PREFACE

In a bid to standardise higher education in the country, the University Grants Commission (UGC) has introduced Choice Based Credit System (CBCS) based on five types of courses: core, generic discipline specific elective, and ability/ skill enhancement for graduate students of all programmes at Elective/ Honours level. This brings in the semester pattern, which finds efficacy in tandem with credit system, credit transfer, comprehensive and continuous assessments and a graded pattern of evaluation. The objective is to offer learners ample flexibility to choose from a wide gamut of courses, as also to provide them lateral mobility between various educational institutions in the country where they can carry acquired credits. I am happy to note that the University has been recently accredited by National Assessment and Accreditation Council of India (NAAC) with grade “A”.

UGC (Open and Distance Learning programmes and Online Programmes) Regulations, 2020 have mandated compliance with CBCS for all the HEIs in this mode. Welcoming this paradigm shift in higher education, Netaji Subhas Open University (NSOU) has resolved to adopt CBCS from the academic session 2021-22 at the Under Graduate Degree Programme level. The present syllabus, framed in the spirit of syllabi recommended by UGC, lays due stress on all aspects envisaged in the curricular framework of the apex body on higher education. It will be imparted to learners over the six semesters of the Programme.

Self Learning Materials (SLMs) are the mainstay of Student Support Services (SSS) of an Open University. From a logistic point of view, NSOU has embarked upon CBCS presently with SLMs in English. Eventually, these will be translated into Bengali too, for the benefit of learners. As always, we have requisitioned the services of the best academics in each domain for the preparation of new SLMs, and I am sure they will be of commendable academic support. We look forward to proactive feedback from all stake-holders who will participate in the teaching-learning of these study materials. It has been a very challenging task well executed, and 1 congratulate all concerned in the preparation of these SLMs.

I wish the venture a grand success.

Professor (Dr.) Subha Sankar Sarkar

Vice-Chancellor

Netaji Subhas Open University

Under Graduate Degree Programme

Choice Based Credit System (CBCS)

Subject: Honours in Mathematics (HMT)

Course Code : SE-MT-21

Course: Graph Theory

First Print : May, 2022

Printed in accordance with the regulations of the Distance Education Bureau
of the University Grants Commission.

Netaji Subhas Open University

Under Graduate Degree Programme

Choice Based Credit System (CBCS)

Subject: Honours in Mathematics (HMT)

Course Code : SE-MT-21

Course: Graph Theory

: Course Writer :

Dr. Sovan Samanta
Assistant Professor of Mathematics
Tamralipta Mahavidyalaya

: Course Editor :

Prof. Kajal De
Professor of Mathematics
Netaji Subhas Open University

: Format Editor :

Prof. Kajal De
Professor of Mathematics, NSOU

Netaji Subhas Open University

Under Graduate: HMT
(SEC-2)

Course: Graph Theory

Course Code: SE-MT-21

Unit 1 ❑ Basics of graph theory

Structure

1.0 Objective

This unit will help the students to understand the basic concept of graph theory and how the subject evolved to its present form.

1.1 Introduction

The subject of mathematics is broad. It is a tree of information branches into an ever-growing number of sub-topics. Numbers in problems can either be discrete, like natural numbers 1,2,3,4... Alternatively, they can be continuous, which precisely map our reality as dynamic, changing values.

What is the fastest route from the national capital to each state headquarters? What is the maximum flow per unit time from source to sink in a network? Can we colour the regions of every map using four colours so that neighbouring regions have a different colour? These and many other problems are solved in graph theory. Graph theory, a discrete mathematics topic, is at the highest level the study of the link between things. These things are more formally involved to as vertices, vertexes or nodes, with the links themselves pointed to as edges.

1.2 History of Graph Theory

Swiss mathematician Leonhard Euler first originated the fundamental idea of graphs in the 18th century. His efforts and ultimate solution to the famous Königsberg bridge puzzle pictured below are generally quoted as the origin of graph theory.

The German city of Königsberg (Kaliningrad, Russia) is located on the Pregolya River. The geographic layout is founded of four main bodies of land attached by a total of seven bridges (see Figure 1.1). The question pretended to Euler was straightforward: was it possible to catch a walk by the town in such a way as to traverse over every bridge once, and only once?

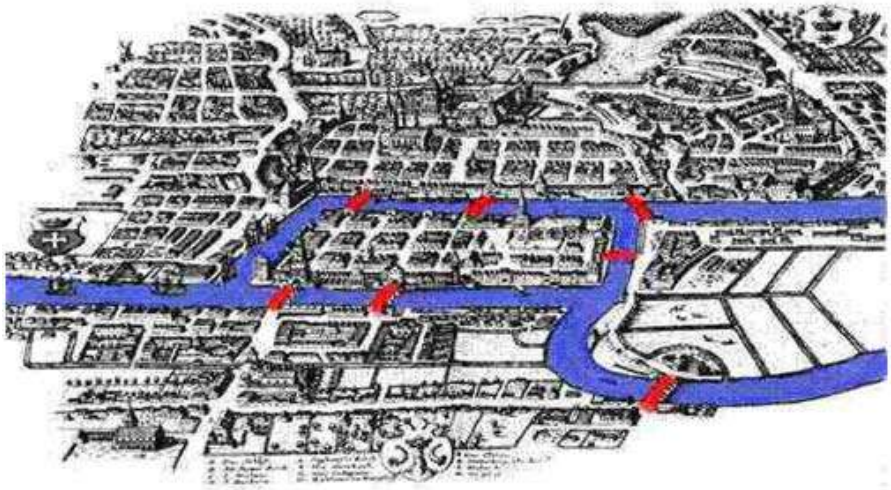


Figure 1.1: City of Königsberg (Kaliningrad, Russia)

Euler, realising that the proper constraints were the four bodies of land & the seven bridges, carried out the first recognized visual representation of a current graph. A graph (see Figure 1.2(c)) is represented by a set of points, named as vertices or nodes, that linked by a set of joining lines known as edges.

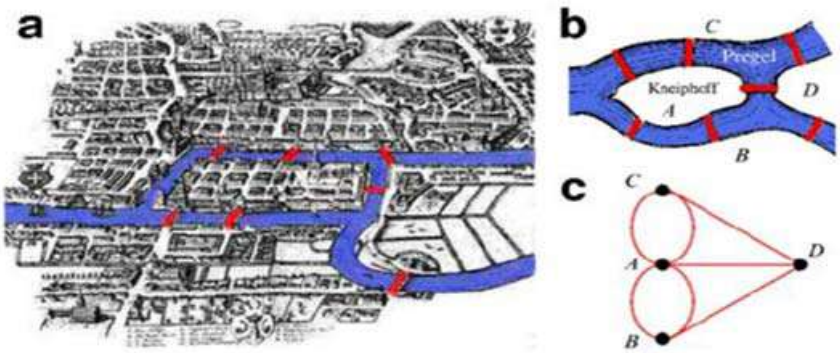


Figure 1.2: Graphical representation of the Konigsberg Bridgeproblem

By first endeavouring to draw paths in the graph above, then later testing with multiple theoretical graphs with an alternating number of vertices and edges, he

ultimately extrapolated a universal rule:

In order to be worthy to walk in a Euler path (without repeating an edge), a graph must have none or twice an odd number of nodes? From there, the chapter of mathematics known as graph theory extended asleep for decades. In recent times, however, it is utilised in many fields.

The paper of Leonhard Euler on the Seven Bridges problem of Königsberg in 1736 is considered as the first paper in the history of graph theory.

1.3 Applications of Graph Theory

Graph Theory is eventually the study of relationships. For a set of nodes and connections, graph theory contributes a helpful tool to quantify and simplify the many evolving parts of dynamic methods. Studying graphs presents answers to many arrangements, optimisation, networking, matching and operational problems.Graph theory has its applicability in various fields of engineering described as follows.

Electrical Engineering : The ideas of graph theory, is applied extensively in designing circuit connections. The classes or organization of connections are defined as topologies. Some examples of topologies arestar, bridge, series, and parallel topologies.

Computer Science : Graph theory is applied to the learning of algorithms. Kruskal's Algorithm, Prim's Algorithm, Dijkstra's Algorithm,Warshall Algorithms, Computer Network, are few to mention. The relationships among interrelated computers in the network follow the postulates of graph theory.

Science : The molecular structure, the DNA structure of an organism and chemical structure of a substance, etc., are depicted by graph theory.

Linguistics : The analyzing tree of a language and grammar of a language use graphs.

General : Routes among the cities can be designed using graph theory. Depicting ordered information (hierarchical) such as family tree can be used as a particular type of graph called a tree.

1.4 Summary

After studying this units, the learners will become interested in the topic of graph theory and wish to understand the applications of it deeply.

Unit 2 ❑ Basic properties of graphs

Structure

2.0 Objective

The objective of this unit is to give some concept of the terminology used in graph theory with a few basic properties.

2.1 Introduction

This section has been planned for students who desire to learn the primary of Graph Theory. Before starting with this chapter, readers need to know the basics of number theory and set-theoretic operations in Mathematics. It is compulsory to have a basic knowledge of Computer Science as well.

2.2 Pseudo-graphs

A graph is a pictorial illustration of a set of objects where links connect some pairs of objects. Points termed as vertices express the objects, and the links that connect the vertices are named edges.

Definition 2.1: A graph is a pair of sets (V,E), where V is the set of vertices and E is the set of edges, connecting the pairs of vertices.

Example 2.1

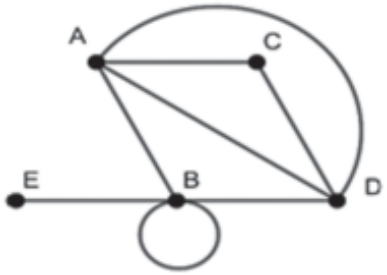


Figure 2.1: Pictorial representation of a graph

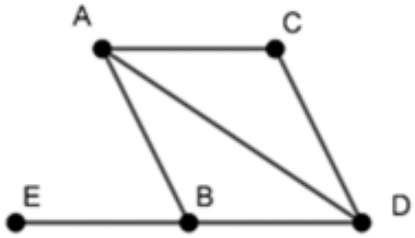


Figure 2.2: Pictorial representation of a simple graph

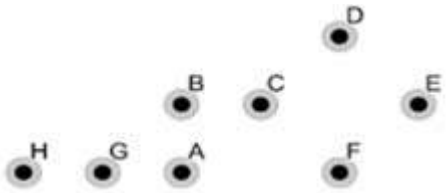


Figure 2.3: Pictorial representation of a null graph

In the given Figure 2.1, {A,B,C,D,E} is the set of vertices. The edges have no direction and hence, (A,B) and (B,A) indicate the same edge. However, it is seen that two edges connect A and D. These edges are called multi-edge or parallel edge (formal definition is given in the following Table 1.1). Thus the edge set will contain the edge (A,D) twice. Also, it is seen that one edge started from B and ended to B. This edge is called a loop. The complete edge set is {(A,B), (A,C), (A,D), (A,D), (B,B), (B,D), (C,D), (B,E)}. Naturally, this edge set is multi-set.

2.2 Pseudo-graphs

A graph which has no finite vertex and /or edge set is called an infinite graph. A pseudograph is a non-simple graph in which loops and (or) multiple edges (parallel edges) are permitted.

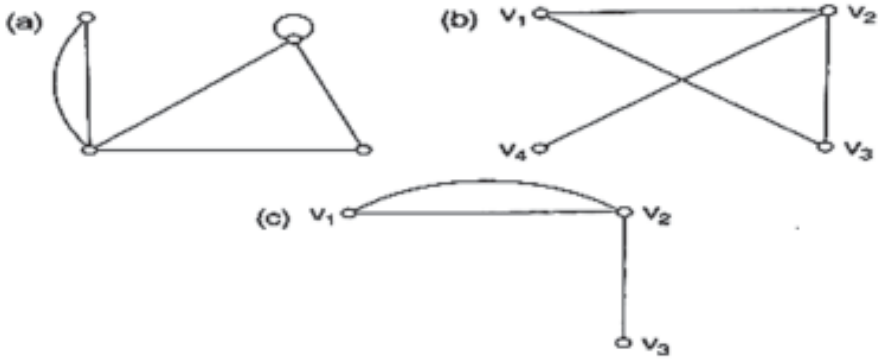


Figure 2.4: Pseudo graphs (a and c), simple graphs(b).

Example 2.2: In Figure 2.4 (a) both parallel edges and self-loop exist. In Figure 2.4 (c) parallel edges exist. So these graphs are pseudo graphs (or multi-graphs). In Figure 2.4 (b), no parallel edges or self-loop exist, so the graph is a simple graph.

Example 2.3: Let $V = \{ v_1,v_2,v_3,v_4\}$, $E = \{e_1,e_2,e_3,e_4,e_5,e_6\}$ such that $e_1 = e_5 = (v_1,v_2)$, $e_2 = (v_3,v_4)$, $e_3 = (v_1,v_3)$, $e_4 = e_6 = (v_2,v_4)$.

The graph is drawn as follows:

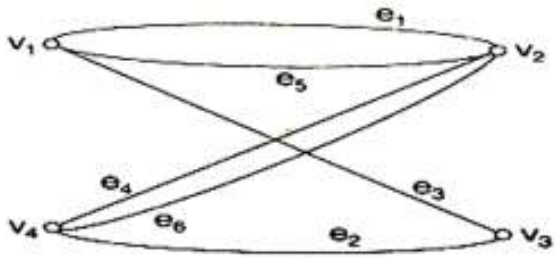


Figure 2.5: Pictorial representation of a multi-graph.

Terminology	Explanation
End vertices	The two vertices u and v are end vertices of an edge (u,v).
Parallel edges	Edges that have the same end vertices are parallel edges
Loop	An edge of the form (v,v) is a loop.
Simple graphs	A graph is simple if it has no parallel edges or loops.
Null graph	A graph (V,E) with no edges (i.e. E is empty) is a null graph
Adjacent vertices	Two vertices uand vare adjacent if an edge connects u and v.
Adjacent edge	Edges are adjacent if they share a common end vertex.
Degree of a vertex	The degree of the vertex v, written as d(v), is the number of edges with vas an end vertex. By the conventional way, we count a loop twice, and parallel edges contribute separately.
Pendant vertex	A pendant vertex is a vertex whose degree is 1.
Pendant edge	An edge that has a pendant vertex as an end vertex is a pendant edge.
Isolated vertex	An isolated vertex is a vertex whose degree is 0.
$\delta(G)$	The minimum degree of the vertices in a graph G is denoted $\delta(G)$
$\Delta(G)$	The maximum degree of vertices in G is denoted $\Delta(G)$.

Table 2.1: Few standard terminologies

Terminology	Examples
End vertices	In Figure 2.1, A and Bare end-vertices of the edge (A,B)
Parallel edges	In Figure 2.1, the edges between A and D are parallel edges
Loop	In Figure 2.1, (B,B) is a loop.
Simple graphs	In Figure 2.2, one simple graph is drawn as it has no parallel edges or loops.

Null graph	One null graph is drawn in Figure 2.3.
Adjacent vertices	In Figure 2.1, A and B are adjacent vertices.
Adjacent edge	In Figure 2.1, the edges (A,B) and (B,D) are adjacent.
Degree of a vertex	In Figure 2.1, the degree of B,d(B) is 5, and in Figure 1.5, d(A)=0.
Pendant vertex	In Figure 2.2, E is a pendant vertex.
Pendant edge	In Figure 2.2, (B,E) is an pendant edge.
Isolated vertex	All vertices in Figure 2.3, all vertices are isolated vertices
$\delta(G)$	In Figure 2.3, δ (G)=0, In Figure 2.2., δ (G)=1, In Figure 2.1,δ (G)=1,
$\Delta(G)$	In Figure 2.3,$\Delta(G)=0$, In Figure 2.2, $\Delta(G)=3$, In Figure 2.1, $\Delta(G)=5$,

Table 2.2: Terminologies with examples of Table 2.1

2.3 Few more terminology

2.3.1 Walks

A finite walk is a finite alternating sequence of vertices and edges, starting and ending with vertices, such that each edge is incident with the vertices preceding and following it. An infinite walk is an alternating sequence of vertices and edges such that each edge is incident with the vertices preceding and following it, but with no first or last vertex.

In graph theory, a finite walk is of the form? $v_0e_0v_1e_1v_2\ldots e_{(n-1)}v_n$. In a walk, both vertices and edges may appear repeatedly. The starting and ending vertices of a walk are called the terminal vertices of walk and rest of vertices are called intermediate vertices. A walk between two vertices u and v is generally denoted by u-v walk or uv walk. A walk may be closed or open. If the beginning and ending vertices of a walk are same, then that walk is called a closed walk or tour, otherwise, it is called an open walk. Note that the length of a walk is simply the number of edges traversed in that walk. A walk may intersect itself.

