

Given any square, symmetric, binary matrix Q of order n , one can always construct a graph G of n vertices (and no parallel edges) such that Q is the adjacency matrix of G .

Powers of X

Let us multiply by itself the 6 by 6 adjacency matrix of the simple graph in Fig. 5-1. The result, another 6 by 6 symmetric matrix X^2 , is shown below

$$X^2 = \begin{bmatrix} 3 & 1 & 0 & 3 & 1 & 0 \\ 1 & 3 & 1 & 1 & 2 & 2 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 3 & 1 & 0 & 4 & 1 & 0 \\ 1 & 2 & 1 & 1 & 3 & 2 \\ 0 & 2 & 1 & 0 & 2 & 2 \end{bmatrix}$$

(3)

VECTOR SPACE ASSOCIATED WITH A GRAPH

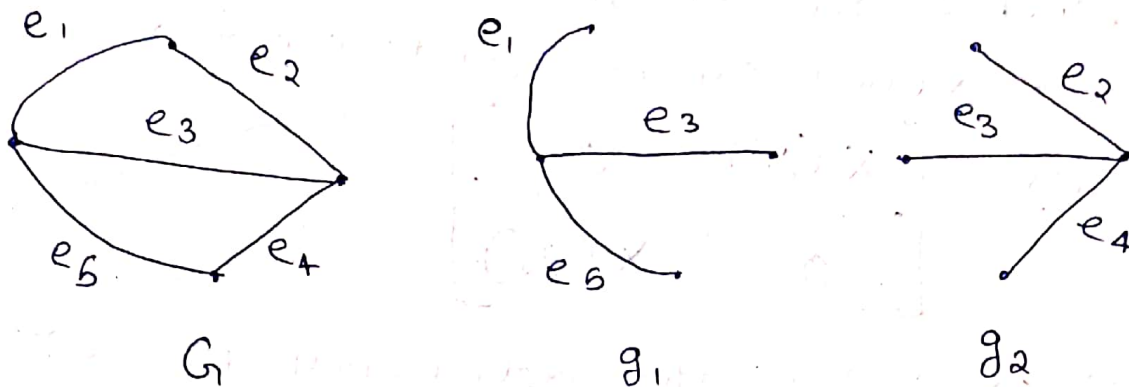


Fig. 5-2 Graph and two of its subgraphs.

Consider the graph G in Fig. 5-2 with four vertices and five edges e_1, e_2, e_3, e_4, e_5 . Any subset of these five edges (i.e., any subgraph g) of G can be represented by a 5-tuple.

$$X = (x_1, x_2, x_3, x_4, x_5)$$

Such that

$$x_i = 1 \text{ if } e_i \text{ is in } g \text{ and}$$

$$x_i = 0 \text{ if } e_i \text{ is not in } g.$$

For instance, the subgraph g_1 in Fig. 5-2 will be represented by $(1, 0, 1, 0, 1)$.

Altogether there are 2^5 or 32 such 5-tuples possible, including the zero vector

$$0 = (0, 0, 0, 0, 0) \text{ - null graph,}$$

$$\text{the graph } G \text{ } (1, 1, 1, 1, 1).$$

There is a vector space W_G associated with every graph G , and this vector space consists of

1. G 's field modulo 2; ($G F(2)$)

that is

with operation addition modulo 2 written as $+$

such that

$$0 + 0 = 0,$$

$$1 + 0 = 1 = 0 + 1$$

$$1 + 1 = 0$$

and multiplication modulo 2 written as \cdot .

such that

$$0 \cdot 0 = 0 = 1 \cdot 0 = 0 \cdot 1$$

$$\text{and } 1 \cdot 1 = 1$$

2. 2^e vectors (e -tuples), where e is the number of edges in G .

3. An addition operation between two vectors X, Y in this space, defined as the vector sum

$$X \oplus Y = (x_1 + y_1, x_2 + y_2, \dots, x_e + y_e),$$

$+$ being addition modulo 2.

4. A scalar multiplication between a scalar c in \mathbb{Z}_2 and a vector X , defined as $c \cdot X = (c \cdot x_1, \dots, c \cdot x_e)$.

The identity element (for the vector sum operation)

in a vector space is 0 , the zero vector.

Linear Dependence

A set of vectors x_1, x_2, \dots, x_r (over some field F) is said to be linearly independent if for scalars c_1, c_2, \dots, c_r , in F the expression

$$c_1 x_1 + c_2 x_2 + \dots + c_r x_r = 0$$

holds only if $c_1 = c_2 = \dots = c_r = 0$.

Otherwise, the set of vectors is said to be linearly dependent.

2. INCIDENCE MATRIX

Let G be a graph with n vertices, e edges, and no self-loops. Define an n by e matrix $A = [a_{ij}]$, whose n rows correspond to the n vertices and the e columns correspond to the e edges, as follows:

The matrix element

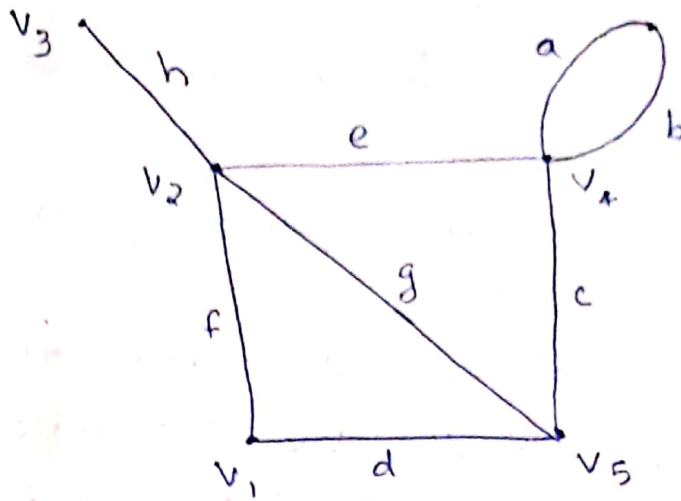
$a_{ij} = 1$, if j th edge e_j is incident on i th vertex v_i , and $a_{ij} = 0$, otherwise.

Such a matrix A is called the vertex-edge incidence matrix, or simply incidence matrix.

Binary matrix

The incidence matrix contains only two elements 0 and 1. Such a matrix is called a binary matrix or a $(0, 1)$ -matrix. The elements 0, 1 are from Galois's field modulo 2.

⑥.



(a)

	a	b	c	d	e	f	g	h
v_1	0	0	0	1	0	1	0	0
v_2	0	0	0	0	1	1	1	1
v_3	0	0	0	0	0	0	0	1
v_4	1	1	1	0	1	0	0	0
v_5	0	0	1	1	0	0	1	0
v_6	1	1	0	0	0	0	0	0

(b)

Fig. 5-3 Graph and its incidence matrix.

Observations about the incidence matrix A

1. Since every edge is incident on exactly two vertices, each column of A has exactly two 1's.
2. The number of 1's in each row equals the

⊗ (7)

degree of the corresponding vertex

3. A row with all 0's, therefore, represents an isolated vertex.
4. Parallel edges in a graph produce identical columns in its incidence matrix.
5. If a graph G is disconnected and consists of two components g_1 and g_2 , the incidence matrix $A(G)$ of graph G can be written in a block-diagonal form as

$$A(G) = \begin{bmatrix} A(g_1) & 0 \\ 0 & A(g_2) \end{bmatrix}$$

where $A(g_1)$ and $A(g_2)$ are the incidence matrices of components g_1 and g_2 . This observation results from the fact that no edge in g_1 is incident on vertices of g_2 , and vice versa. This remark is also true for a disconnected graph with any number of components.

6. Permutation of any two rows or columns in an incidence matrix simply corresponds to relabeling the vertices and edges of the same graph.

THEOREM 5-1

Two graphs G_1 and G_2 are isomorphic if and only if their incidence matrices $A(G_1)$ and $A(G_2)$ differ only by permutations of rows and columns.

Rank of the Incidence Matrix

Theorem 5-2

If $A(G)$ is an incidence matrix of a connected graph G with n vertices, the rank of $A(G)$ is $n-1$.

Proof:

Each row in an incidence matrix $A(G)$ may be regarded as a vector over $GF(2)$ in the vector space of graph G . Let the vector in the first row be called A_1 , in the second row A_2 , and soon. Thus

$$A(G) = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} \quad (5-1)$$

Since there are exactly two 1's in every column of A , the sum of all these vectors is 0 (this being a modulo 2 sum of the corresponding entries). Thus vectors A_1, A_2, \dots, A_n are not linearly independent. Therefore, the rank of A is less than n ; that is, $\text{rank } A \leq n-1$.

Now consider the sum of any m of these n vectors ($m \leq n-1$). If the graph is connected, $A(G)$ cannot be partitioned, as in Eq. (5-1), such that $A(G_1)$ is with m rows and $A(G_2)$ with $n-m$ rows, that is no m by m submatrix of $A(G)$ can be found, for $m \leq n-1$, such that the modulo 2 sum of those m rows is equal to zero.

⊗ (9)

Since there are only two constants 0 and 1 in this field, the additions of all vectors taken m at a time for $m = 1, 2, \dots, n-1$ exhausts all possible linear combinations of $n-1$ row vectors. Thus we have just shown that no linear combination of m row vectors of A (for $m \leq n-1$) can be equal to zero. Therefore, the rank of $A(G)$ must be at least $n-1$.

Since the rank of $A(G)$ is no more than $n-1$ and is no less than $n-1$, it must be exactly equal to $n-1$. Hence the theorem.

