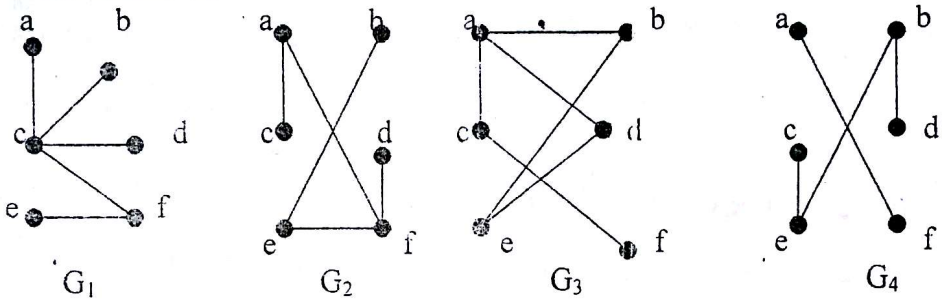
INTRODUCTION TO TREE

Definition: A tree is a connected undirected graph with no simple circuits.

- A tree must be a simple graph, it cannot contain multiple edges or loops.
- Trees were used as long ago as 1857, when the English mathematician Arthur Cayley used them to count certain types chemical compounds.
- An undirected graph with no simple circuit is called *forest*.

Example: Which of the graphs shown in Figure are trees?
Solution: G_1 and G_2 are trees, since both are connected graphs with no simple circuits. G_3 is not a tree because e, b, a, d, e is a simple circuit in this graph. Finally, G_4 is not a tree since it is not connected.



Theorem: An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

Proof: First assume that T is a tree. Then T is a connected graph with no simple circuits. Let x and y be two vertices of T. Since T is connected graph, so there is a simple path between x and y. Moreover, this path must be unique, for if there were a second such path, the path formed by combining the first path from x to y followed by the path from y to x obtain by reversing the order of the second path from x to y would from a circuit. This implies that there is a simple circuit in T. Hence, there is a unique simple path between two vertices of a tree.

Now assuming that there is a unique simple path between any two vertices of a graph T. Then T is connected, since there is a path between any two of its vertices. Furthermore, T can have no simple circuits. To see that this is true, suppose T had a simple circuit that contained the vertices x and y. Since the simple circuit is made up of a simple path from x to y and a second simple path from y to x. Hence, a graph with a unique simple path between any two vertices is a tree.

Tree Terminology:

Root: In many applications of trees a particular vertex of a tree is designed as the root.
Rooted tree: A rooted tree is a tree in which one vertex has been designated as the root and every edge is directed away from the root.
Parent: If v is a vertex in T other than the root, the parent of v is the unique vertex u such that there is a directed edge from u to v.
Child: When u is the parent of v, v is called a child of u.
Siblings: Vertices with same parent are called siblings.
Ancestors: the ancestors of a vertex other than the root are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root.

Scanned by CamScanner

Descendents: the descendents of a vertex v are those vertices that have v as an ancestor.
Leaf: A vertex of a tree is called leaf if it has no children.
Internal vertices: Vertices that have children are called internal vertex.
Subtree: If *a* is a vertex in a tree, the sub tree with *a* as its root is the subgraph of the tree consisting of *a* and its descendants and all edged incident to these descendents.

m-ary tree: A rooted tree is called an *m-ary* tree if every internal vertex has no more than *m* children.
Full m-ary tree: The tree is called a full *m-ary* tree if every internal vertex has exactly *m* children. An *m-ary tree* with m = 2 is called binary tree.
***Ordered rooted tree*:** An ordered rooted tree is a rooted tree where the children of each internal vertex are ordered.
***Left child*:** In an ordered binary tree, if an internal vertex has two children, the first child is called the left child.
***Right child*:** In an ordered binary tree, if an internal vertex has two children, the second child is called the right child.
***Left subtree*:** The tree rooted at the left child of a vertex is called the left subtree of that vertex.
***Right subtree*:** The tree rooted at the right child of a vertex is called the right subtree of that vertex.

PROPERTIES OF TREE

Theorem: A tree with **n** vertices has **n-1** edges.
Proof: We will use mathematical induction to prove this theorem.
Basis Step: When n =1, the tree has no edges. It follows that the theorem is true for n=1.
Inductive Step: The induction hypothesis states that every tree with k vertices has k-1 edges, where k is a positive integer. Suppose that a tree T has k+1 vertices and that v is a leaf of T (which must exist since the tree is finite), and let w be the parent of v. Removing from T the vertex v and the edge connecting w to v produces a tree T' with k vertices, since the resulting graph is still connected and has no simple circuits. By the induction hypothesis, T' has k-1 edges. It follows that T has k edges since it has one more edge than T', the edge connecting v and w. This completes the induction step.