Example 2.4: For the graph shown in Figure 2.6,

i) 1a3c4e5d3b2 is a finite open walk-in which vertex 3 is repeated.

ii) 1a3c4g6f5e4c3b2 is a finite walk in which vertices 3, 4 and edge c are repeated.

iii) 3c4e5d3 is a closed walk in which starting and ending vertices are the same.

iv) 3c4g6j7k9m8i6f5d3 is a closed walk where terminal vertices are the same, and an intermediate vertex 6 is repeated.

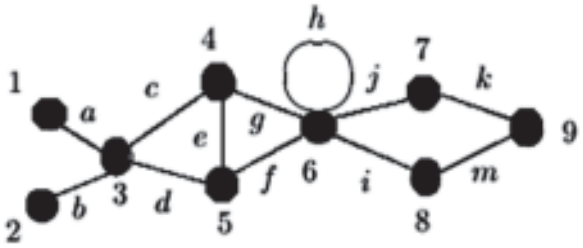


Figure 2.6: A pseudograph.

2.3.2 Trails

A trail is a walk in which all edges are distinct. Note that vertices of a trail may appear repeatedly. A trail may be open or close. If the beginning and ending vertices of a trail are same, then that trail is called closed trail; otherwise, it is called an open trail. Note that a trail may intersect itself. In the graph shown in Figure 2.6, 3c4g6j7k9m8i6f5d3 is a closed trail, and 4g6h6f5 is an open trail.

2.3.3 Paths

A path is an open trail with no repeated vertices. Degree of starting and ending vertices of a path are one and degree of each other vertices of the path are two. Note that all paths are walks but all walks may or may not be paths. Also, a path does not intersect itself. For example, 1a3c4g6 is a path in the graph of Figure 2.7.

2.3.4 Cycles

A closed trail with no other vertices are repeated apart from the start or end vertex is called a cycle. In other words, a cycle is nothing but a closed path. Note that not an edge is repeated in a cycle. A n-cycle is a cycle with n vertices. In Figure 2.6, 3c4e5d3 is a cycle of length three and 6j7k9m8i6 is a cycle of length four.

2.3.5 Circuits

If we traverse a graph in such a way that not an edge is repeated, but vertex can repeat and starting and ending vertices are same, then the traversing is called a circuit.

In other words, a circuit is a closed trail. Note that vertices (excluding start/end vertex) can repeat in a circuit but not in cycle. For examples, 3c4g6j7k9m8i6f5d3 and 6j7k9m8i6 are two circuits in the graph of Figure 2.6. Note that all cycles are circuits, but circuits are not always cycles.

Example 2.5: In Figure 2.7,$v_1,e_1,v_2,e_2,v_3,e_3,v_4$ is a path of length 3, $v_2,e_2,v_3,e_3,v_4,e_5,v_2$ is a cycle of length 3 and $v_2,e_6,v_5,e_7,v_6,e_9,v_7,e_8,v_5,e_4,v_4,e_5,v_2$ is a circuit of length 6.

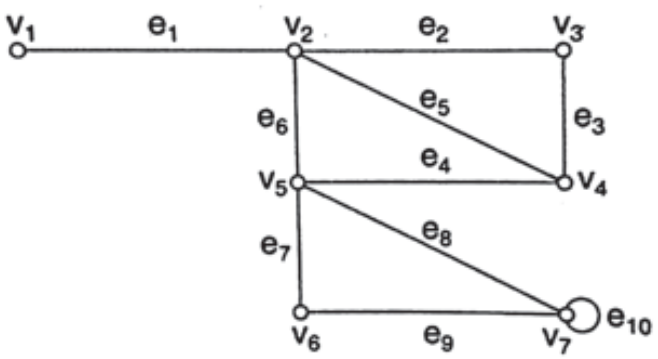


Figure 2.7: A pseudo graph

$v_1e_1v_2e_2v_3e_3v_4e_5v_2e_6v_5$ is a walk of length 5 and it is an open walk. This walk has no repeated edges, and hence it is a trail. Since the walk contains v_2 twice, i.e. repeated vertices, the walk is not a path.

Theorem 2.1: In a graph, any open trail contains a path.

Proof: To prove this theorem by induction on the length of the open trail, we consider an open u-v trail of length n. We know any open trail of length 1 is a path. So, the result is true for n = 1. Now, assume that the result is true for all open trails of length less than n. Let $u = v_0e_0v_1e_1v_2e_2\ldots e_{(n-1)}v_n = v$ be any open trail of length n. If all the vertices of the walk are distinct, then it is already a path. If not, there exist at least two vertices i,j such that $0 \le i < j \le n$ and $v_i = v_j$. Now, if we remove all the vertices (excluding v_i) and edges which belong to the walk between v_i and v_j, then $u = v_0e_0v_1e_1\ldots e_{(i-1)}v_ie_jv_{(j+1)}\ e_{(j+1)}\ v_{(j+2)}\ e_{(j+2)}\ldots e_{(n-1)}\ v_n = v$ is an open trail of length less than n which, by the induction hypothesis, contains a u–v path. Hence the theorem is proved.

Theorem 2.2: Every closed trail contains a cycle.

Proof : We prove this result by induction. For this purpose, let $v = v_0e_0v_1e_1v_2e_2\ldots e_{(n-1)}v_n = v$ be a closed trail of length n. Now, if n = 1, then it is a self-loop, so, the trail is a cycle of length one. So, the result is true for n = 1. Let us assume that the result is true for all trails of length less than n. Now we have to prove that the result is true for n, i.e., the closed trail $v = v_0e_0v_1e_1v_2e_2\ldots e_{(n-1)}\ v_n = v$ has a cycle. If all the vertices of that trail are distinct, then it is a cycle of length n. If not, there exist at least two vertices i,j such that $0 \leq i < j \leq n$ and $v_i = v_j$. Now, if we remove all the vertices (excluding v_i) and edges which belong to the trail between v_i and v_j, then $v = v_0e_0v_1e_1\ldots e_{(i-1)}\ v_i\ e_j\ v_{(j+1)}\ e_{(j+1)}\ v_{(j+2)}\ e_{(j+2)}\ldots e_{(n-1)}\ v_n = v$ is a closed trail of length less than n which, by the induction hypothesis, contains a cycle. Hence the theorem is proved.

2.3.6 Subgraphs

A graph $S = (E_1,V_1)$ is said to be sub-graph of a graph G = (V,E) if $E_1 \subset E$ and $V_1 \subset V$, where the incidence property is preserved between G and S. In graph theory, an induced subgraph of a graph is another graph, formed on a subset of the vertices of the graph and all of the edges are connecting pairs of vertices in that subset.

Formally, let G = (V, E) be any graph, and let $S \subset V$ be any propersubset of vertices of G. Then the induced subgraph G[S] is the graph whose vertex set is S and whose edge set consists of all of the edges in E that have both endpoints in S. The induced subgraph G[S] may also be called the subgraph induced in G by S, the induced subgraph of S.

A graph constructed by adding vertices, edges or both to a given graph is called supergraph. If H is a subgraph of G, then G is a supergraph of H.

A spanning subgraph is a subgraph that contains all the vertices of the original graph.

Example 2.6: In Figure 2.8(a), a graph G = (V, E) where V = {A, B, C, D, E, F, G, H, I}is considered. In Figure 2.8(b), a subgraph is shown whose vertex set is a subset of V. In Figure 2.8(c) an induced subgraph is drawn, and in Figure 2.8(d) a spanning subgraph is shown.

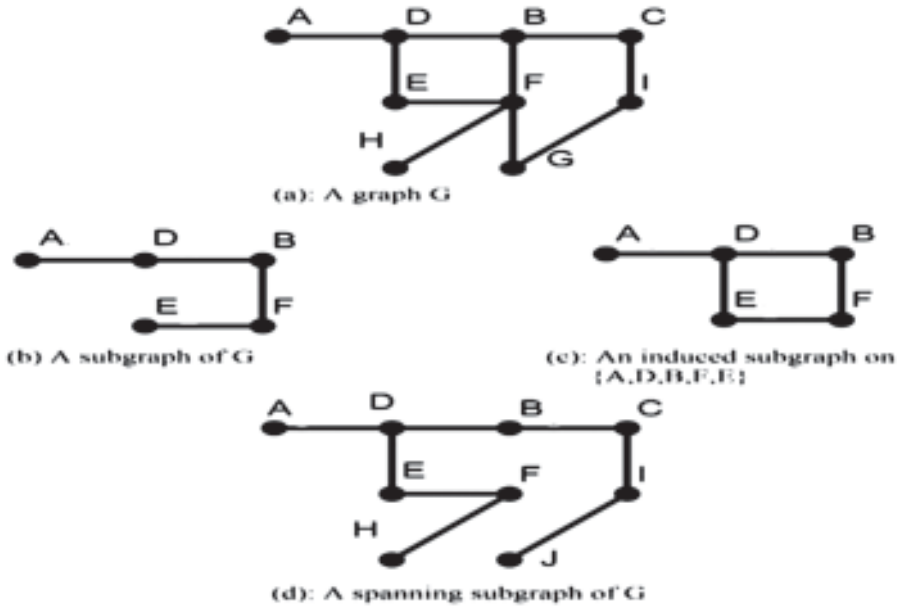


Figure 2.8: Different subgraphs of a graph G

Example 2.7: In Figure 2.9, G is a graph, G_1,G_2 and G_3 are subgraphs. In which G_3 is induced subgraph on the vertices $\{v_1,v_2,v_4,v_5\}$.

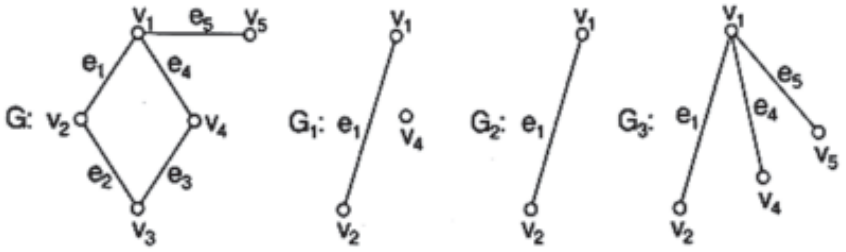


Figure 2.9: Graphs and its sub-graphs

2.4 Summary

This unit helps the learners to understand the fundamental concepts and basic terminology of graph theory with some illustrated examples.

Unit 3 ❑ Connected Graphs

Structure

3.0 Objective

The learners will be acquainted with some well-known graphs going through this unit.

3.1 Introduction

The concept of connectedness of graphs is a very important and basic concept. Other related concept are elaborated later.

3.2 Connected Graphs

An undirected graph with non-empty vertex set is connected when there is a path between every pair of vertices. In a connected graph, there are no unreachable vertices. A graph that is not connected is disconnected. An undirected graph G is said to be disconnected if there exist two nodes in G such that no path in G has those nodes as end points. A graph with just one vertex is connected. An edgeless graph with two or more vertices is disconnected.

3.3 Component

Ingraph theory, a component of an undirected graph is a subgraph in which any two vertices are connected by paths, and which is connected to no additional vertices in the supergraph, in other words, a component is a maximal connected subgraph. Thus a disconnected graph has at least two-component.

Example 3.1: In Figure 3.1, two components of a graph have been shown. It is noted that the subgraph containing the vertices v_1v_2,v_3 is not properly contained in any other subgraphs of the graph.

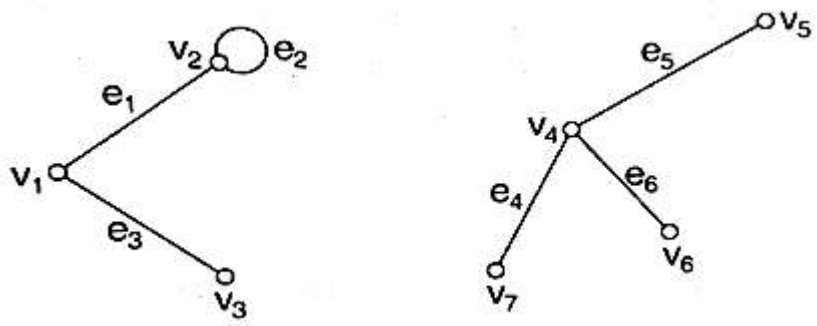


Figure 3.1: Components of a graph

3.4 Directed Graphs

In graph theory, a directed graph (or digraph) is a graph that is composed of a set of vertices connected by edges, where the edges have a direction associated with them.

Definition 3.1 A directed graph is an ordered pair G = (V, E) where

i. V is a set whose elements are termed as vertices, nodes, or points,

ii. E is a set of ordered pairs of vertices, termed as arrows, directed edges, directed arcs, or directed lines.

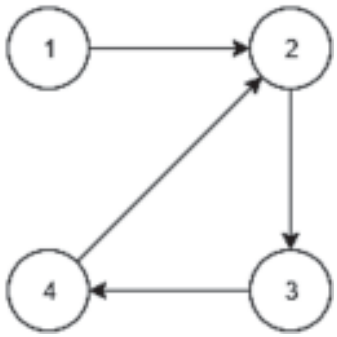


Figure 3.2: A directed graph

3.5 The complement of a graph

In the field of graph theory, the complement of a graph G is a graph H on the same vertices such that two distinct vertices of H are adjacent if and only if they are not adjacent in G.

Definition 3.2: The complement G^c of a simple graph G = (V, E) is the simple graph with the same vertex set V and edge set $E(G^c)$ defined by $(u,v) \in E(G^c)$ if and only if $(u,v) \notin E(G)$.

Example 3.2: In Figure 3.3, a graph and its complement graph are shown. It is seen that there is an edge between two vertices in the complement graph if there does not exist an edge between those vertices.

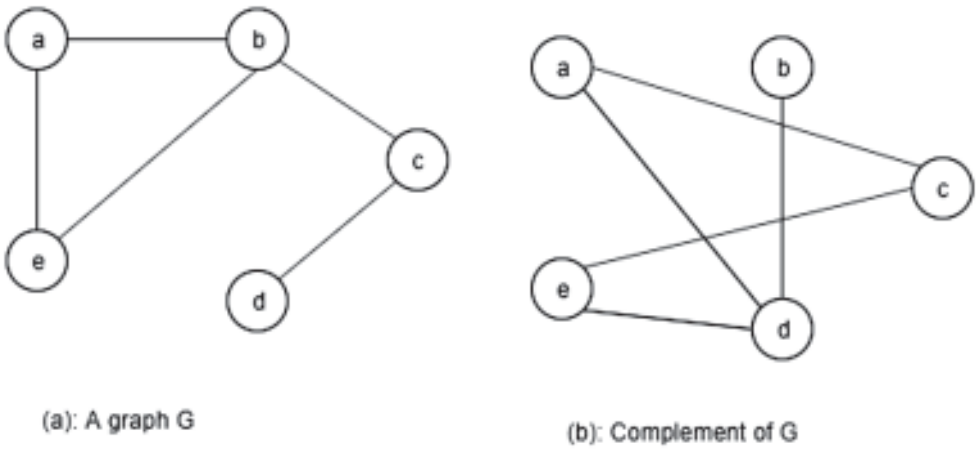


Figure 3.3: Complement of a graph

3.6 Line graphs

In graph theory, the line graph of an undirected graph G is another graph L(G) that represents the adjacencies between edges of G.

Definition 3.3: The line graph of a graph G=(V,E), L(G), is the graph whose vertices are the edges of G and there exists an edge (e,f) in the line graph if e=(u,v) and f=(v,w) are adjacent edges in G.

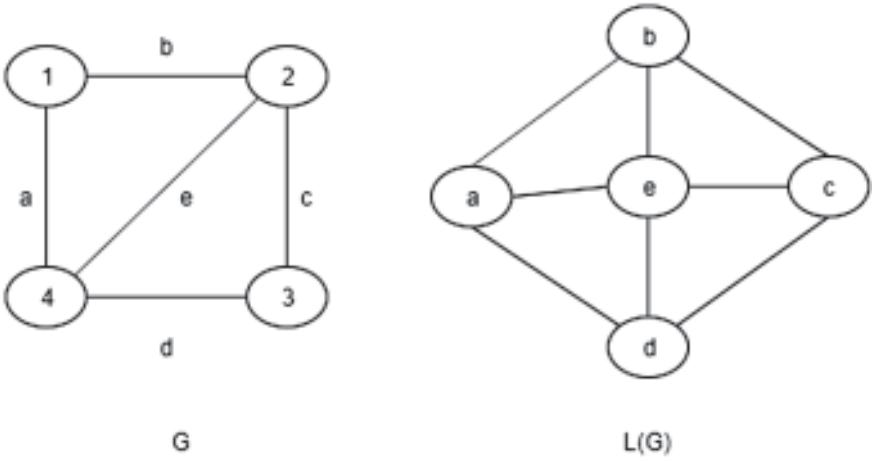


Figure 3.4: Line graphs

Example 3.3: In Figure 3.4, a graph G is shown. Also, its line graph L(G) is drawn in on the right side of Figure 3.4. Note that the number of edges in G is equal to the number of vertices in L(G).