- The rank of $A(G)$ is $n-k$, if G is a disconnected graph with n vertices and k components.

Reduced incidence matrix

Let A be the incidence matrix of a connected graph G and A_f be the matrix got by removing any one row from A , which is an $(n-1) \times e$ submatrix of A is called a reduced incidence matrix. The vertex corresponding to the deleted row in A_f is called the reference vertex.

- Any vertex of a connected graph can be made the reference vertex.

Singular matrix

Singular matrix is square matrix whose determinant is equal to zero.



Non-Singular Matrix

Non-singular matrix is also square matrix whose determinant is not equal to zero.

* The reduced incidence matrix of a tree is nonsingular.

3. CIRCUIT MATRIX

Let the number of different circuits in a graph G be q , and the number of edges in G be e . Then a circuit matrix $B = [b_{ij}]$ of G is a q by e , $(0,1)$ -matrix defined as follows:

$b_{ij} = 1$, if i th circuit includes j th edge, and
 $= 0$, otherwise.

The graph in Fig. 5-3(a) has four different circuits, $\{a, b\}$, $\{c, e, g\}$, $\{d, f, g\}$, and $\{c, d, f, e\}$. Therefore, its circuit matrix is a 4 by 8, $(0,1)$ -matrix as shown:

$$B(G) = \begin{matrix} & \begin{matrix} a & b & c & d & e & f & g & h \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \end{matrix}$$

Observations about a circuit matrix $B(G)$ of a graph G :

1. A column of all zeros corresponds to a noncircuit

edge (i.e., an edge that does not belong to any circuit).

2. Each row of $B(G)$ is a circuit vector.
3. Unlike the incidence matrix, a circuit matrix is capable of representing a self-loop - the corresponding row will have a single 1.
4. The number of 1's in a row is equal to the number of edges in the corresponding circuit.
5. If graph G is separable (or disconnected) and consists of two blocks (or components) g_1 and g_2 , the circuit matrix $B(G)$ can be written in a block-diagonal form as

$$B(G) = \begin{bmatrix} B(g_1) & 0 \\ 0 & B(g_2) \end{bmatrix},$$

where $B(g_1)$ and $B(g_2)$ are the circuit matrices of g_1 and g_2 . This observation results from the fact that circuits in g_1 have no edges belonging to g_2 , and vice versa.

6. Permutation of any two rows or columns in a circuit matrix simply corresponds to relabeling the circuits and edges.
7. Two graphs G_1 and G_2 will have the same circuit matrix if and only if G_1 and G_2 are 2-isomorphic.

(12)

Theorem 5-3

Let B and A be, respectively, the circuit matrix and the incidence matrix (of a self-loop-free graph) whose columns are arranged using the same order of edges. Then every row of B is orthogonal to every row of A ; that is

$$A \cdot B^T = B \cdot A^T = 0 \pmod{2},$$

where superscript T denotes the transposed matrix.

proof:

Consider a vertex v and a circuit Γ in the graph G . Either v is in Γ or it is not. If v is not in Γ , there is no edge in the circuit Γ that is incident on v . On the other hand, if v is in Γ , the number of those edges in the circuit Γ that are incident on v is exactly two.

With this remark in mind, consider the i 'th row in A and the j 'th row in B . Since the edges are arranged in the same order, the nonzero entries in the corresponding positions occur only if the particular edge is incident on the i 'th vertex and is also in the j 'th circuit.

If the i 'th vertex is not in the j 'th circuit, there is no such nonzero entry, and the dot product of the two rows is zero. If the i 'th vertex is in the j 'th circuit, there will be exactly two 1's in the sum of the products of individual entries. Since $1+1=0 \pmod{2}$, the dot product of the two arbitrary rows - one from A

(X) (13)

and the other from B - is zero. Hence the theorem.

Eg:

Let us multiply the incidence matrix and transposed circuit of the graph in Fig. 5-3(a), after making sure that the edges are in the same order in both.

$$A \cdot B^T = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \pmod{2}.$$

A. FUNDAMENTAL CIRCUIT MATRIX AND RANK OF B

A circuit, formed by adding a chord to a spanning tree, is called a fundamental circuit.

In a circuit matrix, if we retain only those rows that correspond to a set of fundamental circuits and remove all other rows, we would not lose any information.

A submatrix (of a circuit matrix) in which all rows correspond to a set of fundamental circuits is called a fundamental circuit matrix B_f .

If n is the number of vertices and e the number of edges in a connected graph, then B_f is an $(e-n+1)$ by e matrix, because the number of fundamental circuits is $e-n+1$, each fundamental circuit being produced by one chord.

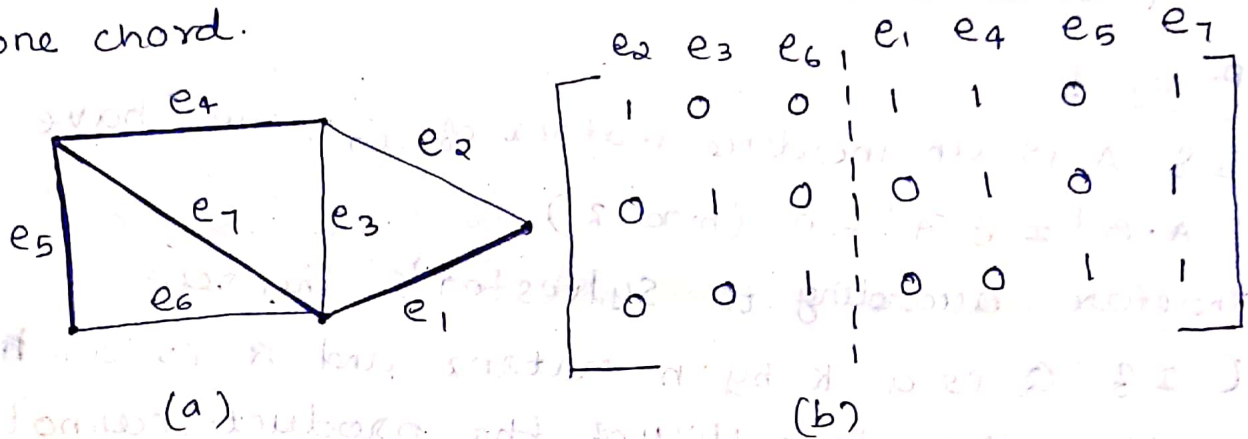


Fig. 5-4 Graph and its fundamental circuit matrix (with respect to the spanning tree shown in heavy lines).

A matrix B_f arranged can be written as

$$B_f = [I_u \mid B_t]$$

where,

I_u is an identity matrix of order $u = e - n + 1$, all the $e - n + 1$ chords correspond to the first $e - n + 1$ columns. and

B_t is the remaining u by $(n-1)$ submatrix. Rearrange the rows such that the first row corresponds to the fundamental circuit made by the chord in the first column, the second row to the fundamental circuit made by the second, and so on.

Theorem 5-1

If B is a circuit matrix of a connected graph G with e edges and n vertices,
 $\text{rank of } B = e - n + 1$.

Proof: 1

If A is an incidence matrix of G , we have

$$A \cdot B^T = B \cdot A^T = 0 \pmod{2}.$$

Therefore, according to Sylvester's theorem

(If Q is a k by n matrix and R is an n by p matrix, then the nullity of the product cannot exceed the sum of the nullities of the factors, that is, $\text{nullity of } QR \leq \text{nullity of } Q + \text{nullity of } R$).

$$\text{rank of } A + \text{rank of } B \leq e;$$

that is,

$$\text{rank of } B \leq e - \text{rank of } A.$$

$$\text{Since rank of } A = n - 1$$

$$\text{we have rank of } B \leq e - n + 1.$$

$$\text{But rank of } B \geq e - n + 1.$$

Therefore, we must have

$$\text{rank of } B = e - n + 1.$$

Proof: 2

Consider the circuit subspace W_F in the vector space W_G of a graph.

Every row in circuit matrix B is a vector in W_F ,

and since the rank of any matrix is equal to the number of linearly independent rows (or columns) in the matrix, we have,

rank of matrix B = number of linearly independent rows in B ;

but the number of linearly independent rows in $B \leq$ number of linearly independent vectors in W_F , and the number of linearly independent vectors in $W_F =$ dimension of $W_F = n$. Therefore, rank of $B \leq e - n + 1$.

Since B_f is a submatrix of the circuit matrix B , the rank of $B \geq e - n + 1$.