Theorem: A full *m-ary* tree with ***i*** internal vertices contains **n = mi + 1** vertices.
Proof: Every vertex, except the root, is the child of an internal vertex. Since each of the ***i*** internal vertices has ***m*** children, there are mi vertices in the tree other than the root. Therefore, the tree contains **n = mi + 1** vertices.

Theorem: A full *m-ary* tree with

(i) ***n*** vertices has ***i = (n -1) / m*** internal vertices and ***l = [(m -1)n+1] / m*** leaves.
(ii) ***i*** internal vertices has ***n=mi+1*** vertices and ***l=(m -1)i +1*** leaves.
(iii) ***l*** leaves has ***n=(ml -1)(m -1)*** vertices and ***i =(l -1) / (m -1)*** internal vertices.

Level: The level of a vertex v in a rooted tree is the length of unique path from the root to this vertex. The level of the root is defines to be zero.
Height: The height of a rooted tree is the maximum of the levels of vertices. In other words, the height of a rooted tree is the length of the longest path from the root to any vertex.

Scanned by CamScanner

- A rooted m-ary tree of height h is **balanced** if all leaves are at levels h or h-1.
- There are at most m^n leaves in an m-ary tree of height h.
- If an m-ary tree of height h has l leaves, then $h \geq \lceil \log_m l \rceil$. If the m ary tree is full and balanced, then $h = \lceil \log_m l \rceil$.

APPLICATION OF TREES

Searching for items in a list is one of the most important tasks that arises in computer science. Our primary goal is to implement a searching algorithm that finds items efficiently when the items are totally ordered. This can be accomplished through the use of a binary search tree.

Binary Search Tree:
Binary search tree is a binary tree in which the vertices are labeled with items so that a label of a vertex is greater than the labels of all vertices in the left subtree of this vertex and is less than the labels of all vertices in the right subtree of this vertex.

Decision Tree:
A rooted tree where each internal vertex represents a possible outcome of a decision and the leaves represent the possible solutions.

Example: Suppose there are seven coins, all with the same weight, and a counterfeit coin that weighs less than the others. How many weighing on a balance scale to determine which of the eight coins is the counterfeit one? Give an algorithm for finding this counterfeit coins.
Solution: There are three possibilities for each weighing on a balance scale. The two pans can have equal weight, the first pan can be heavier, or the second pan can be heavier. Consequently, the decision tree for the sequence of weighings is a 3-ary tree. There are at least eight leaves in the decision tree since there are eight possible outcomes (since each of the eight coins can be the counterfeit lighter coin), and each possible outcomes must be represented by at least one leaf. The largest number of weighings needed to determine the counterfeit coin is the height of the decision tree. We know from previous discussion, it follows that the height of the decision tree is at least $\lceil \log_3 8 \rceil = 2$. Hence, at least two weighings are needed.

Tree Traversal

Traversal algorithms:
Procedures for systematically visiting every vertex of an ordered rooted tree are called **traversal algorithm**. We will describe three of the most commonly used such algorithms, **preorder traversal**, **inorder traversal**, and **postorder traversal**. Each of these algorithms can be defined recursively. We first define preorder traversal.

Preorder traversal: Let T be an ordered rooted tree with root r. if T consists only of r, then r is the preorder traversal of T. Other wise, suppose that $T_1, T_2, \ldots, T_n$ are the subtrees at r from left to right in T. The preorder traversal begins by visiting r.. it continues by traversing T_1 in preorder, then T_2 in preorderd, and so on, until T_n is traversal in preorder.

Scanned by CamScanner

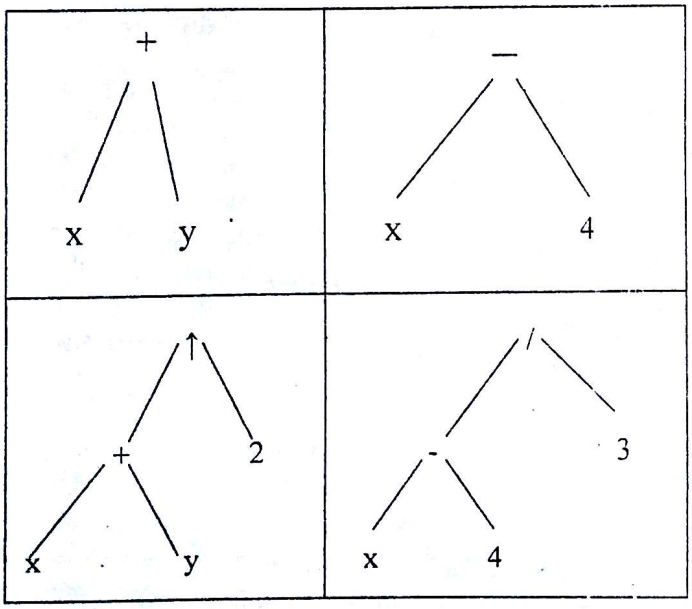
Inorder traversal: Let T be an ordered rooted tree with root r. if T consists only of r, then r is the inorder traversal of T. Other wise, suppose that $T_1, T_2, \ldots, T_n$ are the subtrees at r from left to right. The inorder traversal begins by traversing T_1 in inorder then visiting r. It continues by traversing T_2 in inorder, then T_3 in inorder, ... and finally T_n.