3.7 Some well-known graphs

3.7.1 Complete graphs

In the field of graph theory, a complete graph is a simple undirected graph in which edges connect every pair of distinct vertices.A complete digraph is a directed graph in which every pair of distinct vertices is connected by a pair of edges (two for two directions). A cliquein a graph is a set of pairwise adjacent vertices.

Note that a complete graph on n vertices must have ${}^{n}c_{2}$ number of edges.

Figure 3.5: Complete graph on 4 vertices (left) and 5 vertices

Example 3.4: Complete graph on four vertices (K_4) and complete graph on five vertices (K_5) have been shown in Figure 3.5.

3.7.2 Bi-partite graphs

In graph theory, an independent set of vertices is a collection of isolated vertices, i.e., no vertex in the set is connected to any other vertex in the set. A bipartite graph (or bi-graph) is a graph whose set of vertices can be divided into two disjoint and independent subsets U and V such that every edge connects a vertex in U to one in V. Vertex sets U and V are usually called the parts of the graph. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles. Suppose one vertex (say a) of a cycle belong to an independent set, then the vertex must be connected to a vertex (say b) of separate independent set. Hence, b must be connected to a vertex (say c) of first independent set. Here a and c must be different and must not be connected as they belong to the same independent set. Thus, the minimum length of the cycle is four, if greater then adding by even number to it.

In the field of graph theory, a complete bipartite graph or biclique is a unique kind of bipartite graph where every vertex of the first set is connected to every vertex of the second set.

A complete bipartite graph is a graph whose vertices can be partitioned into two subsets V_1 and V_2 such that no edge has both endpoints in the same subset, and every possible edge that could connect vertices in different subsets is part of the graph. That is, it is a bipartite graph (V_1,V_2,E) such that for every two vertices $v_1 \in V_1$ and $v_2 \in V_2$, $v_1 v_2$ is an edge in E. A complete bipartite graph with partitions of size $|V_1| = m$ and $|V_2| = n$ is denoted $K_{(m,n)}$.

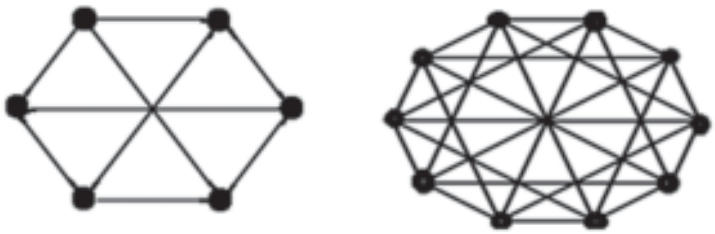


Figure 3.6: Complete bipartite graph, $K_{3,3}$ (left) and $K_{5,5}$

Example 3.5: A complete bipartite graph on six vertices $K_{3,3}$ and another complete bipartite graph on ten vertices $K_{5,5}$ have been shown in Figure 3.6.

Note that a complete bipartite graph on m+n number of vertices must have m×n=mn number of edges.

3.7.3 Regular graphs

In the mathematical field of graph theory, a regular graph is a graph where each vertex has the same number of neighbours, i.e. every vertex has the same degree. A regular directed graph must also satisfy the stronger condition that the indegree and outdegree of each vertex are equal to each other. A regular graph with vertices of degree k is called a regular graph of degree k or k regular graph.

Note that in the field of graph theory, a cubic graph is a graph in which all vertices have degree three. In other words, a cubic graph is a 3-regular graph. Cubic graphs are also called trivalent graphs.

Example 3.6: In Figure 3.7 (a) a 0-regular graph, Figure 3.7 (b) a 1-regular graph, Figure 3.7(c) a 2-regular graph and in Figure 3.7(d) 3-regular graph have been shown.

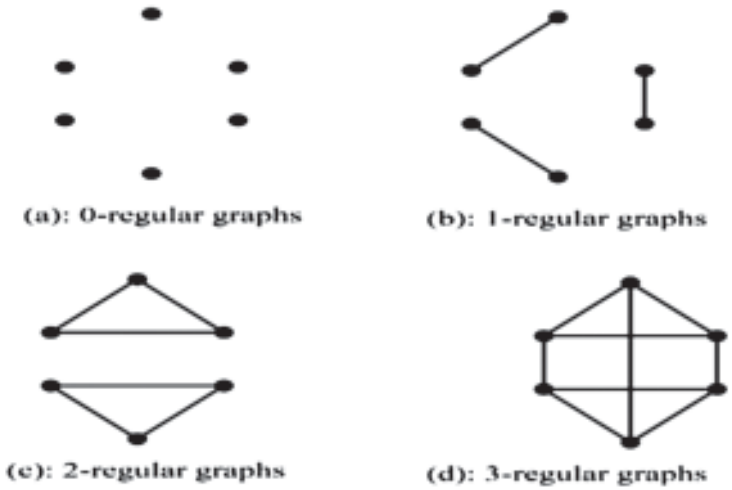
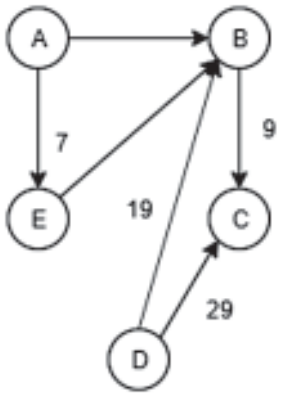


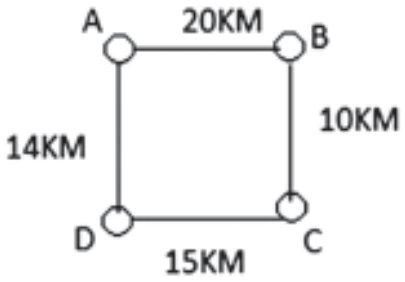
Figure 3.7: Regular graphs

3.7.4 Weighted graphs

A weighted graph is a graph in which each edge is given a numerical weight. If the graph represents any city routes, then weights may be its distance from a source to destination or may be a time of the travelling and many more. Thus the weight of the edges is significant to represent real-world scenario.



(a): Weighted directed graph



(b) Weighted undirected graph

Figure 3.8: Weighted graphs

Example 3.7: In Figure 3.8(a) a weighted directed graph is shown. The edge weights are written beside the edges in the figure. In Figure 3.8(b) a weighted undirected graph is shown. The numbers beside the edges may indicate the distance between the nodes which may represent the cities. The weights may be given on different parameters.

3.8 Isomorphism of graphs

3.8.1 Homeomorphism

Two graphs G_1 and G_2 are said to be homeomorphic if each of these graphs can be obtained from the same graph 'G' by dividing some edges of G with more verticesor inserting one or more vertices in an edge.

Example 3.8: Let us consider a graph G as shown in Figure 3.9. Divide the edge (r,s) into two edges by adding one vertex t (check Figure 3.9)

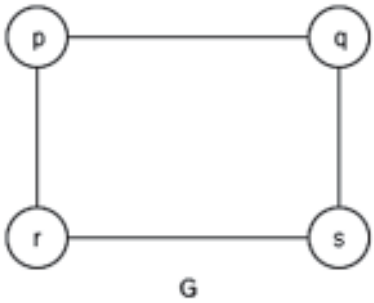


Figure 3.9: A simple graph

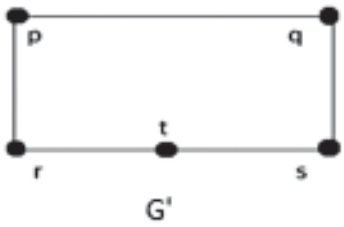


Figure 3.10: A simple graph dividing an edge of Figure 3.9 by a vertex

Now the graph G' can be embedded in a different position as like in Figure 3.10. The graphs shown in Figure 3.11 are homeomorphic to the first graph.

Figure 3.11: Homeomorphic graphs

3.8.2 Isomorphism

A graph can exist in various forms having the same number of vertices, edges, and also the equal edge connectivity. Such graphs are named isomorphic graphs. The formal definition of isomorphic graphs is given as follows.

Definition 3.4: An isomorphism from a simple graph G_1 to a simple graph G_2 is a bijection $f:G_1 \rightarrow G_2$ such that $(u,v) \in E(G_1)$ if and only if $(f(u),f(v)) \in E(G_2)$. It is said that G_1 is isomorphic to G_2.

Note that

If $G_1 \cong G_2$ then?

i. $|V(G_1)| = |V(G_2)|$

ii. $|E(G_1)| = |E(G_2)|$

iii. Degree sequences of G_x and G_2 are the same.

If the vertices $\{V_1,V_2,..V_k\}$ form a cycle of length K in G_1, then the vertices $\{f(V_1), f(V_2),\ldots f(V_k)\}$ should form a cycle of length K in G_2.

All the above conditions are necessary for the graphs G_1 and G_2 to be isomorphic, but not sufficient to prove that the graphs are isomorphic.

Example 3.9: In the graph G_3, vertex 'w' has only degree 3, whereas all the other graph vertices have degree 2. Hence G_3 is not isomorphic to G_1 or G_2.

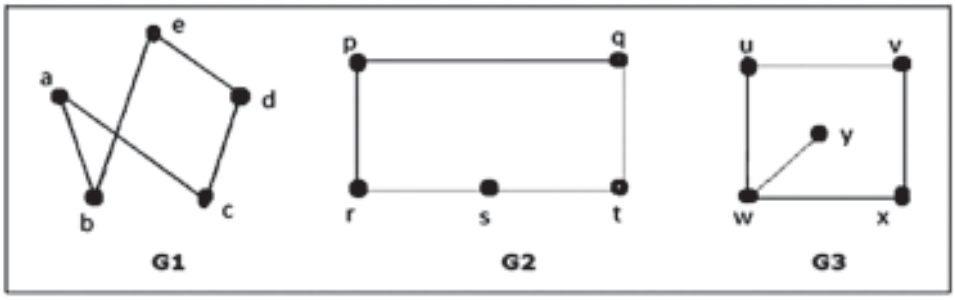


Figure 3.12: Isomorphism of graphs

Taking complements of G_1 and G_2, we have

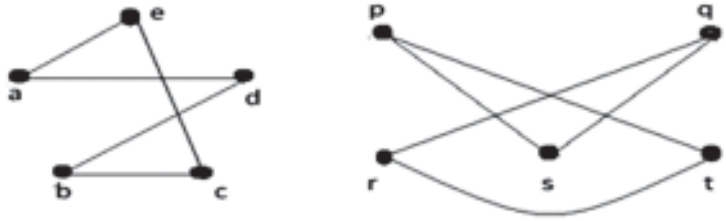


Figure 3.13: Isomorphic graphs

Here, $(G_1^c \equiv G_2^c)$, hence $(G_1 \cong G_2)$.

3.9 Summary

Some important concepts starting from connectivity to isomorphism along with Bipartite graph and weighted graphs are explained with examples in this unit.

Unit 4 ❑ Planar graphs

Structure

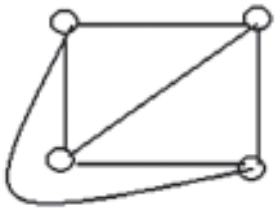
4.0 Objective

This unit gives a basic concept of planarity of graphs briefly.

4.1 Introduction

An embedding of a graph into a surface is a drawing of the graph on the surface in such a way that its edges may intersect only at their endpoints. In the field of graph theory, a planar graph is a graph that can be drawn in a plane in such a way that its edges intersect only at their end vertices. A graph can be drawn in several ways. In certain position, the edges may cross to other edges, but if the graph is drawn without any crossing among edges in atleast one embedding, then the graphs are called planar graphs.

(a): Planar graph (left) and non-planar graphs



(b): A planar graph on four vertices

Figure 4.1: Planar graphs

Example 4.1: In Figure 4.1 (a), a bipartite graph on six vertices and eight edges is shown. This graph has no crossing of edges. Thus it is a planar graph. But the complete bipartite graph on six vertices is not a planar graph, which is shown in the right part of Figure 4.1 (a). In Figure 4.1 (b), a planar graph is shown on four vertices (which is a complete graph on four vertices).

Euler's formula states that if a finite, connected, a planar graph is drawn in the plane without any edge intersections, and v is the number of vertices, e is the number of edges and f is the number of faces (regions bounded by edges, including the outer, infinitely large region), then v–e+f=2

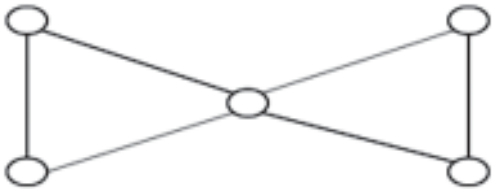


Figure 4.2: Butterfly graph

As an illustration, in the butterfly graph given above, v = 5,e = 6 and f = 3. In general, if the property holds for all planar graphs of f faces, any change to the graph that creates a new face while keeping the graph planar would keep v - e + f an invariant. Since the property holds for all graphs with f = 2, by mathematical induction, it holds for all cases. The result can be described as follows. Let G is connected and has only one face. It is a tree, so e=v-1 and therefore v+f-e=v+1-(v-1)=2.

Now, suppose the formula holds for a connected graph with n faces. We prove that it holds for n+1 faces. Choose an edge connecting two faces of G and remove

it. The resulting graph remains connected. The new graph has one fewer edge and one fewer face. So, by the inductivehypothesis, v+(f-1)-(e-1)=v+f-e=2.

4.2 Kuratowski's Two Graphs

The complete graph K_5 and the complete bipartite graph $K_{3,3}$ are said Kuratowski's graphs, after the Polish Mathematician Kasimir Kurtatowski, who found that K_5 and $K_{3,3}$ are non-planar.

Note thatthe complete graph K_5 with five vertices is non-planar. The following diagram can clarify this note. As we know that a complete graph with five vertices must have 10 edges, the graph with five vertices is started to draw. It is found that without crossing, 10 edges can not be drawn in the graph in any embedding. Thus the graph is non-planar.

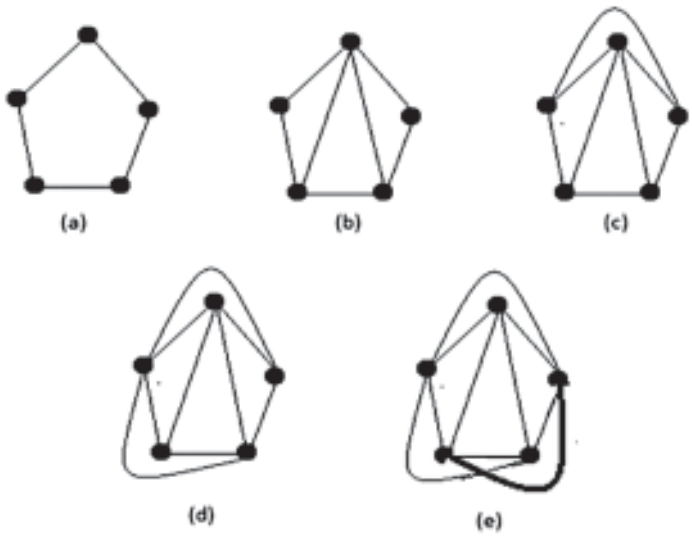


Figure 4.3: The steps of a drawing of planar graphs on five vertices

Note that the complete bipartite graph $K_{3,3}$ is non-planar : $K_{3,3}$ has six vertices and nine edges. In Figure 4.4, it is seen that $K_{3,3}$ cannot be drawn without crossing

between edges in any embedding. And hence $K_{3,3}$ is a non-planar graph.

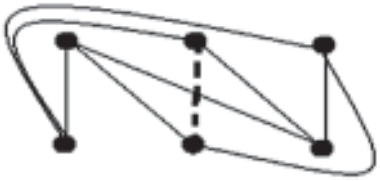


Figure 4.4: Bipartite graph embedding

Note 4.1 In a planar embedding of a connected graph with at least three vertices, each face is of length at least three. Suppose a connected planar graph has $n \geq 3$ vertices and e edges. Then $e \leq 3n-6$ is true. This is explained as follows. Suppose a connected graph with n vertices and e edges has a planar embedding with f faces. Then, every edge is traversed exactly twice by the face boundaries. So the sum of the lengths of the face boundaries is exactly 2e. Also when $n \geq 3$, each face boundary is of length at least three, so this sum is at least 3f. This implies that $3f \leq 2e$

But $f=e-n+2$ by Euler's formula, and substituting into the above gives $3(e-n+2) \leq 2e$

Which implies that $e \leq 3n-6$

4.3 Summary

The learners will have the fundamental idea of planar graphs and their types after studying this unit.

Unit 5 ❑ Matrix representation of graphs

Structure

5.0 Objective

The objective of this unit is to introduce the concept of representation of a graph using matrices.