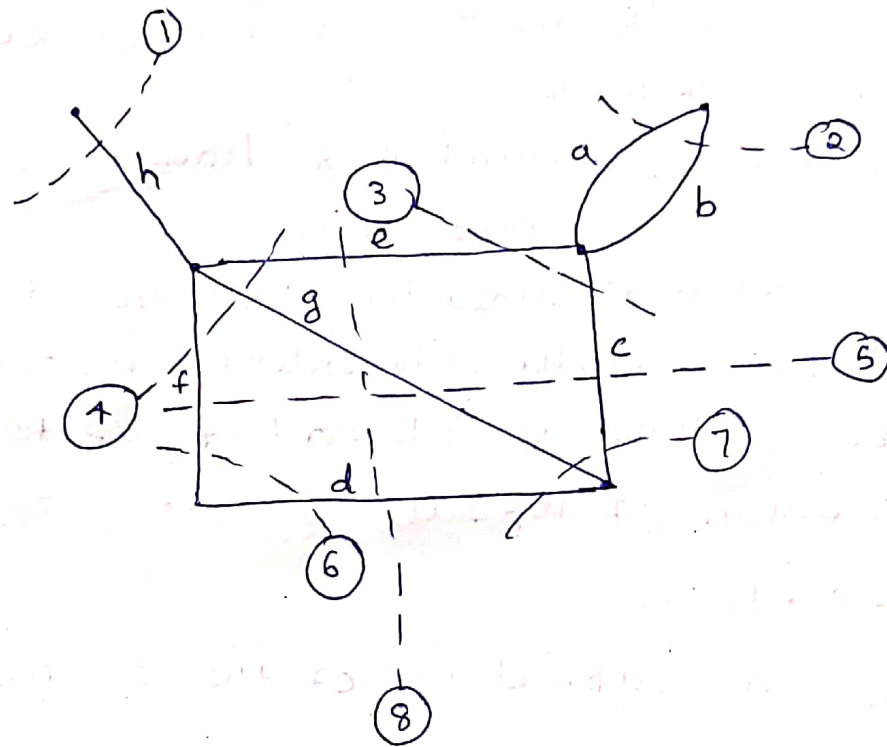
- If G is a disconnected graph with k components, e edges, and n vertices,

$$\text{rank of } B = n = e - n + k.$$

5. CUT-SET MATRIX

A cut-set matrix $C = [c_{ij}]$ in which the rows correspond to the cut-sets and the columns to the edges of the graph, as follows:

$$c_{ij} = 1, \text{ if } i\text{th cut-set contains } j\text{th edge, and} \\ = 0, \text{ otherwise.}$$



$$C = \begin{bmatrix} & a & b & c & d & e & f & g & h \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 6 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 7 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 8 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Fig. 5-5 Graph and its cut-set matrix.

Observations can be made from a cut-set matrix $C(G)$ of a graph G .

1. A permutation of rows or columns in a cut-set matrix corresponds simply to a renaming of the cut-sets and edges, respectively.
2. Each row in $C(G)$ is a cut-set vector.
3. A column with all 0's corresponds to an edge forming a self-loop.
4. Parallel edges produce identical columns in the cut-set matrix.
5. For a nonseparable graph G , $C(G)$ contains incidence matrix $A(G)$.

For a separable graph, the incidence matrix of each block is contained in the cut-set matrix.

6. $\text{rank of } C(G) \geq \text{rank of } A(G)$.

Hence, for a connected graph of n vertices,

$$\text{rank of } C(G) \geq n-1. \quad (5-2)$$

7. Since the number of edges common to a cut-set and a circuit is always even, every row in C is orthogonal to every row in B , provided the edges in both B and C are arranged in the same order. In other words,

$$B \cdot C^T = C \cdot B^T = 0 \pmod{2}. \quad (5-3)$$

on applying $\text{rank of } B + \text{rank of } C \leq e$ to Eq. (5-3),

(19)

and since for a connected graph

$$\text{rank of } B = e - n + 1,$$

$$\text{rank of } C \leq n - 1.$$

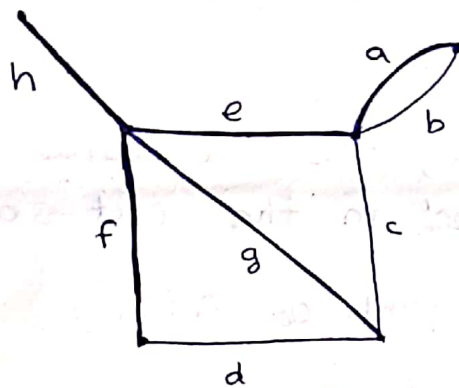
(5-4)

Combining Eqs. (5-2) and (5-4),

$$\text{rank of } C = n - 1.$$

Theorem 5-5

The rank of cut-set matrix $C(G)$ is equal to the rank of the incidence matrix $A(G)$, which equals the rank of graph G .



$$C_f = \begin{bmatrix} & b & c & d & | & a & e & f & g & h \\ \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{matrix} & | & \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{matrix} \end{bmatrix} = [C_c \mid I_5]$$

Fig. 5-6 Spanning tree in a graph and the corresponding fundamental cut-set matrix.

1. The cut-set matrix generally has many redundant (or linearly dependent) rows.
2. Fundamental cut-set matrix C_f

A fundamental cut-set matrix C_f (of a connected graph G with e edges and n vertices) is an $(n-1) \times e$ submatrix of C such that the rows correspond to the set of fundamental cut-sets with respect to some spanning tree.

3. A fundamental cut-set matrix C_f can also be partitioned into two submatrices, one of which is an identity matrix I_{n-1} of order $n-1$. That is,

$$C_f = [C_c \mid I_{n-1}],$$

where

the last $n-1$ columns forming the identity matrix correspond to the $n-1$ branches of the spanning tree, and

the first $e - n + 1$ columns forming C_c correspond to the chords.

RELATIONSHIP AMONG A_f , B_f and C_f

Let A_f be the reduced incidence matrix A_f , the fundamental circuit matrix B_f , and the fundamental cut-set matrix C_f of a connected graph.

It has been shown that

$$B_f = [I_u \mid B_t],$$

$$C_f = [C_c \mid I_{n-1}].$$

Where subscript t denotes the submatrix corresponding to the branches of a spanning tree, and subscript c denotes the submatrix corresponding to the chords.

1. Given A or A_f , we can readily construct B_f and C_f , starting from an arbitrary spanning tree and its subgraph A_t in A_f .
2. Given either B_f or C_f , we can construct the other. Thus since B_f determines a graph within 2-isomorphism so does C_f .
3. Given either B_f or C_f , A_f in general cannot be determined completely.

6. PATH MATRIX

Used in communication and transportation networks, is the

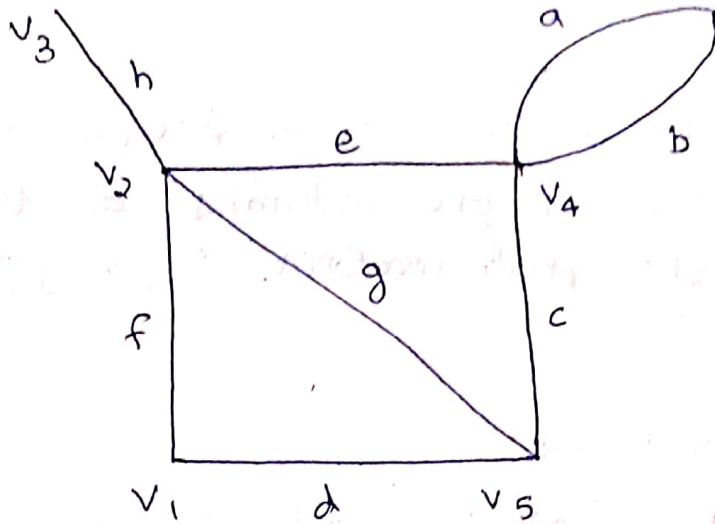
A path matrix is defined for a specific pair of vertices in a graph, say (x, y) , and is written as $P(x, y)$.

The rows in $P(x, y)$ correspond to different paths between vertices x and y , and

The columns in $P(x, y)$ correspond to the edges in G . That is, the path matrix for (x, y) vertices is $P(x, y) = [P_{ij}]$,

where

$P_{ij} = 1$, if j th edge lies in i th path, and
 $= 0$, otherwise.



Consider all paths between vertices v_3 and v_4 . There are three different paths; $\{h, e\}$, $\{h, g, c\}$, and $\{h, f, d, c\}$. Let us number them 1, 2, and 3, respectively. Then we get the 3 by 8 path matrix $P(v_3, v_4)$:

$$P(v_3, v_4) = \begin{matrix} & \begin{matrix} a & b & c & d & e & f & g & h \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \end{matrix}$$

Observations

1. A column of all 0's corresponds to an edge that does not lie in any path between x and y .
2. A column of all 1's corresponds to an edge that lies in every path between x and y .
3. There is no row with all 0's.

4. The ring sum of any two rows in $P(x, y)$ corresponds to a circuit or an edge-disjoint union of circuits.

Theorem 5-6

If the edges of a connected graph are arranged in the same order for the columns of the incidence matrix A and the path matrix $P(x, y)$, then the product (mod 2)

$$A \cdot P^T(x, y) = M$$

where the matrix M has 1's in two rows x and y , and the rest of the $n-2$ rows are all 0's.

Eg:

$$A \cdot P^T(v_3, v_4) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{matrix} & 1 & 2 & 3 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{matrix} \pmod{2}.$$

(24)

A path matrix contains less information about the graph in general than any of the matrices A , B or C does.

Relationship Between Incidence Matrix $A(G)$ and Adjacency matrix $X(G)$

1. If a graph G has no self-loops, its $A(G)$ contains all the information about G .
2. If G has no parallel edges, its adjacency matrix $X(G)$ contains all the information about G .
3. If a graph G has neither self-loops nor parallel edges (i.e., G is a simple graph), both $A(G)$ and $X(G)$ contain the entire information.

Module VI

GRAPHS THEORETIC ALGORITHMS

Most of the practical problems which call for graph theory involve large graphs - graphs that are virtually impossible for hand computation.

computer programs have been written to handle successfully large graphs encountered in PERT, flow problems, transportation networks, electrical networks, circuit layouts, and the like.

The power of the computer must be combined with the ingenuity of mathematical techniques.

ALGORITHMS

Algorithm consists of a set of instructions that when followed step by step will lead to the solutions of the problem.

1) Every algorithm must have five important features:

1. Finiteness,
2. Definiteness,
3. Input,
4. Output, and
5. Effectiveness.

2) Forms of algorithm

- 1) the steps may be written in English;
- 2) It may be in the form of a computer program written in complete detail in the language understandable by the machine in use; or
- 3) the algorithm may be expressed in a form between

①

these two extremes, such as a flow chart.