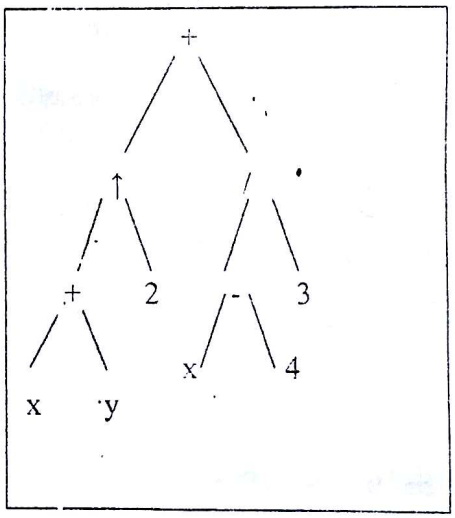
Postorder traversal: Let T be an ordered rooted tree with root r. if T consists only of r, then r is the postorder traversal of T. Other wise, suppose that $T_1, T_2, \ldots, T_n$ are the subtrees at r from left to right. The postorder traversal begins by traversing T_1 in postorder, thenT_2 in postorder, ... then T_n in postorderd and ends by visiting r.

Example 2,3,4: See Rosen [Page – 662-5, 5th Edition].

Example: What is the ordered rooted tree that represents the expressions ((x + y) ↑ 2) + ((x –4) / 3)?

Solution: The binary tree for this expression can be built from the bottom up. First, a subtree for the expression x + y is constructed. Then this is incorporated as part of the larger subtree representing (x + y) ↑ 2. Also, a subtree for x - 4 is constructed, and then this is incorporated into a subtree representing (x – 4) / 3. Finally the subtrees representing (x + y) ↑ 2 and (x – 4) / 3 are combined to form the ordered rooted tree representing ((x + y) ↑ 2 + ((x - 4) / 3). These steps are shown in the following figure.





Example: What is the prefix from for ((x + y) ↑ 2) + ((x – 4) / 3)?

Solution: We obtain the prefix from for this expression by preorder traversing the binary tree that represents it, shown in figure 1. This produces + ↑ + x y 2 / - x 4 3.

Example: What is the postfix from for ((x + y) ↑ 2) + ((x – 4) / 3)?

Solution: We obtain the postfix form for this expression by postorder traversing the binary tree that represents it, shown in figure 1. This produces x y + 2 ↑ x 4 – 3 / +.

Scanned by CamScanner

SPANNING TREES

Definition: Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G.

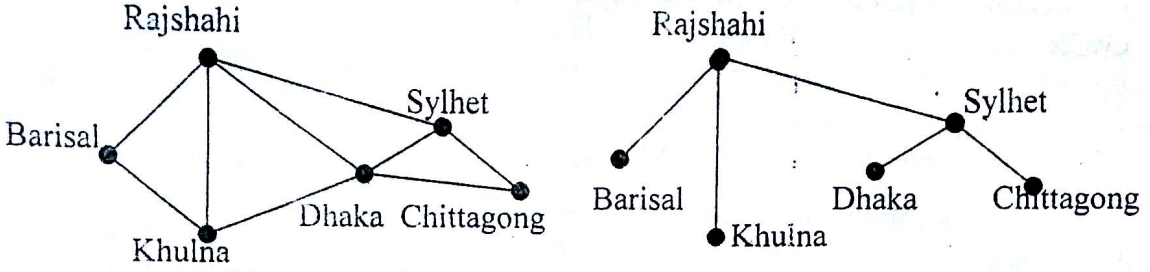


Figure: Road system and spanning system

Example: Find the spanning tree of the simple graph G shown in figure 2.
Solution: The graph G is a connected, but it is not a tree because it contains simple circuits. Remove the edge $\{a, e\}$. This eliminates one simple circuit, and the resulting subgraph is still connected every vertex of G. Next remove the edge $\{e, f\}$ to eliminate a second simple circuit. Finally, remove edge$\{c, g\}$ to produce a simple graph with no circuits. This subgraph is a spanning tree; since it is a tree that contains every vertex of G.