5.1 Introduction

The graph theory has many practical applications as is described earlier. For many of them, graphs of large dimensions are required to be handled and then we have to use computers for efficient handling. This unit describes how a graph can be represented by matrix in different ways which can be easily used in computers.

5.2 Adjacency matrix

Suppose that G = (V,E) is a simple graph where |V| = n. Suppose that the vertices of G are listed arbitrarily as $v_1, v_2, \ldots, v_n$. The adjacency matrix A of G, with respect to this listing of the vertices, is the n × n zero-one matrix with 1 as its (i,j)-th entry when v_i and v_j are adjacent, and 0 as its (i,j)-th entry when v_i and v_j are not adjacent. In other words, if its adjacency matrix is $A = [a_{ij}]$, then

$$a_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \text{ is an edge of G} \\ 0, & \text{otherwise} \end{cases}$$

Note that an adjacency matrix of a graph is based on the ordering chosen for the vertices. Hence there are as many as n! different adjacency matrices for a graph with n vertices, since there are n! different orderings of n.

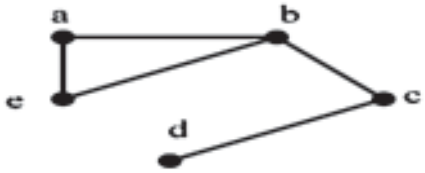


Figure 5.1: A simple graph

Example 5.1: Let us consider the graph in Figure 5.1. Its corresponding adjacency matrix is shown as follows. Note that the rows and columns denote the vertices of the graph.

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Also, note that the matrix is symmetric.

The following observations are made.

1. The entries along the principal diagonal of the matrix are 0 if and only if the graph has no self-loops. For self-loop, the entry will be 1.
2. The adjacency matrix is not defined for parallel edges.
3. If a graph has no self-loop, the degree of a vertex is equal to the number of 1 in the corresponding row or column of the matrix.
4. A graph G is disconnected if and only if its adjacency matrix can be partitioned into block matrices as follows

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

Where g_1 and g_2 are two components of G.

5.3 Incidence matrix

Let G be a graph with n vertices, m edges and without self-loops. The incidence matrix A of G is an n×m matrix X = [a_{ij}] whose n rows correspond to the n vertices and the m columns correspond to m edges such that

$$a_{ij} = \begin{cases} 1, \text{if jth edge } m_j \text{ is incident on the ith vertex} \\ 0, \text{ otherwise} \end{cases}$$

It is also called the vertex-edge incidence matrix and is denoted by A(G).

Example 5.2: Let us consider the graph of Figure 4.1 Its corresponding incidence matrix is shown as follows. **Note that** the rows denote the vertices {a,b,c,d,e} and columns denote the edges {(a,e),(a,b),(b,e),(b,c),(c,d)}of the graph.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Observations:

1. Each column in the incidence matrix contains only two '1' for a simple graph as each column represents an edge and an edge contains two end vertices.
2. Each row of the matrix may contain more than one '1'. The sum of such '1' is the degree of the corresponding vertex.
3. If all entries of a row are '0', it represents an isolated vertex.
4. The matrix is not symmetrical. It is a rectangular array.
5. Parallel edges indicate identical columns in the matrix
6. If a graph is disconnected into two components, then the incident matrix will be shown as block matrices as follows;

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

Where $g_1$and g_2 are two components of G.

Property: 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. In incidence matrix, '1' in a row indicate that the corresponding vertex in the column is incident with the corresponding edge representing the row. If two graphs are isomorphic, then their edge and vertex incidence are preserved. Thusrows and columns of the corresponding incidence matrices will only be differed by positions.

5.4 Worked out Examples (Unit 1-5)

1. Prove that the sum of the degrees of the vertices of any finite graph is even.

Solution: If we begin with just the vertices and no edges (i.e. starting with the null graph), every vertex has degree zero, so the sum of those degrees is zero, an even number. Now add edges one at a time, each of which connects one vertex to another, or connects a vertex to itself. Each edge is ended at two vertices. Either the degree of two vertices is increased by one (for a total of two) or one vertex's degree is increased by two. In either case, the sum of the degrees is increased by two, so the sum remains even.

2. The number of odd degree vertices is even in an undirected graph.

Solution: Let G be a graph with e edges and n vertices $v_1,v_2,v_3,...,v_n$. Since each edge is incident on two vertices (different or same), it contributes 2 to the sum of the degree of vertices in the graph G. Thus the sum of degrees of all vertices in G is twice the number of edges in G. Hence, $\sum_{i=1}^{n}\text{degree}(v_i) = 2e$

Let the degrees of first r vertices be even and the remaining (n-r) vertices have odd degrees, then clearly,

$$\sum_{i=1}^{r} \text{degree}(v_i) = \text{even number}$$

$$\Rightarrow \sum_{i=1}^{n}\text{degree}(v_i) = \sum_{i=1}^{r}\text{degree}(v_i) + \sum_{i=r+1}^{n}\text{degree}(v_i)$$

$$\Rightarrow \sum_{i=1}^{n} \text{degree}(v_i) - \sum_{i=1}^{r} \text{degree}(v_i) = \sum_{i=r+1}^{n} \text{degree}(v_i)$$

However, the for i = r + 1, r + 2,...,n each $d(v_i)$ is odd. So, the number of terms in $\sum_{1=r+1}^{n}\text{degree}(v_i)$ must be evenas the left-hand side of the above equation is even.

Thus (n–r) is an even number.

Hence the result is true.

3. Given a graph G with vertex set, $V = \{v_1,...,v_n\}$ we define the degree sequence of G to be the list $d(v_1),...,d(v_n)$ of degrees in decreasing order. For each of the following lists, give an example of a graph with such a degree sequence or prove that no such graph exists:

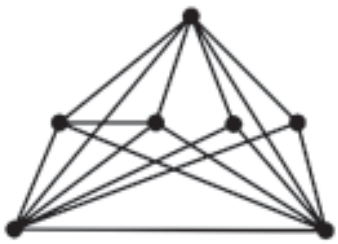
(a) 3, 3, 2, 2, 2, 1

(b) 6, 6, 6, 4, 4, 3, 3

Solution:

(a) There is no such graph since the number of odd-degree vertices in a graph is always even.

(b) Consider the following graph:



4. Do simple graphs with the following degree sequences exist:

(a) 6, 6, 6, 4, 4, 2, 2

(b) 6, 6, 6, 6, 5, 4, 2, 1?

Solution: (a) No, since otherwise, we have 3 vertices of degree 6 which are adjacent to all other vertices of the graph; so each vertex in the graph must be of degree at least 3.

(b) No! Note that each vertex of the degree 6 is adjacent to all but one other vertex. In particular, each such vertex is adjacent to at least one of v_1 and v_2 (where $d(v_1) = 1$ and $d(v_2) = 2$). However, that would mean at least four edges touching v_1 or v_2, contradicting $d(v_1) + d(v_2) = 3$

5. How many graphs exist on a given set of n vertices? How many of them contain exactly m edges?

Solution: Since there are C_2^n possible edges on n vertices and a graph may or may not have each of these edges, we get that there are $2^{C_2^n}$ possible graphs on n vertices. For the second problem, out of the C_2^n possible edges, we want to choose m ones. So there are $C_m^{C_2^n}$) possible graphs on n vertices and with m edges.

6. Show that a simple graph with n vertices must be connected if it has more than $\frac{(n-1)(n-2)}{2}$ edges.

Solution: Let G = (V,E) a simple graph with n vertices say $v_1,v_2,\ldots,v_{(n-1)},v_n$. Choose n–1 vertices $v_1,v_2,\ldots,v_{(n-1)}$ of G. Now, we can draw maximum $\frac{(n-1)(n-2)}{2}$ number of edges between these n–1 vertices. Therefore, if we have more than $\frac{(n-1)(n-2)}{2}$ number of edges, then at least one edge exists between v_n and some vertices among $v_1,v_2,\ldots,v_{(n-1)}$. Hence G must be connected.

7. Show that if u and v are the only two odd degree vertices of a graph G, then u and v are connected in G.

Solution: If G is a connected graph, then nothing to prove. Let G be disconnected. If possible assume that u and vare not connected. Then u and vlie in the different components of G. Hence the component of G containing u (similarly the component of G containing v) contains only one odd-degree vertex u which is not possible as each component of G is itself a connected graph and in a graph number of odd degree vertices should be even. Therefore u and v lie in the same component of G. Hence they are connected.

8. Every simple graph with n vertices and k edges has at least n-k components provided n>k.

Solution: A simple graph with n vertices and 0 edges has n components. By adding one edge, the number of components is reduced by 1. Similarly, by adding k edges, the components remain n-k.

9. (a) Is C_n a subgraph of K_n?

(b) For what values of n and m is $K_{(n,n)}$ a subgraph of K_m?

(c) For what n is C_n a subgraph of $K_{(n,n)}$?

Solution: (a) Yes! (left as an exercise)

(b) We have $|V(K_m)| = m\,|V'(K_{(n,n)})| = 2n$. On the other hand, by similar reasoning as part (a), we get that the statement holds for all m,n with m ? 2n.

(c) First, note that a bipartite graph cannot have any cycle of odd length, so n cannot be odd. For even n, one can check that $K_{(n,n)}$ has a cycle of length n.

10. A disconnected graph G with n even vertices has two components each of which is complete, prove that G has a minimum of $\frac{n(n-2)}{4}$ edges.

Solution: Let x be the number of vertices in one of the components. So, the other component has n-x number of vertices, since both componentsof G are complete graphs. Then, the total number of edges of G is

$$m = \frac{x(x-2)}{2} + \frac{(n-x)(n-x-1)}{2} = x^2 - nx + \frac{n(n-1)}{2}$$

Therefore, differentiating $\frac{dm}{dx} = 2x - n$ and $\frac{d^2m}{dx^2} = 2 > 0$

Now, for optimality, $\frac{dm}{dx} = 0 \Rightarrow x = \frac{n}{2}$

Therefore, min $m = \left[x^2 - nx + \frac{n(n-1)}{2}\right]_{atx=\frac{n}{2}} = \frac{n(n-2)}{4}$

11. Prove that a path with n vertices is of length n–1.

Solution : We know that in a path, every vertex except the last is followed by precisely one edge and no edge is used more than once. Therefore, if a path has n vertices, it must have n–1 edges.

12. Show that a cycle with n vertices has n edges.

Solution : In a cycle, every vertex is followed by precisely one edge. Therefore, if a circuit has n vertices, it must have n edges.

13. Prove that every cycle is a 2-connected graph.

Solution : We know that in a cycle exactly two edges are incident on every vertex. Therefore, every cycle is a 2-connected graph.

14. Show that every simple graph has two vertices of the same degree.

Solution : Assume that the graph has n vertices. Each of those vertices is connected to either 0,1,2,...,n–1 other vertex. If any of the vertices are connected to n–1 vertices, then it is connected to all the others, so there cannot be a vertex connected to others with degree 0. Thus it is impossible to have a graph with n vertices where one vertex has degree 0, and another has a degree n–1. Thus the vertices can have at most n–1 different degree, but since there are n vertices, at least two must have the same degree.

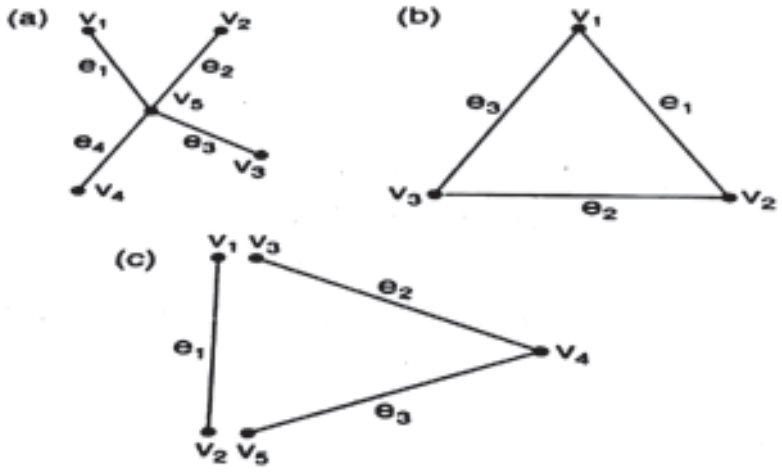
Alternate Solution: Suppose that the n vertices all have different degrees, and look at the set of degrees. Since the degree of a vertex is at most n ? 1, the set of degrees must be {0,1,2,...,n - 2,n - 1}. But that's not possible, because the vertex with degree n ? 1 would have to be adjacent to all other vertices, whereas the one with degree 0 is not adjacent to any vertex

Prove that a complete graph with n vertices contains n(n–1)/2 edges.

Solution : This is easy to prove by induction. If n = 1, zero edges are required, and 1(1–0)/2 = 0.

Assume that a complete graph with k vertices has k(k–1)/2. When we add the (k + 1)st vertex, we need to connect it to the k original vertices, requiring k new edges. We will then have k(k - 1)/2 + k = (k + 1)((k + 1) - 1)/2 vertices, and hence the proof is done by mathematical induction.

16. State which of the following graphs are bipartite graphs.



Solution: (a) The given graph is a bipartite graph as the vertex set can be partitioned into two disjoint independent vertex sets namely $\{v_1,v_2,v_3,v_4\}$and $\{v_5\}$. Also, vertices of one of these sets are connected to vertices of other sets.

(b) This graph is not a bipartite graph as the vertex set can not be written as a disjoint union of two independent sets.

(c) The given graph is a bipartite graph. The two disjoint independent sets are $\{v_1,v_3,v_5\}$ and $\{v_2,v_4\}$. Such that edges join from one set to another.

17. A graph is bipartite if and only if it has no odd cycle.

Necessary Part: Let G be a bipartite graph. Every closed walk alternates between two bipartite sets, so every return to the original set happens after an even number of steps. Hence, G has no odd cycle.

Sufficient Part: LetG be a graph with no odd cycle.

First, the result will be proved in the case where G is connected. Assume G has no odd cycle. Let $u \in V(G)$ be an arbitrary vertex and d(u,v) denotes the number of edges between the vertices u and v.

Also, let,

$X = \{v \in V(G): d(u,v) \text{ is even}\}$,

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

Clearly, $X \cap Y = \varnothing$ and $X \cup Y = V(G)$ since G is connected. We claim that X,Y is a bipartition of G. Suppose not - then there exists an edge incident to two vertices of X or an edge incident to two vertices of Y. Without loss of generality, assume the former.

Let $x_1,x_2 \in X$ and $x_1 \sim x_2$. It follows that

$$x1 \in X \Rightarrow \exists\ u,x_1\text{-path } P_1 \text{ of even length,}$$

$$x2 \in X \Rightarrow \exists\ u,x_2\text{-path } P_2 \text{ of even length.}$$

Concatenate u,x_1-path P_1, the edge $x_1\ x_2$, and the x_2,u-path P_2 to obtain a closed odd walk. Now it is obvious that the graph must contain an odd cycle. Hence, it must be the case the X,Y is a valid bipartition, so G is bipartite.

18. (a) How many edges do a planar graph with 5 vertices and 3 faces have.?

(b) How many vertices does a planar graph with 7 edges and 5 faces have?

Solution: Euler's formula is V + F = E + 2 where V is the number of vertices, F is the number of faces (regions), and E is the number of edges.

(a) There are E = V + F - 2 = 6 edges.

(b) There should be V = E – F + 2 = 4 vertices. However, this is not possible without creating duplicate edges. With duplicate edges, it is possible, and the formula gives the correct answer if we count the space between two duplicate edges also as a face. Note, however, that this is not a graph, but a multigraph.

19. What is the maximum number of edges in a bipartite planar graph with n vertices?

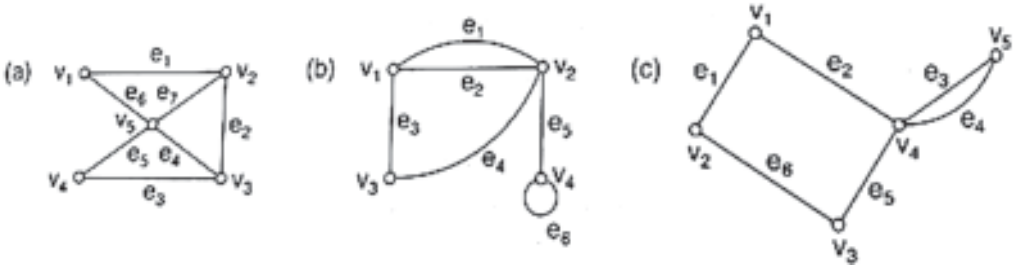
Solution: In a bipartite graph, every cycle has even length, so in particular, every face has an even number of edges. A face cannot have only 2 edges, because then there would be a double edge. A face with 6 edges can always be subdivided into two faces with 4 edges each (and the same is true for faces with even more edges), so in any planar bipartite graph with a maximum number of edges, every face has length 4. Since every edge is used in two faces, we have 4F = 2E. Plugging this into Euler's formula, we find that E = 2V – 4. Since n = V, the answer is 2n – 4 edges.