Usually, an algorithm is first expressed in ordinary language, then converted into a flow chart, and finally written in the detailed and precise language so that a machine can execute it.

3) Efficiency of Algorithms:

Criteria for efficiency of an algorithm includes

1) the memory and

2) computation-time, requirements as a function of the size of the input.

Here,

The input — is a graph

The size — is the number of vertices, n , and the number of edges, e .

For most graph problems the memory requirement is generally not the bottleneck, but the computation time can be.

Execution time.

The time taken by the system to execute.

a) "worst-case" execution time

The time taken for the worst possible choice of a graph of the given size,

b) "best-case" execution time

The time taken for the best possible choice of a graph of the given size,

c) "average-case" execution time

The time taken for the average possible choice of a graph of the given size.

(2)

INPUT: COMPUTER REPRESENTATION OF A GRAPH

Input: the data with which the algorithm begins.

Types of graph presentation to a digital computer includes

a) Adjacency Matrix.

After assigning a distinct number to each of the n vertices of the given graph (or digraph) G , the n by n binary matrix $X(G)$ is used for representing G during input, storage, and output.

Since each of the n^2 entries is either a zero or a one, the adjacency matrix requires n^2 bits of computer memory.

Bits can be packed into words.

Let w be the word length (i.e., the number of bits in a computer word) and

Let n be the number of vertices in the graph.

Then each row contains $\lceil n/w \rceil$ machine words.

($\lceil x \rceil$ denotes the smallest integer not less than x .)

The no. of words required to store the adjacency matrix is, therefore, $n \lceil n/w \rceil$.

The adjacency matrix of an undirected graph is symmetric and therefore storing only the upper triangle is sufficient. This requires only $n(n-1)/2$ bits of storage.

Adjacency matrix is defined for graphs without parallel edges. It is not possible to represent parallel edges in an adjacency matrix.

b) Incidence Matrix:

An incidence matrix requires $n \cdot e$ bits of storage, which might be more than the n^2 bits needed for an adjacency matrix, because the number of edges e is usually greater than the number of vertices n .

Incidence matrices are particularly favored for electrical networks and switching networks.

c) Edge Listing:

To list all edges of the graph as vertex pairs, having numbered the n vertices in some arbitrary order.

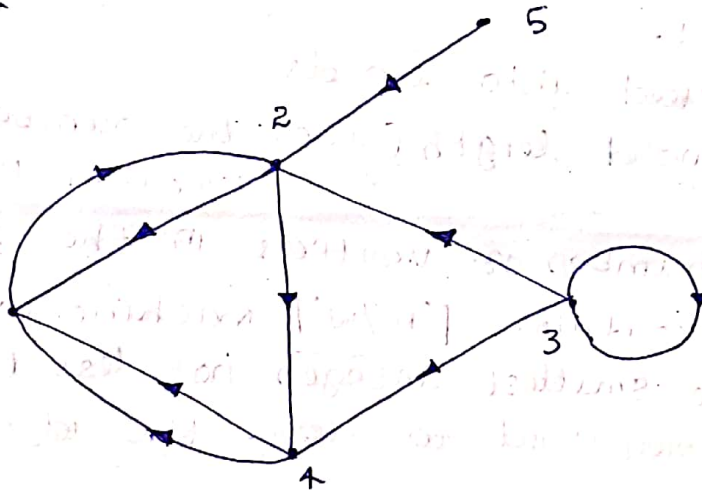


Fig. 6.1 A digraph.

eg: the digraph in Fig 6.1 would appear as a set of the following ordered pairs:

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

— parallel edges and self-loops can be included in Edge listing of a graph or digraph.

(4)

The number of bits required to label (1 through n) each vertex is b , where

$$2^{b-1} < n \leq 2^b.$$

And each of the e edges requires storing two such numbers, the total storage required is

$$2e.b \text{ bits.}$$

and $2e.b < n^2$.

Edge listing is a very convenient form for inputting a graph into the computer, but the storage, retrieval, and manipulation of the graph within the computer become quite difficult.

d) Two Linear Arrays:

Two linear arrays, say $F = \{f_1, f_2, \dots, f_e\}$ and $H = \{h_1, h_2, \dots, h_e\}$. Each entry in these arrays is a vertex label. The i th edge e_i is from vertex f_i to vertex h_i . G is a digraph.

Eg: Fig. 6.1 would be represented by the two arrays

$$F = (5, 2, 1, 3, 2, 4, 4, 3, 3),$$

$$H = (2, 1, 2, 2, 4, 1, 1, 4, 3).$$

The storage requirements are the same as edge listing.

e) Successor Listing:

when e/n is not large is by means of n linear arrays.

- represent each vertex k by a linear array,
- first element is k

- remaining elements are the vertices that are immediate successors of k (the vertices which have a directed path of length one from k).

The five-vertex digraph in Fig. 6-1 will appear as follows in this representation.

1: 2

2: 1, 4

3: 2, 3, 4

4: 1, 1

5: 2

For an undirected graph the neighbors (rather than the successors) of every vertex are listed.

Storage efficiency

Total storage requirement for an n -vertex graph is

$$n(1 + d_{av})$$

where

d_{av} - the average degree (out degrees in the case of a digraph) of the vertices in the graph.

successor listing is more efficient than the adjacency matrix if

$$d_{av} < \lceil \frac{n}{w} \rceil - 1,$$

w being the word length.

OUTPUT

The output will vary from problem to problem.

1. If the output consists of subgraphs, - make the program print the appropriate adjacency matrices.

2. A yes or no to the question of planarity of a given graph - ask the program to simply print YES or NO. Yes - planar representation No - Thickness of the graph.
3. For a shortest-path algorithm - print the distance (shortest) between a pair of specified vertices readily.

Algorithm 1: Connectedness and Components

The connectedness algorithm, is very basic and may serve as a subroutine in more involved graph-theoretic algorithms.

Description of the Algorithm:

Basic step: = Fusion of adjacent vertices.

1. Start with some vertex and fuse all vertices that are adjacent to it.
2. Take the fused vertex and again fuse with it all those vertices that are adjacent to it now.
3. Repeat step 2 until no more vertices can be fused.

A connected component has been "fused" to a single vertex. If this exhausts every vertex in the graph we start with a new vertex and continue the fusing operation.

Fusion in adjacency matrix

Logically add the j th row to the i th row as well as the j th column to the i th column. Then the j th row and the j th column are discarded from the matrix.

Execution time:

The upper bound on the execution time is proportional to $n(n-1)$ for n number of vertices.

where, $n-1$ number of fusions
 n logical additions.

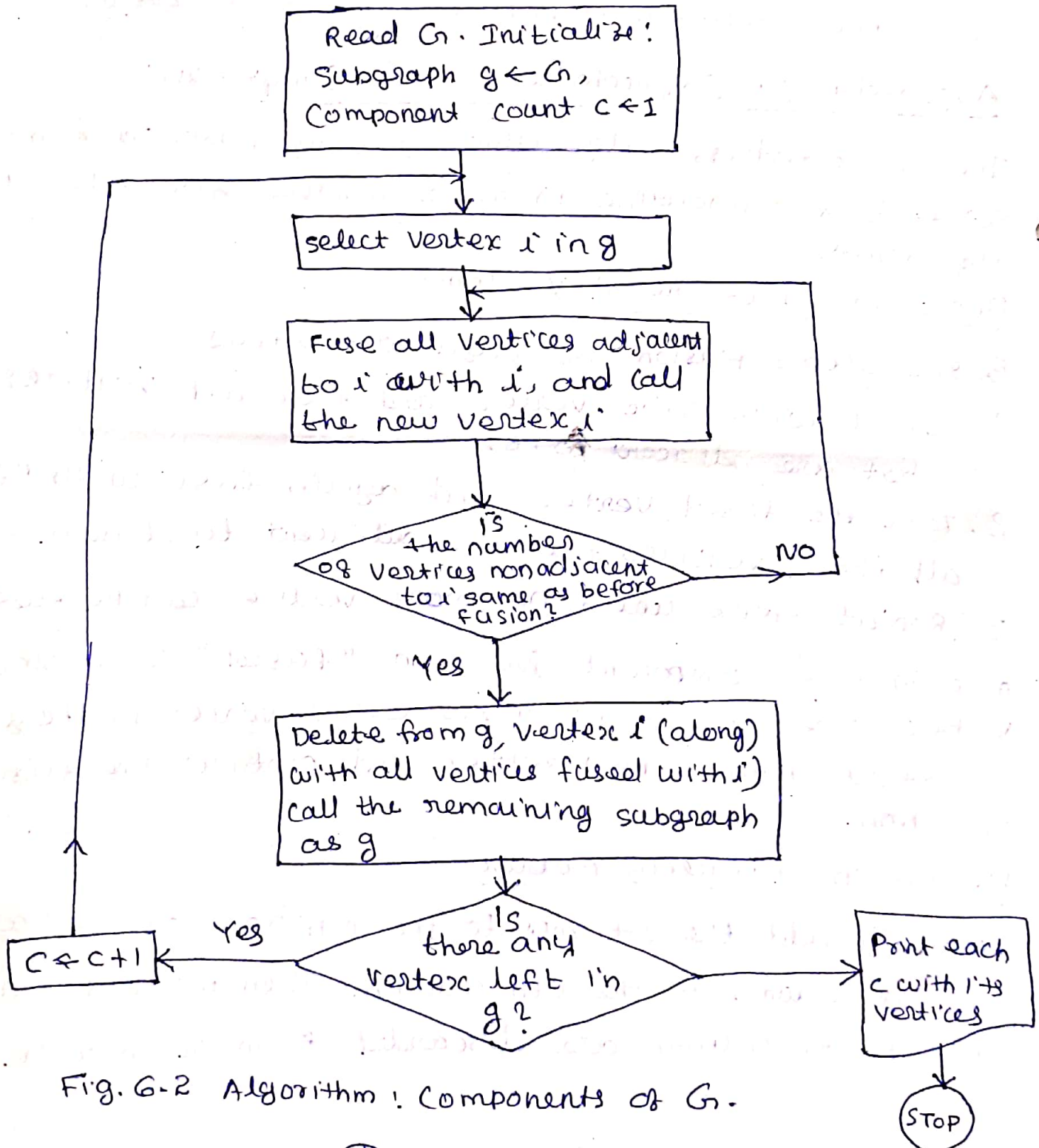


Fig. G-2 Algorithm 1: Components of G.