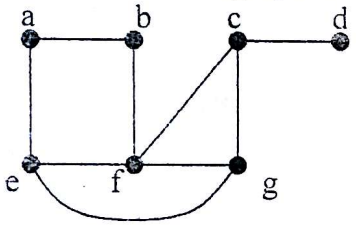


Fig: The simple graph G.

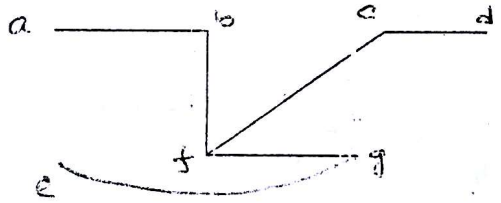


Fig: Spanning tree for G

Theorem: A simple graph is connected if and only if it has a spanning tree.
Proof: First, suppose that a simple graph G has a spanning tree T. T contains every vertex of G. Furthermore; there is a path in T between any two of its vertices. Since T is a subgraph of G, there is a path in G between any two of its vertices. Hence, G is connected.
Now suppose that G is connected. If G is not a tree, it must contain a simple circuit. Remove an edge from one of these simple circuits. The resulting subgraph has one fewer edge but still contains all the vertices of G and is connected. If the is subgraph is not a tree, it has a simple circuit; so as before, remove an edge that is in a simple circuit. Repeat this process until no simple circuits remain. This is possible because there are only a finite number of edges in the graph. The process terminates when no simple circuit remain. A tree is produced since the graph stays connected, as edges are removed. This tree is a spanning tree since it contains every vertex of G.

Depth First Search
Example 3: See Rosen [Page – 678, 5th Edition].

Breadth First Search
Example 5: See Rosen [Page – 680, 5th Edition].

Backtracking
Example 7: See Rosen [Page – 683, 5th Edition].

Scanned by CamScanner

MINIMUM SPANNING TREES

Definition: A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

- The weight of a minimal spanning tree is unique, but the minimum spanning tree itself is not.
- There are many algorithms to construct minimum spanning tree.

Algorithm: The input is a connected weighted graph G with n vertices.

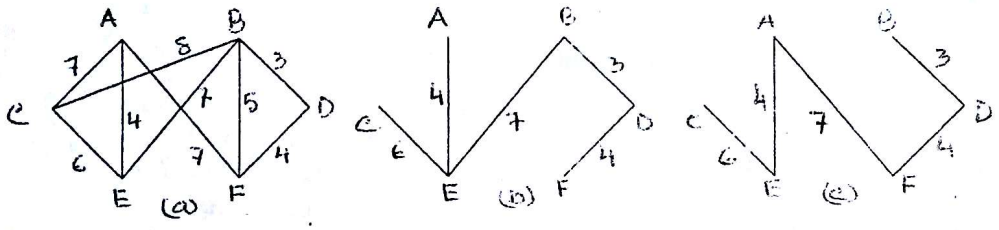
1. Arrange the edges of G in the order of decreasing weights.
2. Proceeding sequentially, delete each edge that does not disconnect the graph until n-1 edges remain.
3. Exit.

Kruskal's Algorithm: [Joseph Kruskal, 1956]

The input is a connected weighted graph G with n vertices.

1. Arrange the edges of G in the order of increasing weights.
2. Starting only with the vertices of G and proceeding sequentially, add each edge which does not result in a cycle until n-1 edges are added.
3. Exit.

Example: Find the minimum spanning tree of the following weighted graph G using the above two algorithms. [Page – 200, SOS]



Solution:

a. First we order the edges by decreasing weights and then we successively delete edges without disconnecting G until five edges remain. This yields the following data:

Edges	BC	AF	AC	BE	CE	BF	AE	DF	BD
Weight	8	7	7	7	6	5	4	4	3
Delete?	Y	Y	Y	N	N	Y			

The minimum spanning tree has weight 24 and it is shown in Fig-(b).

b. First we order the edges by increasing weights and then we successively add edges without forming any cycles until five edges are included. This yields the following data:

Edges	BD	AE	DF	BF	CE	AC	AF	BE	BC
Weight	3	4	4	5	6	7	7	7	8
Add?	Y	Y	Y	N	Y	N	Y		

This minimum spanning tree also has weight 24 and it is shown in Fig-(c).