5.5 Summary

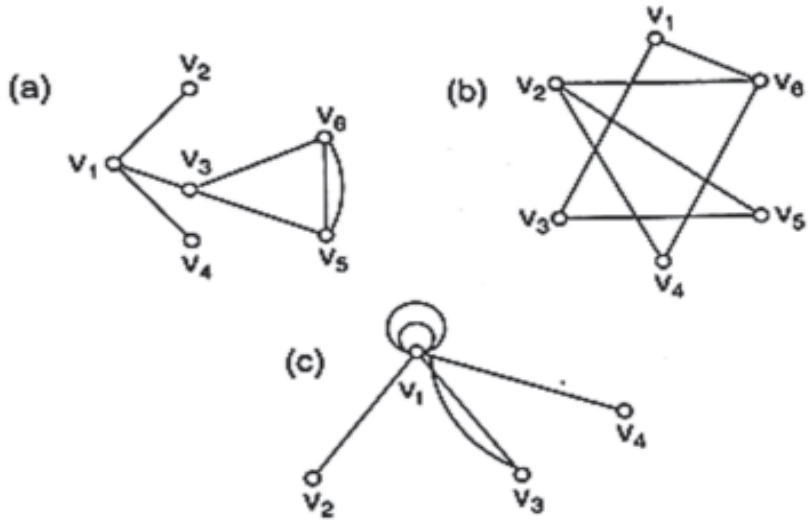
This unit gives an overview of different types of matrix representation of graphs which makes it easy to use in computers. With lots of examples given in this unit the learner will now easily grasp the subject.

5.6 Exercises (Unit 1-5)

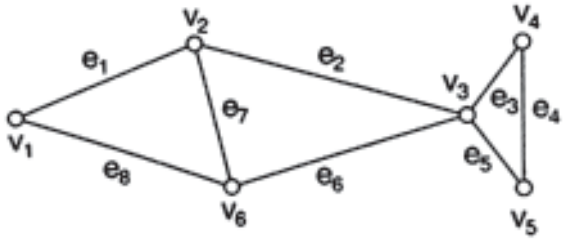
1. Draw all types of graphs on five vertices.
2. State which of the following graphs are simple.



3. Find the degrees of the vertices in the following graphs



4. Determine which complete bipartite graphs are complete graphs.
5. Prove that if every vertex of a simple graph has degree 2, then the graph is a cycle.
6. Find the maximum number of edges in a graph with p vertices and no even cycles.
7. Prove that $\sum_{i=1}^{n} d_i = 2e$ where d_i is the degree vertex v_i, n is the number of vertices and e is the number of edges.
8. Prove or disprove that the complement of a simple disconnected graph is a connected graph.
9. Prove or disprove that every self-complementary graph has 4n or 4n+1 number of vertices.
10. Determine whether the given walk is (a) a path, (b) a trail, (c) a closed walk, (d) a circuit, (e) a cycle.

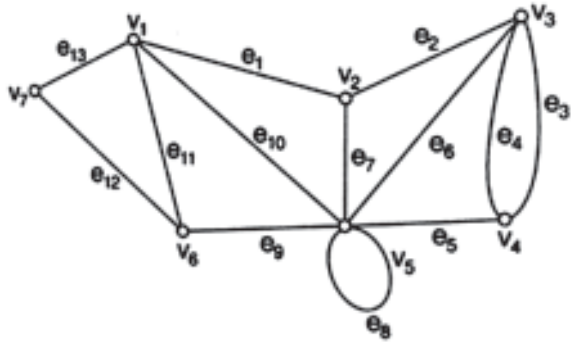


11. In the following graph, find

(a) a walk of length 5.

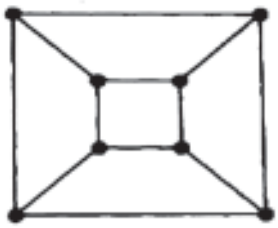
(b) a trail of length 9.

(c) a circuit of length 7.

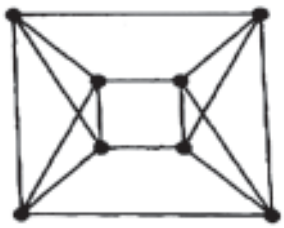


12. Prove that if G and H are isomorphic iff their complements are isomorphic.

13. Prove that the on the left side of the following figure is isomorphic to the complement of the graph on the right side of the figure.



(a)



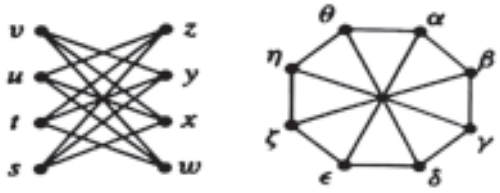
(b)

14. Prove that the complement of a simple disconnected graph must be connected.

15. Check the isomorphism of the following figures.

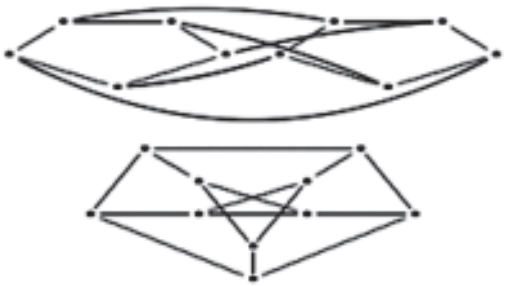
a. 

b.

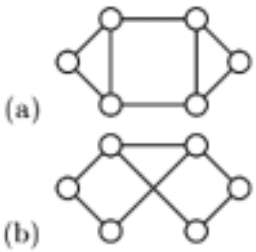


c.

d.



16. Is the following graph planar? Is it bipartite?



15. Prove that any two simply connected graphs with n vertices, all of degree two, are isomorphic.

16. Prove that a simple graph with n verticesmust be connected if it has more than (n-1)(n-2)/2 edges.

17. Draw a connected graph that becomes disconnected when any edge is removed from it.

18. A be an adjacency matrix of a simple graph G, then the ij-th entry of A^r is the number of different edge sequences of r edges between v_i and v_j.

Unit 6 ❑ Eulerian graphs and Hamiltonian graphs

Structure

6.0 Objective

In this unit, the details of Euler graphs, Hamiltonian graphs and their properties are described. This chapter concludes with the famous Konigsberg Bridge problem that all edges will be traversed or not.

6.1 Introduction

The Konigsberg Bridge Problem is one of the famous problems in graph theory. This problem was unsolved until 1736. In the eighteenth century, Konigsberg was the capital of East Prussia but now renamed as Kaliningrad and transferred to West Soviet Russia. The Pregel River flowed through town and split into two branches and formed two islands, C and D, as shown in Fig 6.1(a). There were seven bridges crossed the river and linked the four landmasses labelled by A, B, C, D, as shown in Figure 6.1(a). The problem was if it is possible to start on one of the four landmasses, walk over each of the seven bridges exactly once, and return to the starting point (without swimming across the river). Famous Swiss mathematician Leonhard Euler solved this problem in 1736. To solve this problem, Euler modelled a graph, as shown in Figure 6.1(b), corresponding to this real problem where vertices represent the landmasses and edges represent the bridges.

6.2 Eulerian circuits and Eulerian graphs

Now, Konigsberg Bridge Problem became that if it is possible to start on one of the four vertices, walk over each of the edges exactly once, and return to the starting vertex, i.e., is it possible to find a circuit which is passing through each edge of the graph exactly once (allowing vertex repetition). This circuit may not include all vertices of the graph because if some isolated vertices are present in the graph they do not have any contribution to find the circuit. Euler also raised (and then solved) a new question of what type of graphs are suitable for finding this type of circuit.

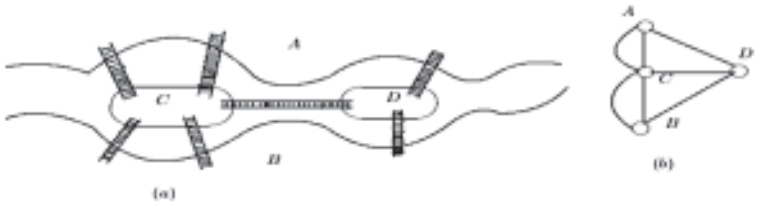


Figure 6.1: Geographic map of the Konigsberg bridge problem and its graph.

So, without any loss of generality, we can neglect the isolated vertices and say that the graph in which this type of circuit exists is a special type of connected graph. This type of circuit is known as the Eulerian circuit. An Eulerian circuit of a connected graph G is a circuit which passes through all edges of G exactly once.Note that vertices may repeat in an Eulerian circuit. A graph is called anEulerian graph if it contains an Eulerian circuit. Eulerian circuits and Eulerian graph are named, of course, after Leonhard Euler who solved the Konigsberg Bridge Problem. The graph as shown in Figure 6.2 is an Eulerian graph.

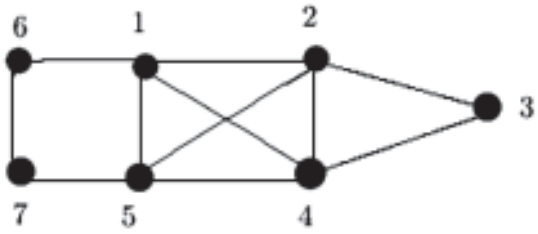


Figure 6.2: An Eulerian graph.

Now, if we observe the Eulerian circuit, we see that at every vertex other than the common starting and ending vertex, we come into the vertex along one edge and go out along another; this can happen more than once, but since we traverse all edges

exactly once, the number of edges incident at such a vertex must be even. The common starting and ending vertex may be visited more than once; except for the very ?rst time we leave the starting vertex, and the last time we arrive at the vertex, each such visit uses exactly two edges. Together with the edges used ?rst and last, this means that the starting vertex must also have even degree. For this reason, the graph corresponding to the since the Konigsberg Bridge Problem is not an Eulerian graph (it's all vertices have odd degrees). So, the question should arise in our mind immediately that if a graph is connected and all vertices have even degrees, is there an Euler circuit? The answer is yes.

Example 6.1: In Figure 6.3, $v_1,e_1,v_2,e_2,v_3,e_3,v_1,e_4,v_4,e_5,v_5,e_6,v_1$ is an Euler circuit.

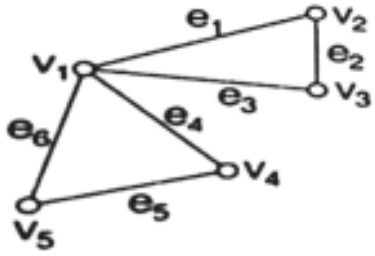


Figure 6.3: A simple graph

Example 6.2: In Figure 6.4, a graph is shown such that each vertex is of even degree. Hence, anEuler circuit can be found. The Euler circuit is given as follows: $v_2,e_6,v_4,e_3,v_3,e_2,v_2,e_5,v_6,e_4,v_4,e_7,v_5,e_8,v_1,e_1,v_2$

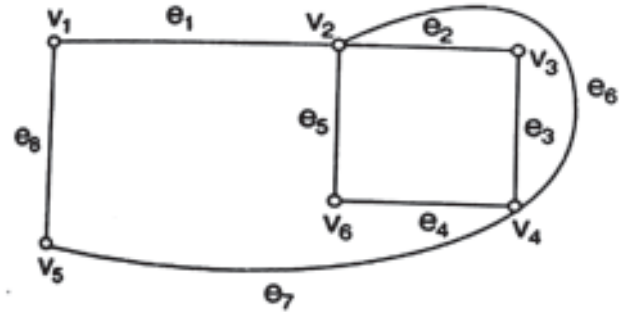


Figure 6.4: A simple graph.

Theorem 6.1: A connected graph G is an Eulerian graph iff every vertex of G has even degree.

Proof: Sufficient part: Let G is a connected finite graph with all vertices of even degree. If G has only one vertex, then the problem is trivial, so we assume that G has edges. We have to prove that G has an Eulerian circuit. Let v be any vertex of G. If there are any loops incident with v, traverse these first, one after the other without repetition. Since we are assuming that G is connected and has edges, there exists at least one vertex v_1 ($\neq$ v) which is adjacent to v. If there are any loops incident with v_1, traverse these ones after the other without repetition. Again, since all vertices have even degrees, there exists at least one vertex v_2 ($\neq v_1$,v) which is adjacent to v_1. Thus we have a trail from v to v_2 which we continue if possible. Each time we arrive at a new vertex through an edge which is not visited before, traverse all the loops (incident at that vertex) without repetition. Since the degree of each vertex is even, it is always possible to leave a vertex apart from v at which we arrive, until we return to the starting vertex v, and every edge incident with the starting vertex has been used. The sequence of vertices and edges formed in this way is a closed-circuit say C. If C uses every edge, we are done, i.e., C is an Eulerian circuit as every vertex in C is of even degree since each time we entered and left each vertex through different edges. If C is not an Eulerian circuit, then we delete from G all the edges of C and all the vertices of G which are become isolated (that is, acquire degree 0) by this procedure. All vertices of the remaining graph say G_1 are even (since both G and C have vertices of even degrees) and of positive degree. Also, G_1 and C have at least one common vertex say u because G is connected. Now, starting at u, and proceeding in G_1, using the same traversing procedure as we applied in G, we construct a circuit C_1 in G_1 which returns to u. Now, starting at v, moving along C to u, then traversing through C_1 back to u, and then traversing through C back to v. Thus we construct a circuit say C_2 in G which contains more edges than C_1. If it contains all the edges of G, it is an Eulerian circuit; otherwise, we repeat the previous process to construct a sequence of larger and larger circuits. Since our graph is finite, the process must eventually stop, and it stops only with a circuit which passing through all edges exactly once, i.e., with an Eulerian circuit. Therefore, the graph G is an Eulerian graph.

Necessary part: Let the connected graph G is an Eulerian graph. So, G contains an Eulerian circuit. Since the graph G is connected, so the Eulerian circuit not only passing through all the edges exactly once but also includes all the vertices. Also, it is observed that at every vertex other than the common starting and ending vertex, we come into the vertex along one edge and go out along another; this can happen

more than once, but since we traverse all edges exactly once, the number of edges incident at such a vertex must be even. The common starting and ending vertex may be visited more than once; except for the very first time we leave the starting vertex, and the last time we arrive at the vertex, each such visit uses exactly two edges. Together with the edges used first and last, this means that the starting vertex must also have even degree. Hence all vertices have even degrees.

Now, looking back to the graph of the Konigsberg problem (Figure 2.4(b)), we see that there is no even vertex. Hence, this graph is not an Eulerian graph. Therefore it is not possible to start on one of the four landmasses, walk over each of the seven bridges exactly once, and return to the starting point (without swimming across the river).

Theorem 6.2: If G is a graph in which the degree of every vertex is at least two, then G contains a cycle.

Proof: Let G be a graph in which the degree of every vertex is at least two. If G has any self-loops or multiple edges, the result is trivial. Suppose that G is a simple graph. Now, we choose any vertex v_0 of G. Since the degree of every vertex is at least two, so, there exists at least one vertex $v_1 \neq v_0$ such that v_1 is adjacent to v_0. Again, there exists at least one vertex $v_2 \neq v_0, v_1$ which is adjacent to v_1. Thus we get a path from v_0 to v_2 which we continue if possible. Now, at any stage, if the vertex $v_{i,i} \neq 2$ is already chosen, then we choose $v_{(i+1)}$ which is adjacent to v_i other than $v_{(i-1)}$. Since the degree of each vertex is at least 2, the existence of $v_{(i+1)}$ is always guaranteed. Since G has an only finite number of vertices, so, at some stage, we have to choose a vertex which has been chosen before. Let v_k be the first such vertex and let $v_k = v_i$ where $i < k$. Thus we get a closed cycle which is $v_i \rightarrow v_{(i+1)} \rightarrow ... \rightarrow v_k$ $(=v_i)$.

Hence the result is proved.

Theorem 6.3: If G is a non-trivial Eulerian graph, then set of edges of G can be partitioned into cycles.

Proof: Since G is a non-trivial Eulerian graph, so, every vertex is even and has a degree of at least 2. Therefore, by theorem 6.2, G contains a cycle C_1. The removal of the edges of C_1 forms a spanning sub-graph G_1 in which every point still has even degree. If G_1 has no edges, then the result is proved; otherwise, we repeat the same procedure (which we applied on G to get G_1) on G_1 to forms a graph G_2 in which again all vertices are even. This process will be continued until a disconnected graph G_n is obtained. Therefore, we have a partition of the edge set E of G into n cycles.