Algorithm : A spanning Tree

A spanning-tree algorithm yields one spanning tree in a given connected graph.

If the graph is disconnected, the algorithm should produce a spanning forest containing $n-p$ edges, where $p > 1$ is the number of components in the disconnected graph.

Spanning tree algorithm can be used to

1. Find out whether or not the graph is connected.
2. If the graph is disconnected, its components can be identified.
3. Find a spanning tree with smallest possible weight.
4. Obtain a fundamental set of circuits.

Description of the Algorithm:

Let G be a graph with

- undirected self-loop free

(if the graph has any self-loops, they may be discarded)

- n vertices and labeled $1, 2, \dots, n$
- e edges.

The graph is described by two linear arrays F and H such that $f_i \in F$ and $h_i \in H$ are the end vertices of the i th edge in G .

At each stage in the algorithm a new edge i is tested to see if either or both of its end vertices appear in any tree formed so far.

At the k th stage, $1 \leq k \leq e$, in examining the edge (f_k, h_k) five different conditions may arise:

1. If neither vertex f_k nor h_k is included in any of the trees constructed so far in G , the k th edge is named as a new tree and its end vertices f_k, h_k are given the component number c , after incrementing the value of c by 1.
2. If vertex f_k is in some tree T_i ($i=1, 2, \dots, c$) and h_k in tree T_j ($j=1, 2, \dots, c$, and $i \neq j$), the k th edge is used to join these two trees; therefore, every vertex in T_j is now given the component number of T_i . The value of c is decremented by 1.
3. If both vertices are in the same tree, the edge (f_k, h_k) forms a fundamental circuit and is not considered any further.
4. If vertex f_k is in a tree T_i and h_k is in no tree, the edge (f_k, h_k) is added to T_i by assigning the component number of T_i to h_k also.
5. If vertex f_k is in no tree and h_k is in a tree T_j , the edge (f_k, h_k) is added to T_j by assigning the component number of T_j to f_k also.

Efficiency of spanning tree Algorithm.

Depends mainly on the speed with which we can test whether or not the end vertices of the edge under consideration have occurred in any tree formed so far. For this maintain a linear array of size n [VERTEX]. Entries in the linear array include

- a) when an edge (i, j) is included in the c th tree,

the i th and j th entries in this array are set to c .

- b) When another edge (f_k, h_k) is examined, it is only necessary to check if the f_k th and the h_k th entries in array in array VERTEX are nonzero.
- c) A zero in the q th position in the array indicates that the vertex q has not so far been included in any tree.

At the end of the execution, this array VERTEX identifies the components of the graph.

Output of spanning tree algorithm.

An array of edges as the output. Let this linear array be called EDGE.

Entries in array EDGE includes:

- a) If the k th edge (in the original order in which the edges were placed) is in the c th tree,

$$EDGE(k) = c;$$

- b) Otherwise, it is zero.

All zero entries in array EDGE correspond to the chords (i.e. the edges not included in the spanning tree or forest).

The array EDGE with arrays F and H, uniquely identifies the spanning tree (or forest) generated by this algorithm.

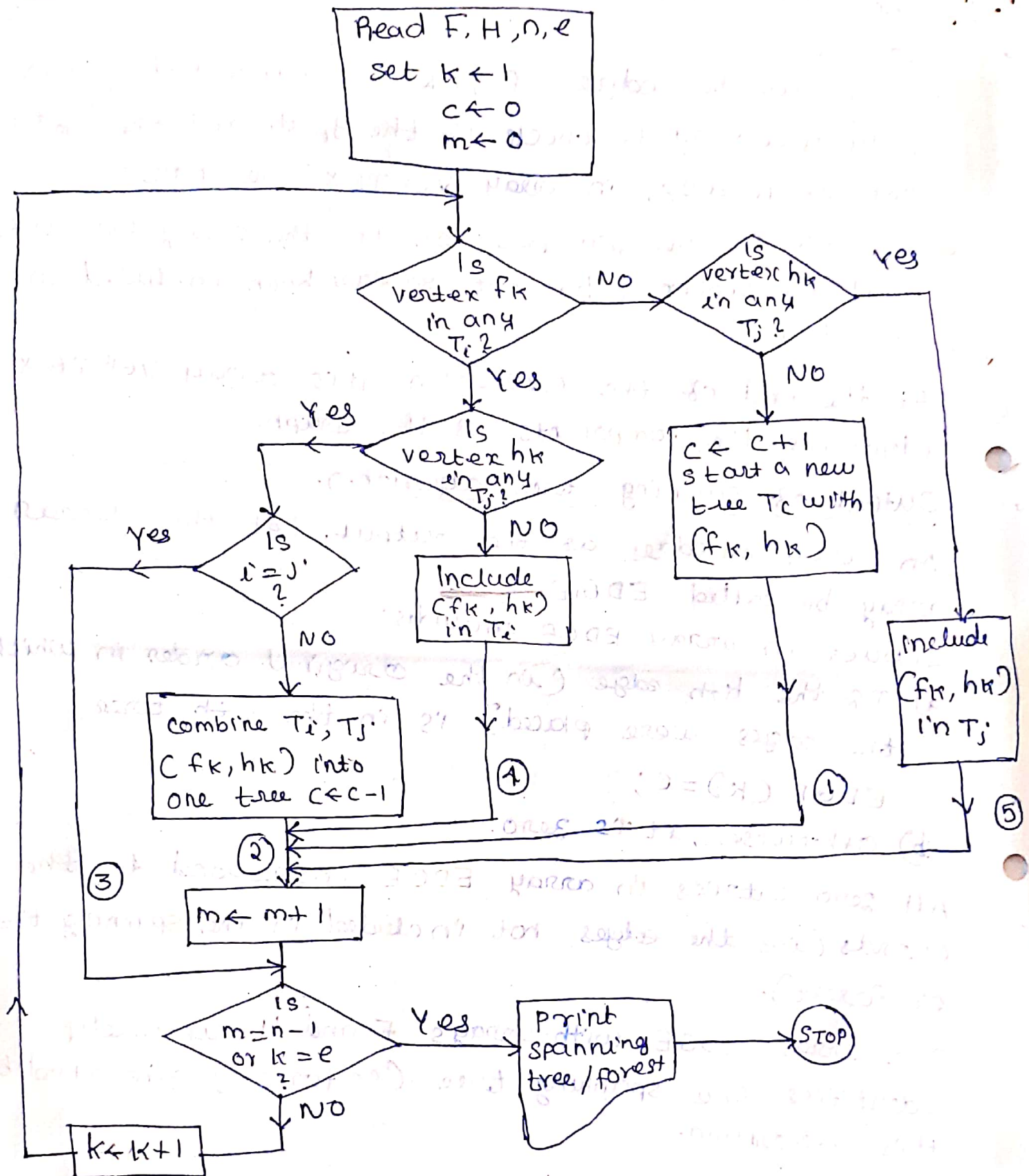


Fig. 6-3 Algorithm: Spanning tree / forest.

Execution of Algorithm:

- main loop is executed e times (e being the number of edges).
- The time required to test whether or not the end vertices have appeared in any tree is constant, independent of both e and n .
- The time bound for the execution of the algorithm is proportional to e .
- If e/n is high, execution time can be reduced by introducing a new variable to keep count of the edges included in the tree.
When this variable reaches the value of $n-1$, the program would terminate.

SHORTEST-PATH ALGORITHMS

Shortest-path problems include:

- Shortest path between two specified vertices.
 - Shortest paths between all pairs of vertices.
 - Shortest paths from a specified vertex to all others.
 - Shortest path between specified vertices that passes through specified vertices.
 - The second, third, and so on, shortest paths.
- Shortest path from a specified vertex to another specified vertex - Algorithm.

Problem statement:

A simple weighted digraph G of n vertices is described by an n by n matrix $D = [d_{ij}]$, where d_{ij} = length (or distance or weight) of the directed edge from vertex i to vertex j , $d_{ij} \geq 0$,

$$d_{ii} = 0,$$

$d_{ij} = \infty$, if there is no edge from i to j (in program ∞ is replaced by a large number).

$d_{ij} = \infty$, if there is no edge from i to j

In general, $d_{ij} \neq d_{ji}$, and the triangle inequality need not be satisfied. That is, $d_{ij} + d_{jk}$ may be less than d_{ik} .

The distance of a directed path P is defined to be the sum of the lengths of the edges in P . The problem is to find the shortest possible path and its length from a starting vertex s to a terminal vertex t .