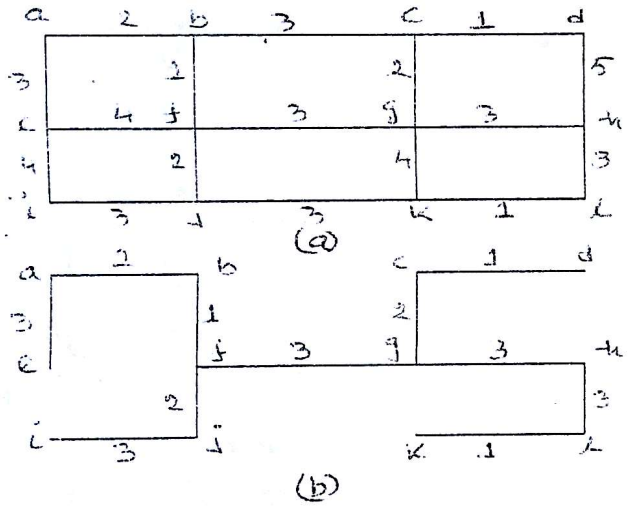
- Observe that the two minimum spanning tree is not same.

Scanned by CamScanner

Prim's Algorithm: [Robet Prim 1957]

Algorithm: *Prim's Algorithm*
Procedure *Prim* (G: weighted connected undirected graph with n Vertices)
T := a minimum-weight edge
for i := 1 to n -2
begin
e := edges of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T
T := T with e added
End {T is a minimum spanning tree of G}

Example: Use Prim's algorithm to find a minimum spanning tree in the graph shown in the following figure. [Page-691, Rosen, 5th Edition]
Solution: A minimum spanning tree constructed using Prim's algorithm is shown below. The successive edges chosen are displayed.



Choice	Edge	Weight
1	{b,f}	1
2	{a,b}	2
3	{f,j}	2
4	{a,e}	3
5	{i,j}	3
6	{f,g}	3
7	{c,g}	2
8	{c,d}	1
9	{g,h}	3
10	{h,l}	3
11	{k,l}	1
	Total	24

Difference between Prim's & Kruskal's Algorithm
There are some differences between Prim's and Kruskal's algorithms.

- In Prim's algorithm edges of minimum weight that are incident to a vertex already in the tree, and not forming a circuit, are chosen. On the other hand, in Kruskal's algorithm edges of minimum weight that are not necessarily incident to a vertex already in the tree, and that do not form a circuit, are chosen.
- To find a minimum spanning tree of a graph with e edges and v vertices, Kruskal's algorithm can be carried out using O (e log e) operations and Prim's algorithm can be carried out using O (e log v) operations. [Consequently, it is preferable to use Kruskal's algorithm for graphs that are sparse, that is, where e is very small. Otherwise there is a little difference in the complexity of these two algorithms.]

Mir@cle Point
আলেয়া ফটোগ্রাফির নীচে
কেয়াবীর সামনে, কলেজ রোড, চকবাজার, চট্টগ্রাম।
IIUC এর সকল লেকচার শীট ও পূর্ববর্তী সেমিস্টারের প্রশ্ন পাওয়া যায়।
মোবাইল : 01775943269

Scanned by CamScanner

Introduction to Graphs

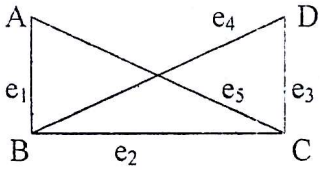
Graph theory is an old subject with many modern applications. Its basic ideas were introduced in the 18th century by the great Swiss mathematician **Leonhard Euler**. He used graphs to solve the famous Konigsberg Bridge problem.

Graphs are used to solve problems in many fields. For instance,

- To determine whether a circuit can be implemented on a planner circuit board.
- Distinguish between two chemical compounds with the same molecular formula but different structures using graph.
- To determine whether two computers are connected by a communications link using graph models of computer networks
- Finding shortest path between two cities in a transportation network.
- To make the schedule of exams and to assign channels to television stations.

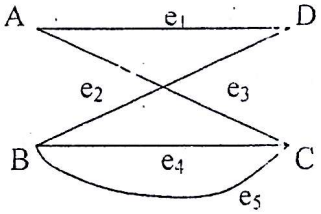
Simple Graph

A simple graph G = (V, E) consist of V, a nonempty set of *vertices*, and E, a set of unordered pairs of distinct elements of V called *edges*.



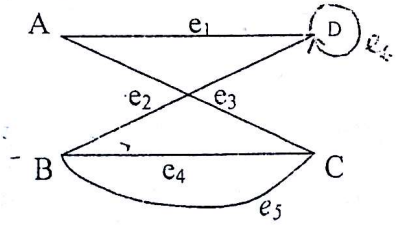
Multigraph

A multigraph G = (V, E) consist of a set V of vertices, a set E of edges, and a function *f* from E to $\{\{u, v\} \mid u,v \in V, u \neq v\}$. The edges e_1 and e_2 are called *multiple* or *parallel edges* if $f(e_1) = f(e_2)$.