6.3 Hamiltonian Graphs and Hamiltonian cycles

A path in a connected graph G is called a Hamiltonian path if it traverses each vertex in G exactly once (starting and ending vertices are distinct, of course). A cycle in a connected graph G is called a Hamiltonian cycle if it traverses each vertex in G exactly once, except for the starting and ending vertex that appears twice. A connected graph G is called a Hamiltonian graph, if it contains a Hamiltonian cycle. A connected graph is called a traceable or Semi-Hamiltonian graph if it contains a Hamiltonian path. Such paths, cycles and graphs are named after famous Irish mathematician Sir William Rowan Hamilton, who discovered, in 1856, the icosian game, now also known as Hamilton's puzzle on the regular dodecahedron of wood, as shown in Figure 6.5, a solid with 12 congruent faces, each of which is a regular pentagon. The 20 corner points of the dodecahedron were marked with the names of 20 cities of the world, and the object of the game was to find a route (Hamiltonian cycle) "around the world," along the edges of the solid, which passed through each city exactly once and come back to the city where the tour started.Hamilton applied such cycles in his early investigations into group theory, the three edges incident to a vertex corresponding to three generators of a group. The graph of the dodecahedron is a Hamiltonian graph; a Hamiltonian cycle is indicated in Figure 6.5 by thick lines.

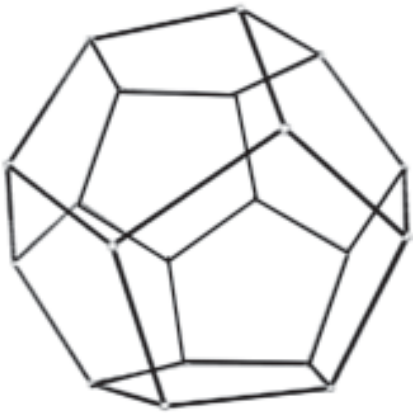


Figure 6.5: Dodecahedron.

On the other hand, the Herschel graph of Figure 6.5 is a non-Hamiltonian graph, because it is bipartite and has an odd number of vertices. This graph is, however, traceable, i.e., the Herschel graph contains a Hamiltonian path. Note that an Eulerian circuit traverses every edge exactly once, but may repeat vertices, while a Hamiltonian cycle visits each vertex exactly once. Also, if we remove anyone edge from a

Hamiltonian cycle, then we will get a Hamiltonian path. So, Hamiltonian cycle (or path) cannot include a self-loop or parallel edges. Thus a general graph may be made simple by removing parallel edges and self-loop before tracing a Hamiltonian cycle in it. Since a Hamiltonian path is a sub-graph of a Hamiltonian cycle, so, every graph that has a Hamiltonian cycle also has a Hamiltonian path. However, the converse is always not true. Obviously, not each connected graph has a Hamiltonian circuit. In this context, one question may arise in our mind that what is the necessary and sufficient condition for a connected graph to have a Hamiltonian circuit? Hamilton first proposed this question in 1859, which is still an unsolved problem in graph theory. While there is a criterion for determining whether or not a graph contains an Eulerian circuit, a similar criterion does not exist for a Hamiltonian cycle.

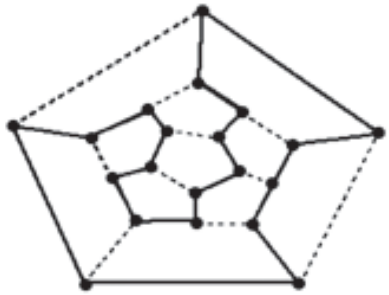


Figure 6.6: The graph of the dodecahedron of Figure 6.5.

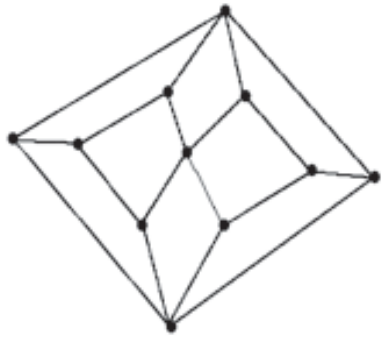


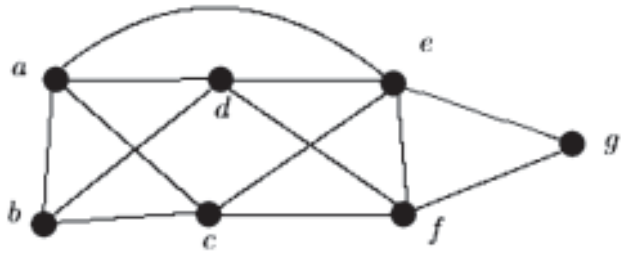
Figure 6.7: The Herschel graph.

6.4 Illustrative examples

1. **Which diagrams can be drawn without lifting one's pen from the paper not covering any line segment more than once?**

Ans. An Eulerian graph or semi-Eulerian graph.

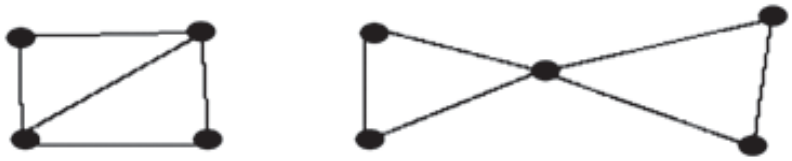
2. Check whether the following graph has is Eulerian or Semi-Eulerian or not?



Ans. We see that the given graph is connected and this graph has only two odd vertices which are b and e. All other vertices are of even degree. So, the given graph is a semi-Eulerian graph.

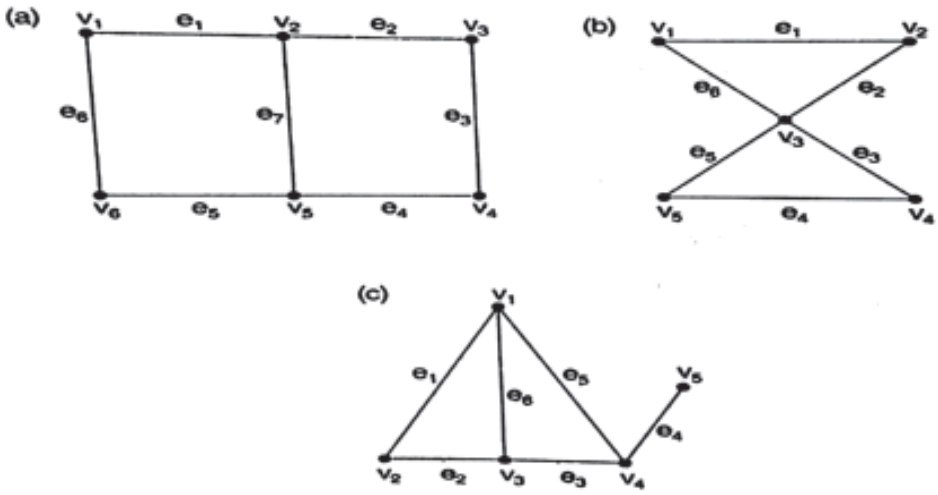
3. Give an example of a graph which is Hamiltonian but not Eulerian and vice versa.

Ans. The graph is shown in the following figure, part (a) is Hamiltonian but not Eulerian. Also, the graph as shown in part (b) is Eulerian but not Hamiltonian.



(*a*) (*b*)

4. Find out Euler trail in the following graph, if exists.

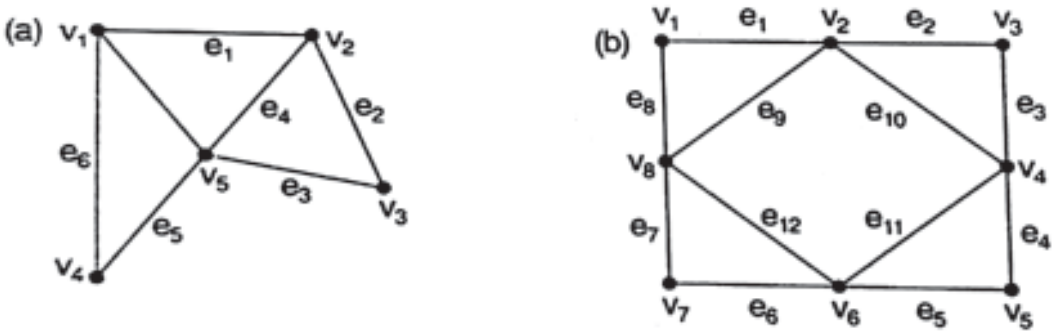


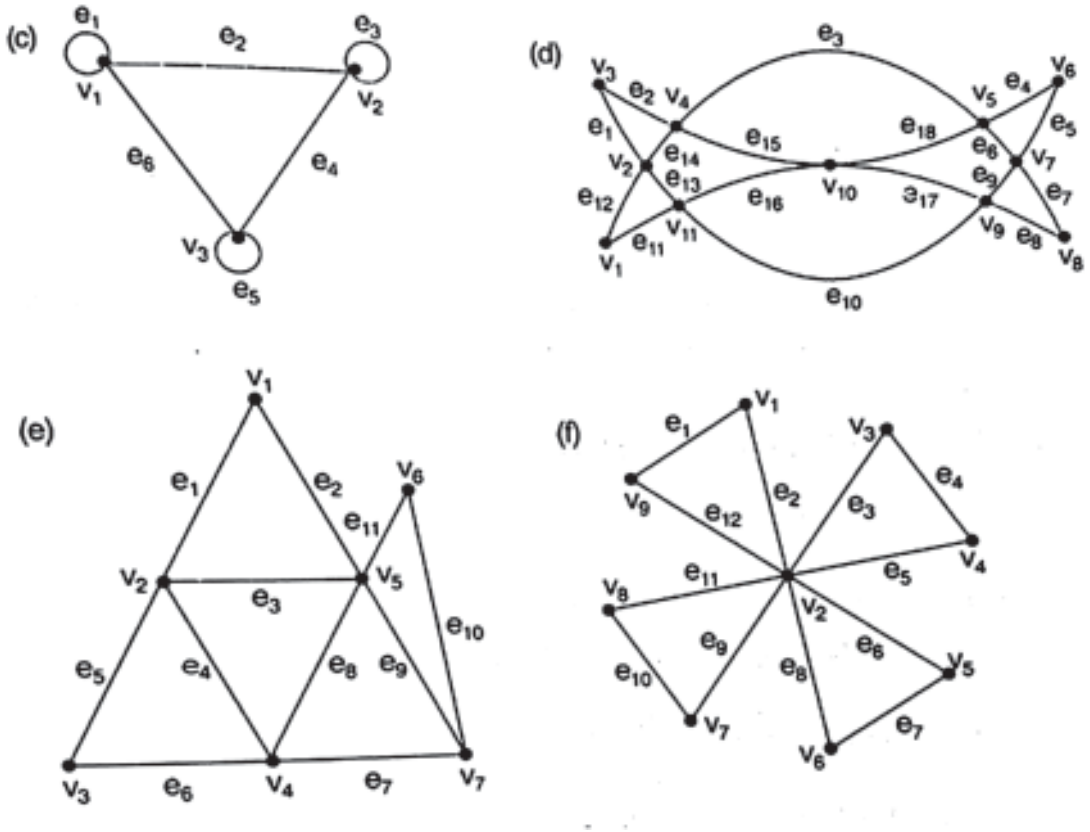
Ans. (a): Here v_2 and v_5 are of odd degree. Hence the Euler trail will start from one and end in another. The trail is as follows: $v_2,e_2,v_3,e_3,v_4,e_4,v_5,e_7,v_2,e_1,v_1,e_6,v_6,e_5,v_5$.

(b): The graph has no vertices of odd degree, hence it has no Euler Trail.

(c): The graph has more than two odd degree vertices, so the graph has no Euler Trail.

5. Find Euler circuits from the following graphs, if exists.





Ans.

(a): The graph has odd degree vertices. So Euler circuit does not exist in the graph.

(b): The given graph is connected and all vertices are of even degrees. Hence, a circuit can be found containing all the edges of the graph without repeatation. The Euler circuit is $v_1,e_1,v_2,e_2,v_3,e,v_4,e_4,v_5,e_5,v_6,e_6,v_7,e_7,v_8,e_9,v_2,e_{10},v_4,e_{11},v_6,e_{12},v_8,e_8,v_1$.

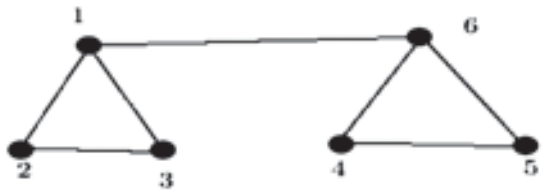
The remaing graphs are left for reader as assignments.

6.5 Summary

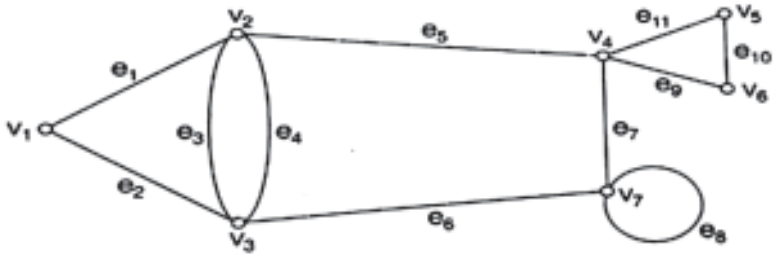
This unit describes the traversibility of a graph viz. Eulerian graph and Hamiltonian graph which has a wide range of applications.

6.6 Exercise

1. Show that if there is a xy-walk in a graph G, then there is also an xy-path in G.
2. Show that the equivalence classes determined by the relation of connectedness between vertices are precisely the vertex sets of the components of the graph.
3. Let G be a graph with two distinct non-adjacent vertices x and y, and let $G \cup \{e\}$ be the graph obtained from G by the addition of a new edge e=(x,y). Show that G has an open Eulerian trail connecting x and y if and only if $G \cup \{e\}$has an Euler circuit.
4. Show that the Herschel graph is non-Hamiltonian.
5. Draw a graph which is Hamiltonian but not Eulerian.
6. Examine whether a Hamiltonian path or cycle exists in the following graph.

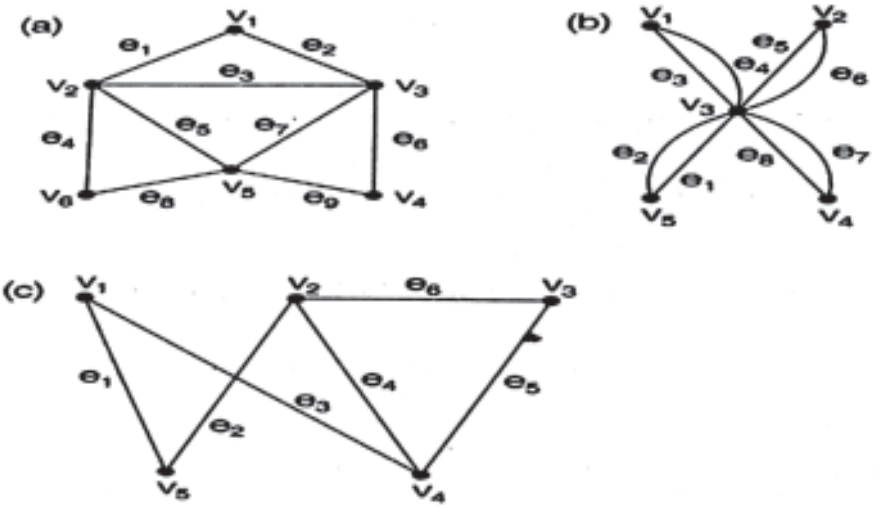


7. Show that in a complete graph $K_{(2n+1)}$ there are n edge-disjoint Hamiltonian cycles.
8. Write down the difference between circuit and cycle.
9. What is the difference between walk and trail?
10. Prove that there can no path longer than a Hamiltonian path (if exists) in a graph.
11. Draw a graph in which an Eulerian circuit is also a Hamiltonian cycle. What can you say about such a graph in general?
12. In the following figure find an eulerian path.

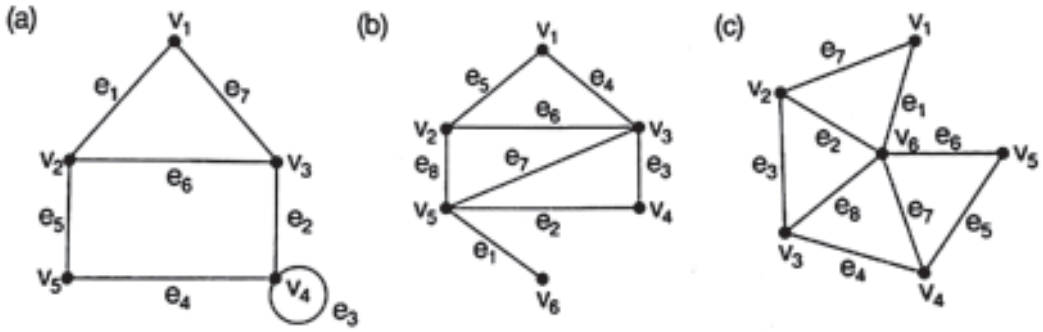


13. Prove that a connected graph remains connected after any edge is removed from the graph if and only if the edge belongs to any cycle.