Description of the Algorithm.

Dijkstra's algorithm: labels the vertices of the given digraph.

At each stage in the algorithm, some vertices have permanent labels and others temporary labels.



will not change
is the shortest distance of
the vertex from source s .

(1)



will change.

Step 1: Assign a permanent label 0 to the starting vertex s , and a temporary label ∞ to the remaining $n-1$ vertices.

Step 2: In each iteration another vertex gets a permanent label, according to the following rules:

- a) Every vertex j that is not yet permanently labeled gets a new temporary label whose value is given by

$$\min[\text{old label of } j, (\text{old label of } i + d_{ij})],$$

where i is the latest vertex permanently labeled, in the previous iteration, and d_{ij} is the direct distance between vertices i and j . If i and j are not joined by an edge, then $d_{ij} = \infty$.

- b) The smallest value among all the temporary labels is found, and this becomes the permanent label of the corresponding vertex. In case of a tie, select any one of the candidates for permanent labeling.

Step 3: Step 2 is repeated alternately until the destination vertex t gets a permanent label.

Execution

- 1) The first vertex to be permanently labeled is at a distance of zero from s .
- 2) The second vertex to get a permanent label (one of the remaining $n-1$ vertices) is the vertex closest to s .

3) From the remaining $n-2$ vertices, the next one to be permanently labeled is the second closest vertex to s . And so on.

The permanent label of each vertex is the shortest distance of that vertex from s .

Example: Find out the distance from vertex B to G in the digraph shown in Fig. 6-4.

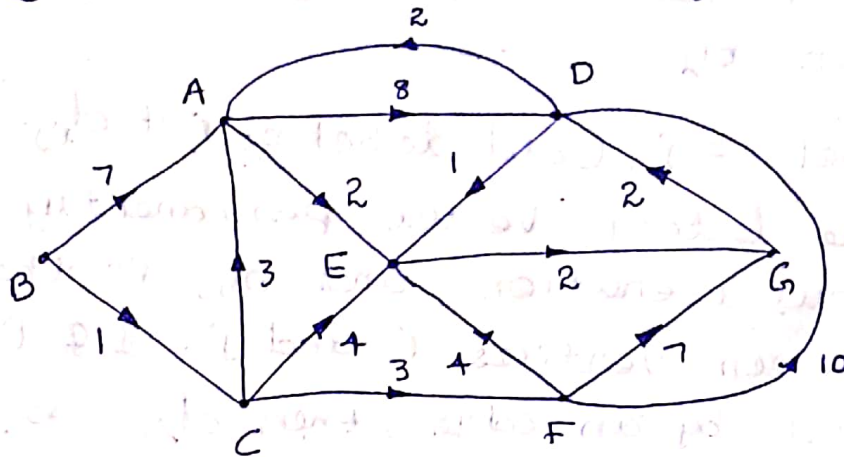


Fig. 6-4 Simple weighted digraph.

Answer: We shall use a vector of length seven to show the temporary and permanent labels of the vertices as we go through the solution.

The permanent labels will be shown enclosed in a square, and the most recently assigned permanent label in the vector is indicated by a tick \square^v .

The labeling proceeds as follows:

A	B	C	D	E	F	G	
∞	$\boxed{0}^{\vee}$	∞	∞	∞	∞	∞	: Starting Vertex B is labeled 0.
7	$\boxed{0}$	1	∞	∞	∞	∞	: All successors of B get labeled.
7	$\boxed{0}$	$\boxed{1}^{\vee}$	∞	∞	∞	∞	: Smallest label becomes permanent.
4	$\boxed{0}$	$\boxed{1}$	∞	5	4	∞	: Successors of C get labeled.
4	$\boxed{0}$	$\boxed{1}$	∞	5	$\boxed{4}^{\vee}$	∞	
4	$\boxed{0}$	$\boxed{1}$	14	5	$\boxed{4}$	11	
$\boxed{4}^{\vee}$	$\boxed{0}$	$\boxed{1}$	14	5	$\boxed{4}$	11	
$\boxed{4}$	$\boxed{0}$	$\boxed{1}$	12	5	$\boxed{4}$	11	
$\boxed{4}$	$\boxed{0}$	$\boxed{1}$	12	$\boxed{5}^{\vee}$	$\boxed{4}$	11	
$\boxed{4}$	$\boxed{0}$	$\boxed{1}$	12	$\boxed{5}$	$\boxed{4}$	7	
$\boxed{4}$	$\boxed{0}$	$\boxed{1}$	12	$\boxed{5}$	4	$\boxed{7}^{\vee}$: Destination vertex gets permanently labeled.

- Shortest Path from B to G -

The permanent label of the destination vertex is the shortest distance from the source vertex. Therefore the shortest distance from vertex B to the destination (terminal) vertex G is 7.

- 1) The shortest path can be easily constructed by working backward from the terminal vertex such that we go to that predecessor whose label differs exactly by the length of the connection edge.
- 2) Alternatively, the shortest path can be determined by keeping a record of the vertices from which each vertex was labeled permanently. This

record can be maintained by another linear array of length n , such that whenever a new permanent label is assigned to vertex j , the vertex from which j is directly reached is recorded in the j th position of this array.

Therefore the shortest path is BCEG

Programming

An efficient method of distinguishing the permanently labeled vertices from the temporarily labeled ones is to associate indices $1, 2, \dots, n$ with the vertices, and keep a binary vector $VECT$ of order n . When the i th vertex becomes permanently labeled, the i th element in this binary vector changes from 0 to 1.

Remarks

1. Algorithm for the shortest paths from starting vertex s to all other vertices.

In the single pair shortest path algorithm had we continued the labeling until every vertex got a permanent label, we would have gotten an algorithm for the shortest paths from starting vertex s to all other vertices.

2. Shortest-distance ~~abscissa~~ arborescence (shortest-distance tree)

If we take a shortest path from the starting

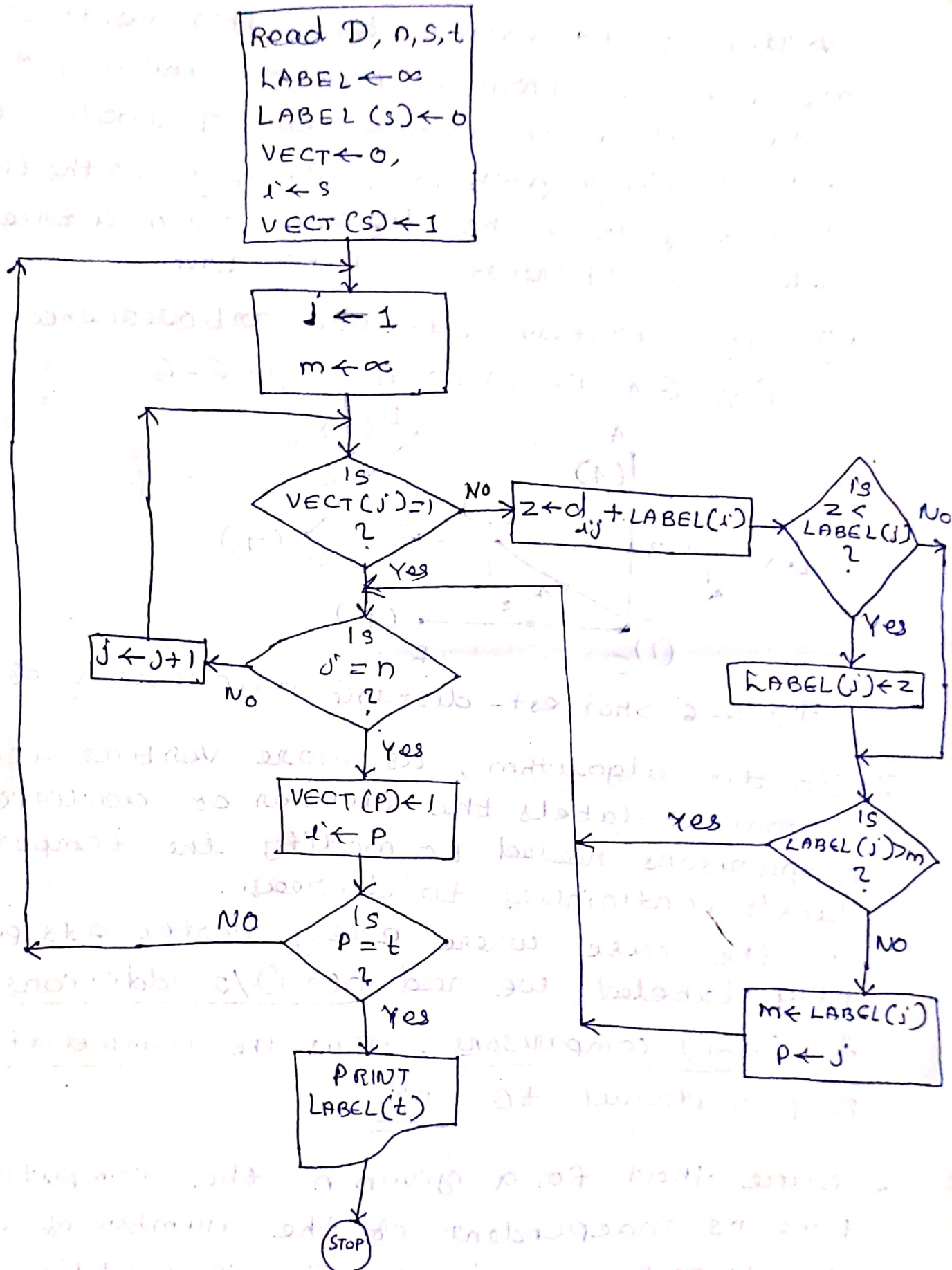


Fig. 6-5 Algorithm: Shortest distance from s to t .

vertex s to each of the other vertices (which are accessible from s), then the union of these paths will be an arborescence T rooted at vertex s . Every path in T from s is the (unique) shortest path in the digraph. Such a tree is called the shortest-distance tree.