Pseudograph

A pseudograph G = (V, E) consist of a set V of vertices, a set E of edges, and a function *f* from E to $\{\{u, v\} \mid u, v \in V\}$. An edge is a *loop* if $f(e) = \{u, u\} = \{u\}$ for some $u \in V$.



Summary

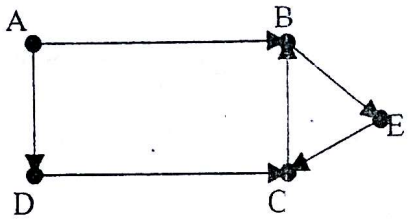
Pseudographs are the most general type of undirected graphs since they may contain loops and multiple edges. Multigraphs are undirected graphs since they may contain multiple edges but may not have loops. Finally, simple graphs are undirected graphs with no multiple edges or loops.

Scanned by CamScanner

Directed Graph

A directed graph (V,E) consists of a set of vertices V and a set of edges E that are *ordered pairs* of elements of V.

- The edges of directed graph are ordered pairs. Loops are allowed but multiple edges in the same direction between two vertices are not allowed.



Directed Multigraph

A directed multigraph G = (V, E) consists of a set V of vertices, a set E of edges, and a function *f* from E to {(u, v) | u, v ∈ V }. The edges e_1 and e_2 are called multiple edges if f (e_1) = f (e_2).

Type	Edges	Multiple Edges Allowed?	Loops Allowed?
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple Directed Graph	Directed	No	[redacted] No
Directed Multigraph	Directed	Yes	Yes

Table-1 page-530 [7th Edition]

Graph Terminology

First, we give some terminology that describes the vertices and edges of undirected graphs.

- Two vertices *u* and *v* in an undirected graph G are called *adjacent* or *neighbors* in G if {u, v} is an edge of G.
- If e = {u, v}, the edges *e* is called *incident with* the vertices u and v.
- The edge *e* is also said to *connect u* and *v*.
- The vertices *u* and *v* are called *endpoints* of the edge {u, v}.

Degree of a vertex

The *degree* of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributed twice to the degree of that vertex. The degree of the vertex *v* is denoted by *deg(v)*.

- A vertex of degree zero (0) is called *isolated*. It follows that an isolated vertex is not adjacent to any vertex.
- A vertex is *pendant* if and only if it has degree one (1). Consequently, a pendant vertex is adjacent to exactly one other vertex.

Example-1 : See Rosen [page-546, 5th Edition].

Scanned by CamScanner

Handshaking Theorem

The sum of the degree of all vertices of a graph G is equal to twice the number of edges in G. Let G = (u, v) be an undirected graph with e edges. Then

$$2e = \sum_{v \in V} \deg(v)$$

- This applies even if multiple edges and loops are present.

Problem: How many edges are there in a graph with ten vertices each of degree six?
Solution: Since the sum of the degrees of the vertices is 6.10 = 60, it follows that 2e = 60. Therefore, e = 30.

Theorem: An undirected graph has even number of vertices of odd degree.
Proof : Let V_1 and V_2 be the set of vertices of even degree and the set of vertices of odd degree, respectively in an undirected graph G = (V, E). Then

$$2e = \sum_{v \in V} \deg(v) = \sum_{v \in V_1} \deg(v) + \sum_{v \in V_2} \deg(v)$$

Since deg(v) is even for $v \in V_1$, the first term in the right hand side of the last equality is even. Furthermore, the sum of the two terms on the right-hand side of the last equality is even, since the sum is 2e. Hence the second term is also is even. Since all the terms in this sum are odd, there must be an even number of such terms. Thus there are an even number of vertices of odd degree.

- Now we will discuss some useful terminology for graphs with directed edges.
- When (u, v) is an edge of the graph G with directed edges, *u* is said to be *adjacent to v* and *v* is said to be *adjacent from u*.
- The vertex *u* is called the *initial vertex* of (u, v), and *v* is called the *terminal or end vertex* of (u, v).
- The initial vertex and terminal vertex of a loop are the same.

- Some definition concerning the degree of vertices.

- In a graph with directed edges the *in-degree* of a vertex *v*. denoted by $deg^-(v)$, is the number of edges with *v* as their terminal vertex.
- The *out-degree* of *v* denoted by $deg^+(v)$, is the number of edges with *v* as their initial vertex.
- A *loop* at a vertex contributes 1 to both the in-degree and the out-degree of this vertex.