14. Find Eluer circuit of the following graphs, if exists.



15. Describe whether the following graphs have an Euler trail. If the graph has an Euler trail, write one such trail.



Unit 7 ❑ Graph Algorithms

Structure

7.0 Objective

The objective of this unit is to introduce a special type of graph, tree, which is widely used. Then the subject is further developed by discussing some useful algorithms.

7.1 Introduction

Tree is a very important and interesting type of graph which has the property to represent the process of distributing mails by post offices based on PIN code, the family of a river with its tributaries and subtributaries or the genealogy of a family.

7.2 Tree

A tree is an undirected graph G that satisfies any of the following conditions:

i. G is connected and acyclic (contains no cycles).

ii. G is connected but would become disconnected if any single edge is removed from G.

iii. Any two vertices in G can be connected by a unique simple path.

If G has finitely many vertices, say n of them, then the above statements are also equivalent to any of the following conditions:

i. G is connected and has n - 1 edge.

ii. G has no simple cycles and has n – 1 edge.

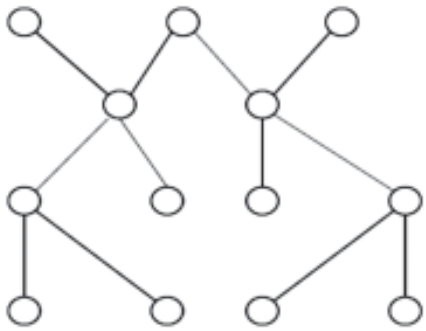


Figure 7.1: A tree (acyclic graph)

7.3 Travelling Salesman Problem

The travelling salesman problem is a problem in graph theory requiring the most efficient (i.e., least total distance) Hamiltonian cycle a salesman can take through each of n cities. No general method of solution is known, and the problem is NP-hard.

The travelling salesman problem (TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?"

A salesman is required to visit a number of cities during a trip. Given the distances between the cities, in what order should he/she travel so as to visit every city precisely once and return to the starting city, with the minimum mileage travelled?

The steps are given below:

Start from any city.

Move to nearest neighbouring city (i.e., a city with minimum distance)

Go to step 2, if all the cities are not visited.

Return to the starting city.

7.4 Minimum Spanning Tree

A graph without a cycle is called an acyclic graph. A tree is a connected acyclic graph. Moreover, A spanning tree is a spanning subgraph that is a tree. For a connected and undirected graph G, a spanning tree T of the graph is a subgraph that is a tree and includes all the vertices of G. A single graph can have many different spanning trees. A minimum spanning tree (MST) is a spanning tree T such that sum of the weight of all edges of T is less than or equal to the sum of weights of all edges of every other spanning tree of the graph G. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

Note that: A minimum spanning tree has (|V |- 1) edges where |V| is the number of vertices in the given graph.

Example 7.1: In Figure 7.2, G is a simple graph, T_1 and T_2 are spanning trees of G.

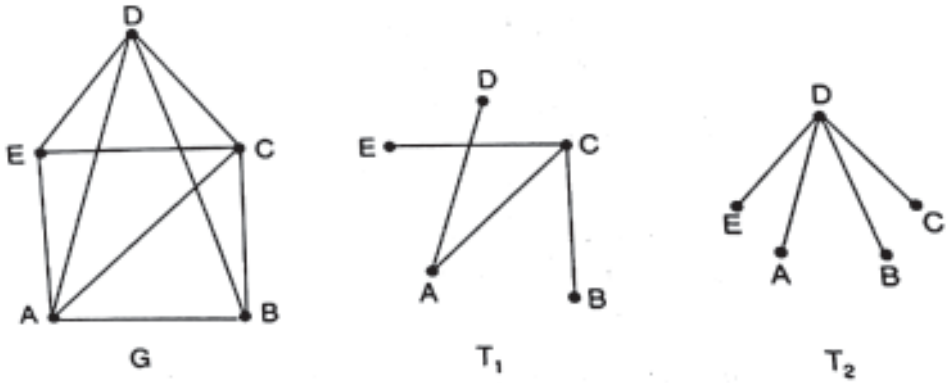


Figure 7.2: Spanning rees

7.4.1 Finding a Minimum Spanning Tree (MST) using Kruskal's algorithm

To obtain a minimum spanning tree of a graph, a novel approach was described by J.B. Kruskal known as Kruskal algorithm. The steps are given below.

Input: A weighted connected graph, G

Step 1: Sort all the edges in non-decreasing order of their weight.

Step 2: Create a set T where edges of spanning tree will be included. Initially.

$T = \phi$.

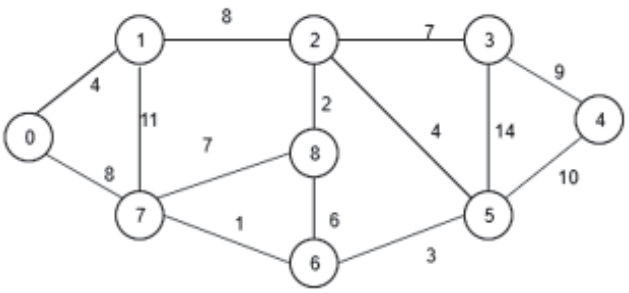
Step 3: Pick the smallest edge of G. Check if the edge forms a cycle with the spanning-tree formed so far in T. If a cycle is not formed with the edges of T, include this edge. Else, discard it.

Step 4: Repeat step 3 until there are (|V|-1) edges in the spanning tree T.

The algorithm is a Greedy Algorithm. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far.

Example 7.2:

Let us considerthe graph in Figure 7.3



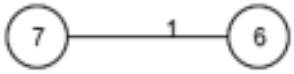
Edges	Weight	Selection
(7,6)	1	Yes
(2,8)	2	Yes
(6,5)	3	Yes
(2,5)	4	Yes
(0,1)	4	Yes
(8,6)	6	No
(7,8)	7	No
(2,3)	7	Yes
(1,2)	8	No
(0,7)	8	Yes
(3,4)	9	Yes
(4,5)	10	No
(1,7)	11	No
(3,5)	14	No

Figure 7.3: A weighted connected graph

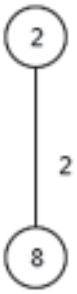
The graph contains 9 vertices and 14 edges. So, the minimum spanning tree will be formed having (9 - 1) = 8 edges. Now the steps of the algorithm are mentioned as follows.

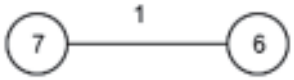
Now pick all edges one by one from sorted list of edges

1. Pick edge 7-6: No cycle is formed, include it.

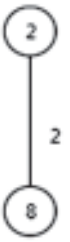


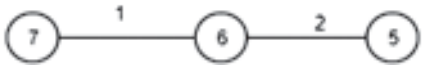
2. Pick edge 8-2: No cycle is formed, include it.



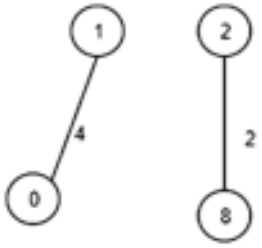


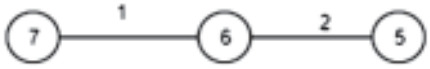
3. Pick edge 6-5: No cycle is formed, include it.



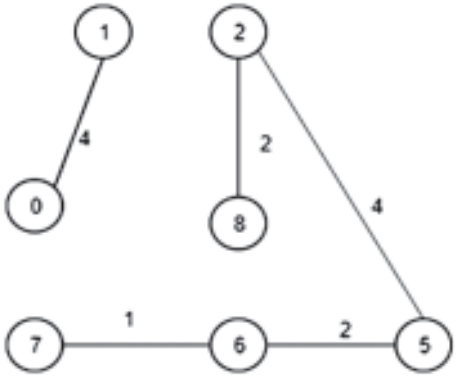


4. Pick edge 0-1: No cycle is formed, include it.

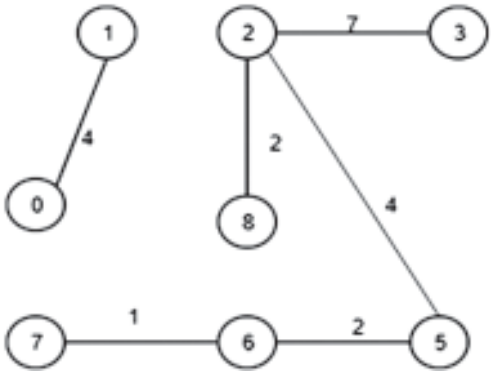




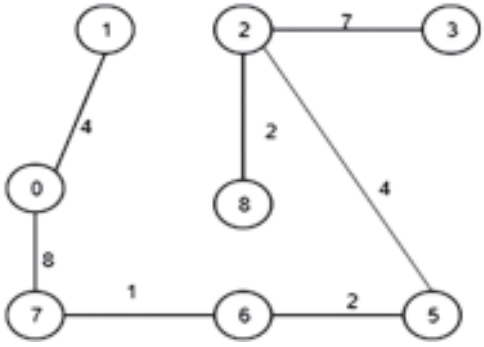
5. Pick edge 2-5: No cycle is formed, include it.



6. Pick edge 8-6: Since including this edge results in cycle, discard it.
7. Pick edge 2-3: No cycle is formed, include it.

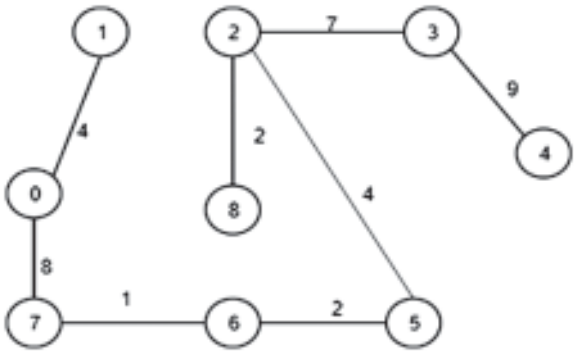


8. Pick edge 7-8: Since including this edge results in cycle, discard it.
9. Pick edge 0-7: No cycle is formed, include it.



10. Pick edge 1-2: Since including this edge results in cycle, discard it.

11. Pick edge 3-4: No cycle is formed, include it.



Since the number of edges included equals (|V| - 1), the algorithm stops here.

Theorem 7.1 (Kruskal): In a connected weighted graph G, Kruskal's Algorithm constructs a minimum-weight spanning tree.

Proof: The first task is to show that the algorithm produces a tree. It is obvious that the graph prepares an acyclic subgraph containing (|V| - 1) number of vertices. As a result, the subgraph is a tree.

Let T be the resulting spanning tree, and T′ be a spanning tree with minimum weight. If T = T′, then there is nothing to prove. If not, let e be an edge in T which is not in T′. If we add e to T′, it will form a cycle. However, then there exists one edge e′ which does not belong to T. Let us consider the spanning-tree T′+e–e′. Thus when the algorithm chooses the last edge between e and e′, it is clear that w(e) ≤ w (e′). Repeating the process, we can conclude the theorem.

7.4.2 Prim's algorithm

Prim's algorithm is used to find a minimum spanning tree. Prim's algorithm is an algorithm that takes a graph as input and finds the subset of the edges of that graph which form a tree that includes every vertex has the minimum sum of weights among all the trees that can be formed from the graph. It falls under a class of algorithms called greedy algorithms which find the local optimum in the hopes of finding a global optimum.We start from one vertex and keep adding edges with the lowest weight until we reach our goal.

The steps of Prim's algorithm are as follows:

Input: A connected weighted graph

Step 1: Create a set T that keeps track of vertices already included in Minimum Spanning Tree (MST).

Step 2: Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.

Step 3: While T doesn't include all vertices of the graph

a) Pick a vertex u which is not there in T and has minimum key value.

b) Include u to T.

c) Update key value of all adjacent vertices of u. To update the key values, iterate through all adjacent vertices. For every adjacent vertex v, if weight of edge (u,v) is less than the previous key value of v, update the key value as weight of (u,v).

The idea of using key values is to pick the minimum weight edge from cut. The key values are used only for vertices which are not yet included in MST, the key value for these vertices indicate the minimum weight edges connecting them to the set of vertices included in MST.

Example 7.3: Let us understand with the following example:

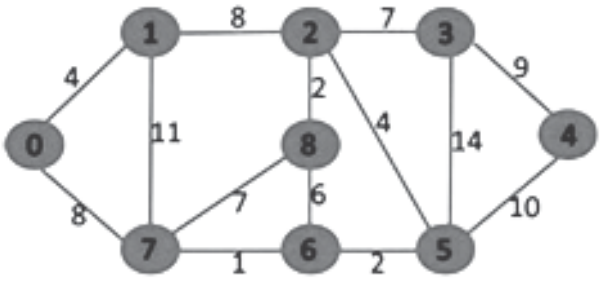
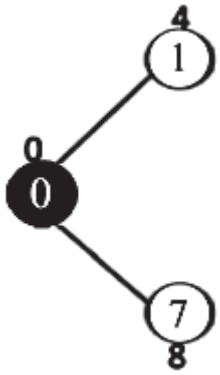


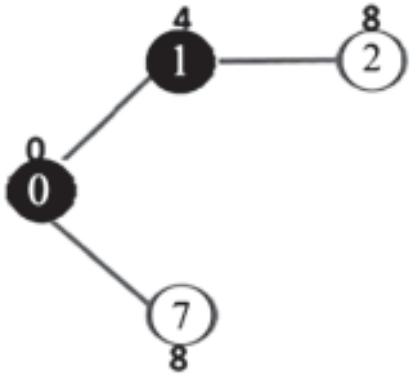
Figure 7.4: A weighted connected graph

The set T is initially empty and keys assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite. Now pick the vertex with the minimum key value. The vertex 0 is picked, include it in T. So T becomes {0}. After including to T, update key values of adjacent vertices. Adjacent vertices of 0 are 1 and 7. The key values of 1 and 7 are updated as 4 and 8. Following subgraph shows

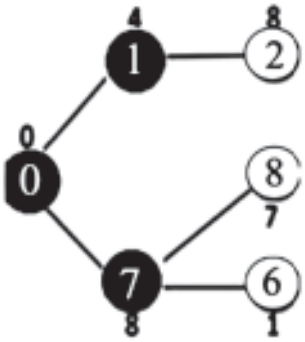
vertices and their key values, only the vertices with finite key values are shown. The vertices included in T are shown in darkcolor.



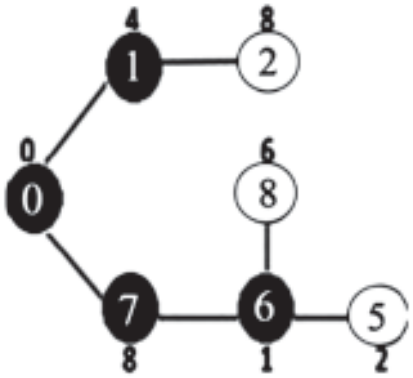
Pick the vertex with minimum key value and not already included in MST (i.e, not in T). The vertex 1 is picked and added to T. So T now becomes {0, 1}. Update the key values of adjacent vertices of 1. The key value of vertex 2 becomes 8.



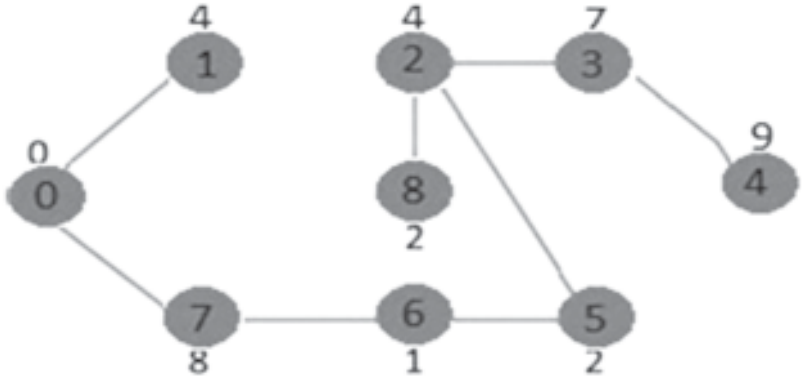
Pick the vertex with minimum key value and not already included in MST (not in T). We can either pick vertex 7 or vertex 2, let vertex 7 is picked. So T now becomes {0, 1, 7}. Update the key values of adjacent vertices of 7. The key value of vertex 6 and 8 becomes finite (1 and 7 respectively).



Pick the vertex with minimum key value and not already included in MST (not in T). Vertex 6 is picked. So T now becomes {0, 1, 7, 6}. Update the key values of adjacent vertices of 6. The key value of vertex 5 and 8 are updated.