Eg: the shortest-distance arborescence of Fig. 6-4 is given in Fig. 6-6.

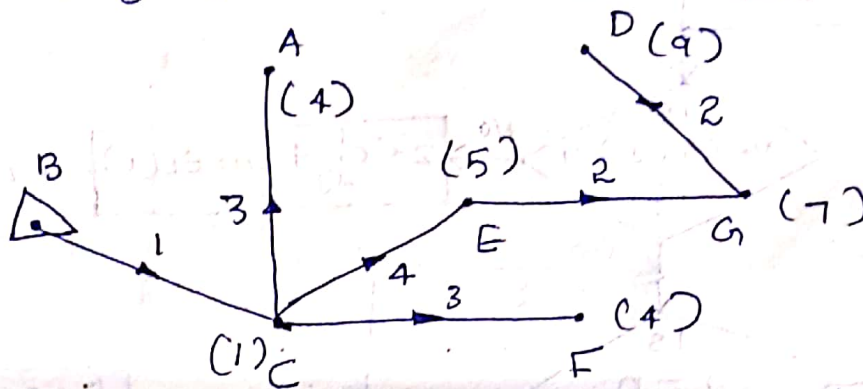


Fig. 6-6 Shortest-distance arborescence of Fig. 6-4.

3. In this algorithm, as more vertices acquire permanent labels the number of additions and comparisons needed to modify the temporary labels continues to decrease.

In the case where every vertex gets permanently labeled, we need $\frac{n(n-1)}{2}$ additions and $2n(n-1)$ comparisons. Thus the computation time is proportional to n^2 .

4. Notice that for a given n the computation time is independent of the number of edges the digraph may have. This is because it is

tacitly assumed that the digraph is complete. Each missing edge is simply given a very large weight.

5. If the digraph is sparse [i.e., the number of edges e is much smaller than $n(n-1)$], it is possible to reduce the time of computation. This can be achieved by incorporating another test which alters the temporary labels of only those vertices that are successors of the most recent permanently labeled vertex.
6. If the given digraph G is not weighted, every edge in G has a weight of one, and matrix D is the same as the adjacency matrix. Then the problem is simpler. We perform logical operations rather than real arithmetic.
7. We have assumed the distances d_{ij} are all nonnegative numbers. If some of the distances are negative, Algorithm will not work. The reason for the failure of Algorithm is that once a vertex is permanently labeled its label cannot be altered.
8. Carrying the shortest-path algorithm simultaneously from both end s and t would improve the speed. The double-ended procedure is actually less efficient than Dijkstra's one ended procedure.

Algorithm: Shortest Path Between All pairs of vertices.

Warshall-Floyd algorithm.

Description of the Algorithm:

- 1) The algorithm works by inserting one or more vertices into paths, whenever it is advantageous to do so.
- 2) Starting with the n by n matrix $D = [d_{ij}]$ of direct distances, n different matrices D_1, D_2, \dots, D_n are constructed sequentially.
- 3) Matrix D_k , $1 \leq k \leq n$, may be thought of as the matrix whose (i, j) th entry gives the length of the shortest directed path among all directed paths from i to j , with vertices $1, 2, \dots, k$ allowed as the intermediate vertices.
- 4) Matrix $D_k = [d_{ij}^{(k)}]$ is constructed from D_{k-1} according to the following rule:

$$d_{ij}^{(k)} = \min [d_{ij}^{(k-1)}, (d_{ik}^{(k-1)} + d_{kj}^{(k-1)})]$$
 for $k = 1, 2, \dots, n$,
 and $d_{ij}^{(0)} = d_{ij}$.

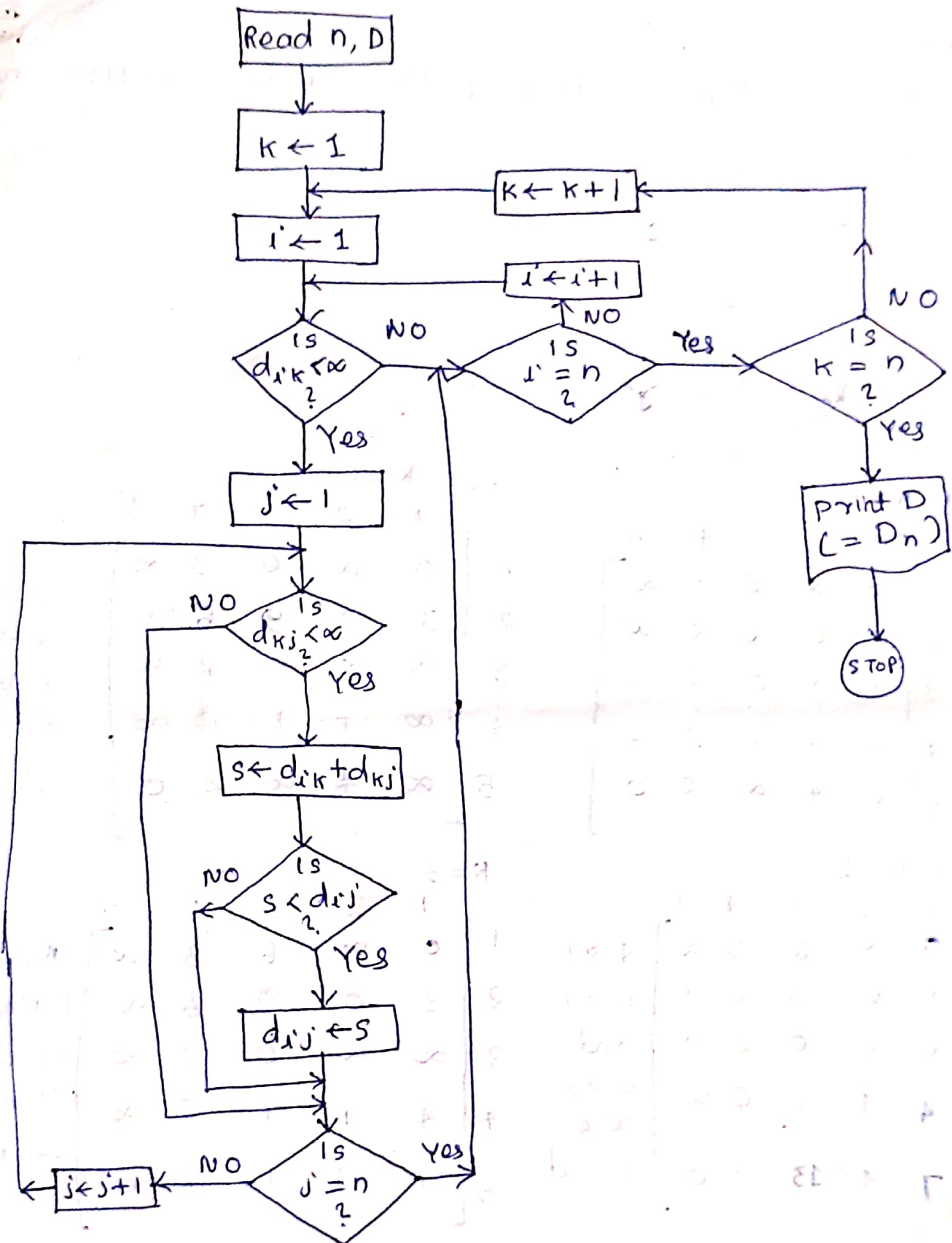
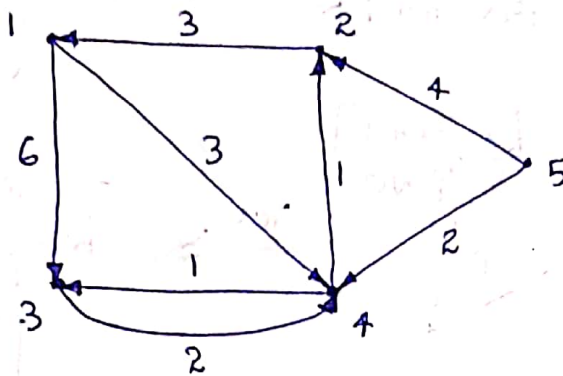


Fig. 6-7 Algorithm : Shortest path between every vertex-pair.

Example

Find All pair shortest path for the following graph.



for $k=0$

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	∞	∞	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

for $k=1$

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	9	6	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

2 \rightarrow 3
2 \rightarrow 4
are
found

for $k=2$

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	9	6	∞
3	∞	∞	0	2	∞
4	4	1	1	0	∞
5	7	4	13	2	0

4 \rightarrow 1
5 \rightarrow 1
and
5 \rightarrow 3
are
found

$k=3$

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	9	6	∞
3	∞	∞	0	2	∞
4	4	1	1	0	∞
5	7	4	13	2	0

No shorter
paths are
found
through
vertex 3.

for $k=4$

	1	2	3	4	5	
1	0	4	4	3	∞	$1 \rightarrow 2$
2	3	0	7	6	∞	$1 \rightarrow 3$
3	6	3	0	2	∞	$2 \rightarrow 3$
4	4	1	1	0	∞	$3 \rightarrow 1$
5	6	3	3	2	0	$3 \rightarrow 2$

are found through vertex 4.