Theorem: The sum of the in-degree and the sum of the out-degree of all the vertices in a directed graph are the same. And this is equal to the number of edges of the graph.

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$$

Example-3: See Rosen [Page-547, 5th Edition].

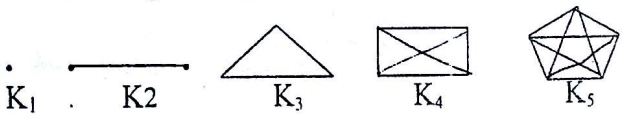
Scanned by CamScanner

Some Special Simple Graph

We will now introduce several classes of simple graphs. These graphs are often used as examples and arise in many applications.

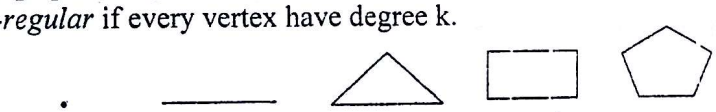
Complete Graph

The complete graph on n vertices, denoted by K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices.



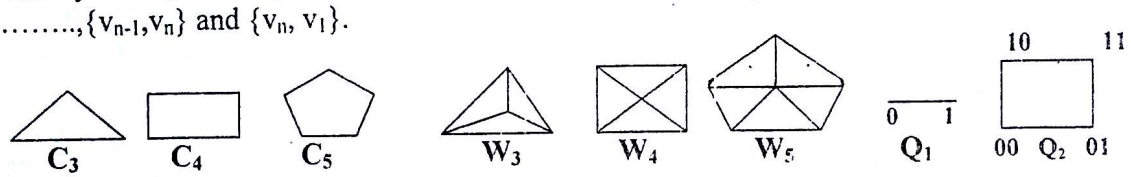
Regular Graph

A graph is regular if every vertex have the same degree. A graph G is regular of degree k or *k-regular* if every vertex have degree k.



Cycles

The cycle $\mathbf{C_n}$, $n \geq 3$, consists of **n** vertices $v_1, v_2, v_3, \ldots, v_n$ and edges $\{v_1,v_2\}$, $\{v_2,v_3\}$, $\ldots\ldots,\{v_{n-1},v_n\}$ and $\{v_n, v_1\}$.



Wheels

We obtain the wheel $\mathbf{W_n}$ when we add an additional vertex to the cycle C_n, for $n \geq 3$, and connect this new vertex to each of the n vertices in C_n, by new edges.

Cubes

The n-dimensional cube, or **n-cube**, denoted by $\mathbf{Q_n}$, is the graph that has the vertices representing the 2^n bit strings of length n. Two vertices are adjacent if and only if the bit strings that they represent differ in exactly one bit position.

Bipartite Graphs

A graph G is said to be bipartite if its vertices V can be partitioned into two disjoint sets M and N such that every edge of G connects a vertex of M to a vertex of N.

- By a complete bipartite graph, we mean that each vertex of M is connected to each vertex of N. This graph is denoted by $K_{m,n}$, where m = Number of vertices in M and n = Number of vertices in N.
- The graph $K_{m,n}$ has *m.n* edges.

Scanned by CamScanner

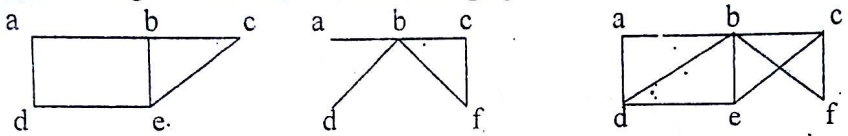
Subgraph

A subgraph of a graph G = (V, E) is a graph H = (W, F) where $W \subseteq V$ and $F \subseteq E$.

Graph G Subgraph H

Union of two Graph

Union of two simple graph $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ is the simple graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$. The union of graph is denoted by $G_1 \cup G_2$.



Representing Graphs

There are many useful ways to represent graphs. One way to represent a graph without multiple edges is to list all the edges of this graph.

Another way to represent a graph with no multiple edges is to use *Adjacency lists*, which specify the vertices that are adjacent to each vertex of the graph.