We repeat the above steps until T includes all vertices of given graph. Finally, we get the following graph.



7.5 Shortest Path

The shortest path problem is about finding a path between two vertices in a graph such that the total sum of the weights of the edges is minimum.

7.5.1 Dijkstra's shortest path algorithm

Given a graph and a source vertex in the graph, find the shortest paths from source to all vertices in the given graph.

Dijkstra's algorithm is very similar to Like Prim's MST, we generate at (shortest-path tree) with a given source as root. We maintain two sets, one set contains vertices included in the shortest-path tree, other set includes vertices not yet included in the shortest-path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.

Below are the detailed steps used in Dijkstra's algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.

The steps are given as follows.

1) Create a set T (shortest path tree set) that keeps track of vertices included in the shortest-path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.

2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.

3) While T doesn't include all vertices

   a) Pick a vertex u which is not there in T and has minimum distance value.

   b) Include u to T

   c) Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if the sum of a distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

Example 7.4: Let us consider Figure 7.5 as an input graph.

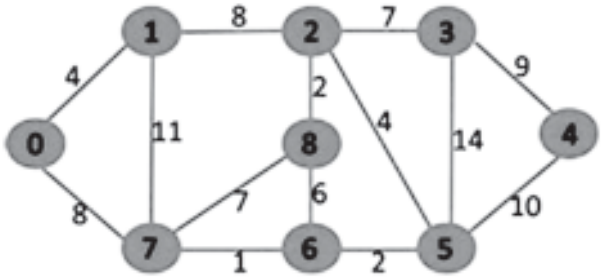
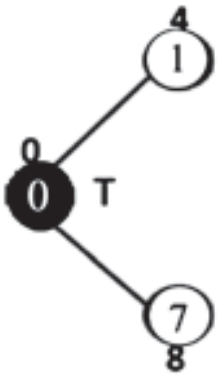
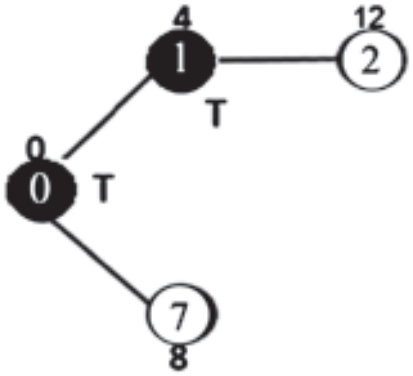


Figure 7.5: A weighted connected graph

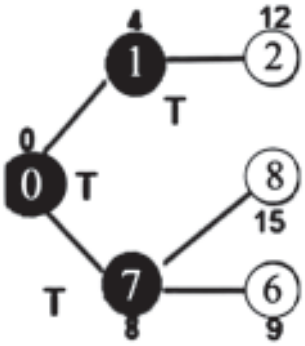
The set T is initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite. Now pick the vertex with minimum distance value. The vertex 0 is picked, include it in T. So T becomes {0}. After including 0 to T, update distance values of its adjacent vertices. Adjacent vertices of 0 are 1 and 7. The distance values of 1 and 7 are updated as 4 and 8. Following subgraph shows vertices and their distance values, only the vertices with finite distance values are shown. The vertices included in T are marked 'T' beside the vertices.



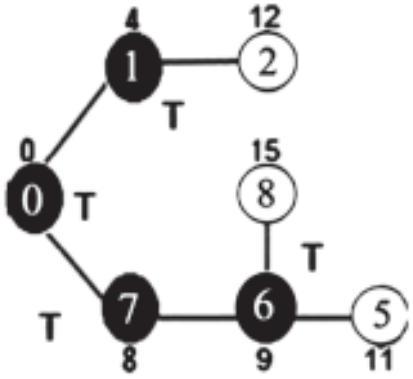
Pick the vertex with minimum distance value and not already included in T. The vertex 1 is picked and added to T. So T now becomes {0, 1}. Update the distance values of adjacent vertices of 1. The distance value of vertex 2 becomes 12.



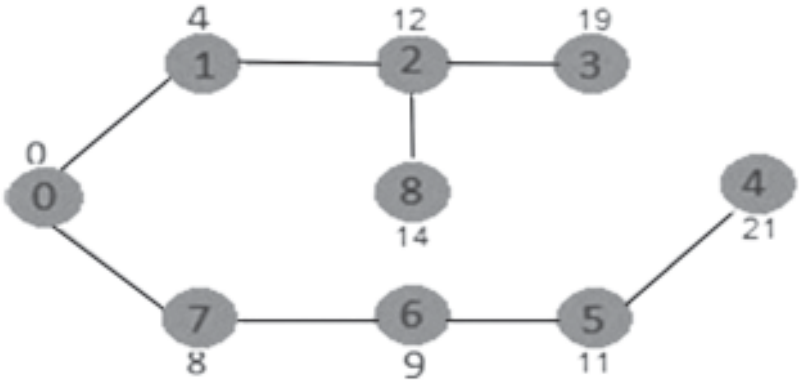
Pick the vertex with minimum distance value and not already included in T. Vertex 7 is picked. So T now becomes {0, 1, 7}. Update the distance values of adjacent vertices of 7. The distance value of vertex 6 and 8 becomes finite (15 and 9 respectively).



Pick the vertex with minimum distance value and not already included in T. Vertex 6 is picked. So T now becomes {0, 1, 7, 6}. Update the distance values of adjacent vertices of 6. The distance value of vertex 5 and 8 are updated.



We repeat the above steps until T does not include all vertices of a given graph. Finally, we get the following Shortest Path Tree (SPT).



7.5.2 Floyd Warshall Algorithm

The Floyd Warshall Algorithm is for solving the All Pairs Shortest Path problem. The problem is to find the shortest distances between every pair of vertices in a given edge-weighted directed Graph.The algorithm is an efficient method of finding the adjacency matrix of the transitive closure of relation on a finite set from the adjacency matrix. It uses properties of the digraph D, in particular, walks of various lengths in D.

We let A be the adjacency matrix of R and T be the adjacency matrix of the transitive closure of R. T is called the reachability matrix of digraph D due to the property that $T_{(i,j)}$ = 1 if and only if v_j can be reached from v_i in D by a sequence of arcs (edges).

1. In Warshall's algorithm, we construct a sequence of Boolean matrices A = $W^{[0]}$, $W^{[1]}$, $W^{[2]}$,...,$W^{[n]}$ = T, where A and T are as above. This can be done from digraph D as follows.
2. $[W^{[1]}]_{(i,j)}$ = 1 if and only if there is a walk from v_i to v_j with elements of a subset of $\{v_1\}$as interior vertices.
3. $[W^{[2]}]_{(i,j)}$ = 1 if and only if there is a walk from v_i to v_j with elements of a subset of $\{v_1, v_2\}$ as interior vertices.
4. Continuing this process, we generalize to $[W^{[k]}]_{(i,j)}$ = 1 if and only if there is a walk from v_i to v_j with elements of a subset of $\{v_1, v_2,\ldots v_k\}$ as interior vertices.

To find $[W^{[k]}]_{i,j}$ we use the formula $[W^{[k]}]_{(i,j)} = [W^{[k-1]}]_{(i,j)}$ or $[W^{[k-1]}]_{(i,k)}$ and $[W^{[k-1]}]_{(k,j)}$

Example 7.5:

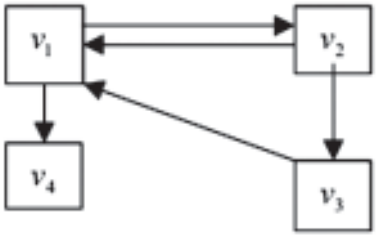


Figure 7.6: A directed graph

Let us consider the graph in Figure 3.3. The adjacency matrix of the graph is drawn as follows.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$ which is taken as $[W^{[0]}]$

Now, $[W^{[1]}] = [W^{[0]}]_{(i,j)}$ or $([W^{[0]}]_{(i,1)}$ and $[W^{[k]}]_{(1,j)})$

Let us construct the $W^{[1]} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ where v_1 is an intermediate vertex.

Now, $[W^{[2]}]_{(i,j)} = [W^{[1]}]_{(i,j)}$ or $([W^{[1]}]_{(i,2)}$ and $[W^{[1]}]_{(2,j)})$

Let us construct the $W^{[2]} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ where v_1is the intermediate vertex.

Now, $[W^{[3]}]_{(i,j)} = [W^{[2]}]_{(i,j)}$ or $([W^{[2]}]_{(i,3)}$ and $[W^{[2]}]_{(3,j)})$

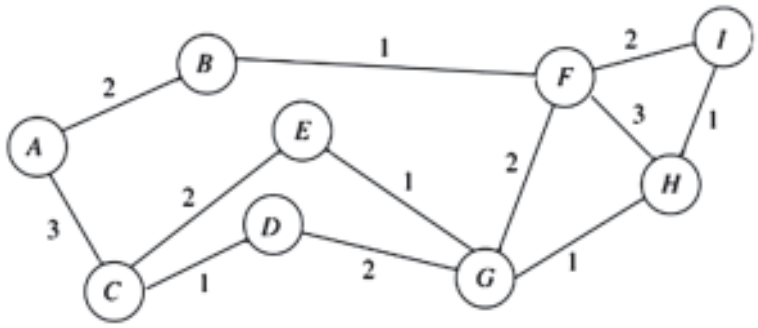
Let us construct the $W^{[3]} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ where v_1 is the intermediate vertex.

Now, $[W^{[4]}]_{(i,j)} = [W^{[3]}]_{(i,j)}$ or $([W^{[3]}]_{(i,4)}$ and $[W^{[3]}]_{(4,j)})$

Let us construct the $W^{[4]} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ where v_1 is the intermediate vertex.

7.6 Illustrated Problems

Use Dijkstra Algorithm to find shortest paths of the following graph.



Solution:

Dijkstra Algorithm: To find the shortest path lengths from each node to the starting node.

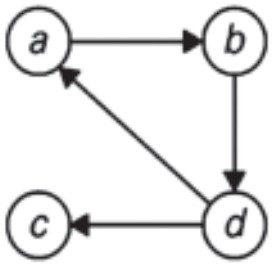
Initially, the shortest path set T = ϕ and gradually in different steps the set is completed with all vertices as follows. In the table, the updated values of vertices are shown.

Steps	Shortest Paths
Step 1	T={A}
Step 2	T={A,B}
Step 3	T={A,B,F}
Step 4	T={A,B,F,C}
Step 5	T={A,B,F,C,D}
Step 5	T={A,B,F,C,D,E}
Step 6	T={A,B,F,C,D,E,G}
Step 7	T={A,B,F,C,D,E,G,I}
Step 8	T={A,B,F,C,D,E,G,I,H}

Vertices	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8
A	×	×	×	×	×	×	×	×
B	2	×	×	×	×	×	×	×
C	3	3	3	×	×	×	×	×
D	?	?	?	4	×	×	×	×
E	?	?	?	5	5	×	×	×
F	?	3	×	×	×	×	×	×
G	?	?	5	5	5	5	×	×
H	?	?	6	6	6	6	6	6
I	?	?	5	5	5	5	5	×

Thus, we obtained the shortest path lengths from each node to the starting nodes as from B to A: 2, C to A: 3, D to A: 4, E to A: 5, F to A: 3, G to A: 5, H to A: 6 and I to A: 5.

2. Use the Warshall Algorithm to find all pair of shortest paths in the given problem.



From the given graph, we have the adjacent Matrix T_0, given as follows.

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
C	0	0	0	0
d	1	0	1	0

Now, let us consider T_1, a matrix representation of paths from each pair passing the vertex "a" as follows.

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
C	0	0	0	0
d	1	1	1	0

Now, similarly, matrix T_2, where "a and b" as the intermediate vertex is formed as follows

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
C	0	0	0	0
d	1	1	1	1

Now,

$T_3 =$

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
C	0	0	0	0
d	1	1	1	1

and

$T_4 =$

	a	b	c	d
a	1	1	1	1
b	1	1	1	1
C	0	0	0	0
d	1	1	1	1

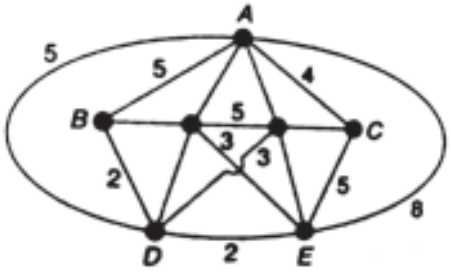
Thus, the matrix T_4 is the representation of all such paths intermediating a, b, c and d. It is the transitive closure of the adjacency matrix.

7.7 Summary

An important type of graph, Tree is introduced in this unit. Then some very useful and simple algorithms for finding minimal spanning tree are described which are applied in various fields and situations.

7.8 Exercise

1. Find the tour using the travelling salesman problem of the given graph.



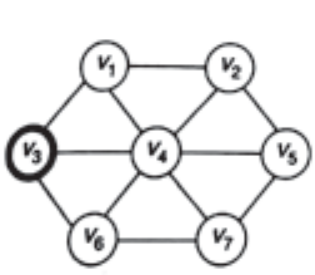
2. For each of the adjacency matrices A given below, (a) draw the corresponding digraph and (b) find the matrix T of the transitive closure using the digraph implementation of Warshall's algorithm.

$$\text{(i)}\ A=\begin{bmatrix}0&1&0\\1&1&0\\0&0&1\end{bmatrix} \qquad \text{(ii)}\ A=\begin{bmatrix}1&1&1\\0&1&1\\0&0&1\end{bmatrix}$$

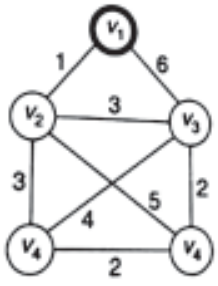
$$\text{(iii)}\quad A=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{bmatrix}$$

3. Show that the following graph having 13 vertices with 9 vertices of degree 1, 3 vertices of degree 4 and one vertex of degree 3 is a tree.

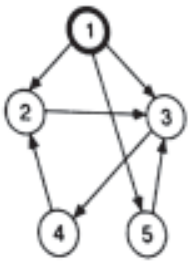
4. Prove or disprove: If T is minimum-weight spanning tree of a weighted graph G, then any path in T is minimum weight path in G.
5. Prove that Prime's algorithm produces a minimum weight spanning tree of G.
6. Apply Dijkstra Algorithm to find the shortest path tree for the following graphs. Thick circles indicate source vertices.



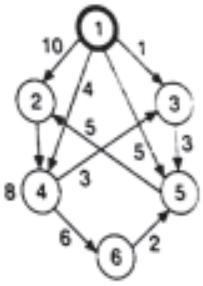
G1



G2



G3



G4

7. Prove that

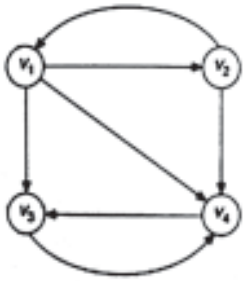
   a. A complete graph with n vertices has atleast $2^{(n-1)}$-1 spanning trees.

   b. A graph of n vertices can have at most $n^{(n-2)}$ number of spanning trees.

8. Give an example such that the Dijkstra Algorithm will fail.

9. For a complete graph with n vertices show that the maximum number of different paths between a pair of vertices is (n-1)!.

10. Find the path matrix for the graph using the Warshall algorithm



References:

1. Bondy, J. A. and Murty, U.S.R., 'Graph Theory with Applications', Springer, 2008.
2. Diestel, R. Graph Theory (Graduate Texts in Mathematics). New York, NY: Springer-Verlag, 1997. ISBN: 3540261834
3. N. Alon and J. Spenser, "Probabilistic Methods", John Wiley and Sons, 2nd edition, 2000. Bollobás, B. Modern Graph Theory (Graduate Texts in Mathematics). New York, NY: Springer-Verlag, 1998. ISBN: 0387984917.
4. M. A. Iqbal, Graph Theory & Algorithms, Electronic edition 2010 Chartrand & Oellermann, Applied and Algorithmic Graph Theory, 1993, McGraw Hill.
5. Harikishan, Shivraj Pundir and Sandeep Kumar, Discrete Mathematics, Pragati Publication, 7th Edition, 2010.
6. Colmun, Busby and Ross, Discrete Mathematical Structure, PHI Publication, 6th Edition, 2009
7. C.L. Liu, Elements of Discrete Mathematics, Tata McGraw Hill, 2nd Edition, 2000.
8. N. Deo, Graph Theory with Applications to Engineering and Computer Science, PHI publication, 3rd edition, 2009.
9. B.A.Davey and H.A. Priestley, Introduction of Lattices and Order, Cambridge University Press, Cambridge, 1990.
10. Edgar G.Goodaire and Michael M. Parmenter, Discrete Mathematics with Graph Theory, 2nd Edition, Pearson Education (Singapore) P. Ltd., Indian Reprint 2003.