$5 \rightarrow 1$
 $5 \rightarrow 2$
 $5 \rightarrow 3$
 $5 \rightarrow$

for $k=5$

	1	2	3	4	5	
1	0	4	4	3	∞	All-pairs shortest distance.
2	3	0	7	6	∞	
3	6	3	0	2	∞	
4	4	1	1	0	∞	
5	6	3	3	2	0	

APPENDIX 1

CONTENT BEYOND THE SYLLABUS

1. THE GIRTH OF A GRAPH

Definition A graph containing no cycles is called a forest. In a forest, every component is a tree. So a tree is a forest. We say that the girth of a forest is infinite.

Definition When G is not a forest, we define the girth of G as the size of the smallest cycle in G . The graph shown below has girth 8.

CHROMATIC NUMBER AND GIRTH

Theorem(Erdős, '59) For every pair (g, t) of positive integers with $g, t \geq 3$, there is a graph G with girth g and chromatic number t .

2. ON-LINE COLORING LINE COLORING – A TWO PERSON GAME

Builder constructs a graph one vertex at a time.

Assigner colors the graph in an on-line manner.

Fact: Even in the class of forests, Builder can force n colors on a graph with $2n-1$ vertices.

Explanation: Let S_n be the Builder's strategy for forcing n colors. Then S_{n+1} can be viewed as adding one new vertex to the disjoint application of $S_1, S_2, S_3, \dots, S_n$ and then adding one new vertex.

Theorem (Kierstead and Trotter, '82) In the class of interval graphs, there is a strategy for Assigner that will enable her to color an interval graph with $3k-2$ colors provided Builder keeps the maximum clique size at most k . Builder does not need to know the value of k in advance. Furthermore, this bound is best possible, since there is a strategy for Builder that will force assigner to use at least $3k-2$ colors, regardless of the strategy used in assigning colors.

GAME COLORING FOR GRAPHS

Definition: The game chromatic number of a graph is the least positive integer t for which there is a strategy for Alice that will enable her, working in “cooperation” with Bob, to color the graph using t colors and alternating turns.

Note: The issue as to who goes first can be important.

Theorem(Kierstead and Trotter, '94) The game chromatic number of a planar graph is at most 33.

TWO CHALLENGING EXERCISES

Observation: The chromatic number of a tree is two if it has an edge. However, the game chromatic number of a tree is at most 4 and this result is best possible. This is a good exercise for a senior level undergraduate course in graph theory.

Follow-Up: Note Kierstead and Zhu have been carrying on a running competition for 20 years, and it is now known that the game chromatic number of a planar graph is at most 17 with Zhu in the winning position for now. From below, a lower bound of 7 is known. If you really want to get an A+++, move either bound.

LIST COLORINGS OF GRAPHS

Definition The list chromatic number of a graph is the smallest integer t so that a proper coloring of the graph can always be found using colors from prescribed lists of size t , one list for each vertex. Note that different vertices can have different lists.

Example When $n = C(2t-1, t)$, the complete bipartite graph $K_{n,n}$ has list chromatic number $t + 1$.

Theorem(Thomassen, 1994) The list chromatic number of a planar graph is at most 5.

COCOLORING

In graph theory, a cocoloring of a graph G is an assignment of colors to the vertices such that each color class forms an independent set in G or in the complement of G . The cochromatic number $z(G)$ of G is the fewest colors needed in any cocolorings of G . The graphs with cochromatic number 2 are exactly the bipartite graphs, complements of bipartite graphs, and split graphs.

As the requirement that each color class be a clique or independent is weaker than the requirement for coloring (in which each color class must be an independent set) and stronger than for subcoloring (in which each color class must be a disjoint union of cliques), it follows that the cochromatic number of G is less than or equal to the chromatic number of G , and that it is greater than or equal to the subchromatic number of G .

Cocoloring was named and first studied by Lesniak & Straight (1977). Jørgensen (1995) characterizes critical 3-cochromatic graphs, while Fomin, Kratsch & Novelli (2002) describe algorithms for approximating the cochromatic number of a graph. Zverovich (2000) defines a class of perfect cochromatic graphs, analogous to the definition of perfect graphs via graph coloring, and provides a forbidden subgraph characterization of these graphs.

TOTAL COLORING

In graph theory, total coloring is a type of graph coloring on the vertices and edges of a graph. When used without any qualification, a total coloring is always assumed to be proper in the sense that no adjacent edges and no edge and its endvertices are assigned the same color. The total chromatic number $\chi''(G)$ of a graph G is the least number of colors needed in any total coloring of G .

The total graph $T = T(G)$ of a graph G is a graph such that (i) the vertex set of T corresponds to the vertices and edges of G and (ii) two vertices are adjacent in T if and only if their corresponding elements are either adjacent or incident in G . Then total coloring becomes a (proper) vertex coloring of the total graph. A total coloring is a partitioning of the vertices and edges of the graph into total independent sets.

Some properties of $\chi''(G)$:

1. $\chi''(G) \geq \Delta(G) + 1$.
2. $\chi''(G) \leq \Delta(G) + 1026$. (Molloy, Reed 1998)
3. $\chi''(G) \leq \Delta(G) + 8 \ln 8(\Delta(G))$ for sufficiently large $\Delta(G)$. (Hind, Molloy, Reed 1998)
4. $\chi''(G) \leq \text{ch}'(G) + 2$.

Here $\Delta(G)$ is the maximum degree; and $\text{ch}'(G)$, the edge choosability.

3. INTERVAL GRAPH

Formally, an interval graph is an undirected graph formed from a family of intervals

$S_i, i = 0, 1, 2, \dots$

by creating one vertex v_i for each interval S_i , and connecting two vertices v_i and v_j by an edge whenever the corresponding two sets have a nonempty intersection, that is,

$$E(G) = \{ \{v_i, v_j\} \mid S_i \cap S_j \neq \emptyset \}.$$

From this construction one can verify a common property held by all interval graphs. That is, graph G is an interval graph if and only if the maximal cliques of G can be ordered M_1, M_2, \dots, M_k such that for any $v \in M_i \cap M_k$, where $i < k$, it is also the case that $v \in M_j$ for any $M_j, i \leq j \leq k$.

4. VISIBILITY GRAPH

In computational geometry and robot motion planning, a visibility graph is a graph of intervisible locations, typically for a set of points and obstacles in the Euclidean plane. Each node in the graph represents a point location, and each edge represents a visible connection between them. That is, if the line segment connecting two locations does not pass through any obstacle, an edge is drawn between them in the graph. When the set of locations lies in a line, this can be understood as an ordered series. Visibility graphs have therefore been extended to the realm of time series analysis.

Visibility graphs may be used to find Euclidean shortest paths among a set of polygonal obstacles in the plane: the shortest path between two obstacles follows straight line segments except at the vertices of the obstacles, where it may turn, so the Euclidean shortest path is the shortest path in a visibility graph that has as its nodes the start and destination points and the vertices of the obstacles.[1] Therefore, the Euclidean shortest path problem may be decomposed into two simpler sub problems: constructing the visibility graph, and applying a shortest path algorithm such as Dijkstra's algorithm to the graph. For planning the motion of a robot that has non-negligible size compared to the obstacles, a similar approach may be used after expanding the obstacles to compensate for the size of the robot.[1] Lozano-Pérez & Wesley (1979) attribute the visibility graph method for Euclidean shortest paths to research in 1969 by Nils Nilsson on motion planning for Shakey the robot, and also cite a 1973 description of this method by Russian mathematicians M. B. Ignat'yev, F. M. Kulakov, and A. M. Pokrovskiy.

Visibility graphs may also be used to calculate the placement of radio antennas, or as a tool used within architecture and urban planning through visibility graph analysis.

The visibility graph of a set of locations that lie in a line can be interpreted as a graph-theoretical representation of a time series.[2] This particular case builds a bridge between time series, dynamical systems and graph theory.

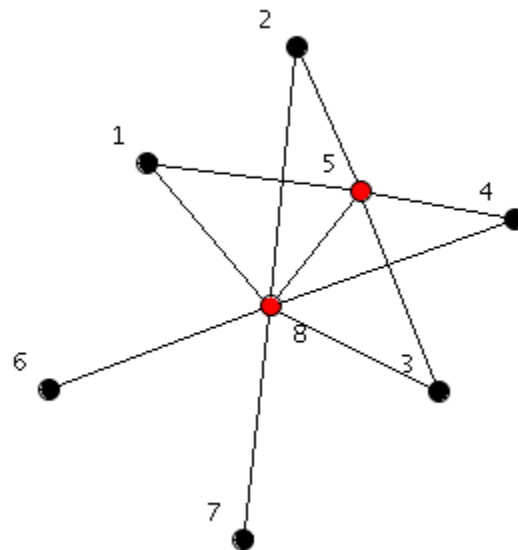
5. THRESHOLD GRAPH

In graph theory, a threshold graph is a graph that can be constructed from a one-vertex graph by repeated applications of the following two operations:

1. Addition of a single isolated vertex to the graph.
2. Addition of a single dominating vertex to the graph, i.e. a single vertex that is connected to all other vertices.

For example, the graph of the figure is a threshold graph. It can be constructed by beginning with a single-vertex graph (vertex 1), and then adding black vertices as isolated vertices and red vertices as dominating vertices, in the order in which they are numbered.

Threshold graphs were first introduced by Chvátal & Hammer (1977). A chapter on threshold graphs appears in Golumbic (1980), and the book Mahadev & Peled (1995) is devoted to them.



An example of a threshold graph