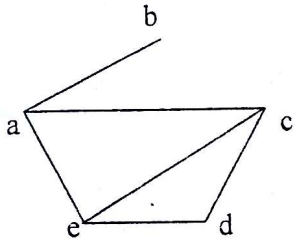


Fig-1: Simple graph

Table 1: An Edge List for a Simple Graph

Vertex	Adjacent Vertices
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d

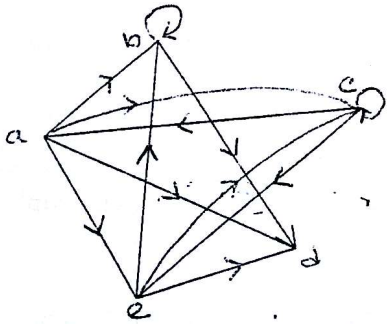


Fig-2: Directed graph

Table 2: An Edge List for a Directed Graph

Initial Vertex	Adjacent Vertices
a	b, c, d, e
b	b, d
c	a, c, e
d	-
e	b, c, d

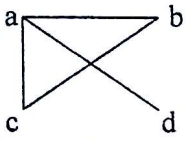
Scanned by CamScanner

Adjacency Matrix

Suppose G = (V,E) is a simple graph where |V| = n. Suppose that the vertices of G are listed arbitrarily as $v_1, v_2, v_3, \ldots, v_n$. The *adjacency matrix* A (or A_G) 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 they are not adjacent. In other words, if its adjacency matrix is $A = [a_{ij}]$, then

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

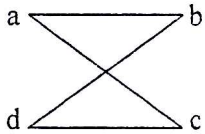
- Example: Find the Adjacency matrix of the following graph.



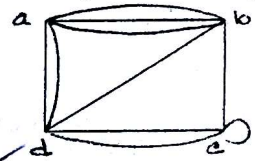
$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

- Example: Draw a graph with the following adjacency matrix.

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



- 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.
- Adjacency matrix can also be used to represent undirected graphs with loops and with multiple edges.
- A loop at the vertex a_i is represented by 1 at the (i, i) th position of the adjacency matrix.
- When multiple edges are present, the adjacency matrix is no longer a zero-one matrix, since the (i, j) th entry of this matrix equals the number of edges that are associated to $\{a_i, a_j\}$.
- All undirected graphs, including multigraphs and pseudographs, have symmetric adjacency matrices.
- Example: Use an adjacency matrix to represent the following pseudograph.



$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

Adjacency matrix for Directed Graphs:

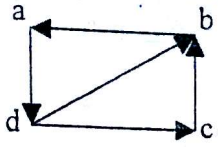
The matrix for a directed graph G = (V,E) has a 1 in its (i, j) th position if there is an edge from v_i to v_j, where $v_1, v_2, v_3, \ldots, v_n$ is an arbitrary listing of the vertices of the directed graph. In other words, if its adjacency matrix is $A = [a_{ij}]$, then

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

- The adjacency matrix for a directed graph does not have to be symmetric, since there may not be an edge from a_j to a_i when there is an edge from a_i to a_j.
- In the adjacency matrix for a directed multigraph, a_{ij} equals the number of edges that are associated to (v_i, v_j).

Scanned by CamScanner

- Example: Find the Adjacency matrix for the following directed graph.



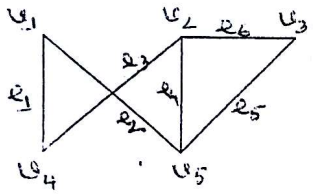
$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Incident Matrix

Let G = (V, E) be an undirected graph. Suppose that $v_1, v_2, \ldots, v_n$ are the vertices and $e_1, e_2, \ldots, e_m$ are the edges of G. Then the incident matrix with respect to this ordering of V and E is the $n \times m$ matrix $M = [m_{ij}]$ where

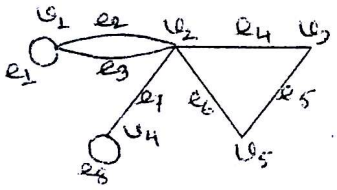
$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i \\ 0 & \text{otherwise} \end{cases}$$

Example: Find the Incidence matrix of the following graph.



	e_1	e_2	e_3	e_4	e_5	e_6
v_1	1	1	0	0	0	0
v_2	0	0	1	1	0	1
v_3	0	0	0	0	1	1
v_4	1	0	1	0	0	0
v_5	0	1	0	1	1	0

Example: Find the Incidence matrix of the following pseudograph.



	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
v_1	1	1	1	0	0	0	0	0
v_2	0	1	1	1	0	1	1	0
v_3	0	0	0	1	1	0	0	0
v_4	0	0	0	0	0	0	1	1
v_5	0	0	0	0	1	1	0	0

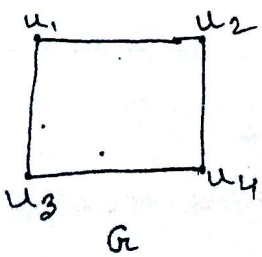
- Incidence matrix can be used to represent multiple edges and loops. Multiple edges are represented in the incidence matrix-using column with identical entries, since these edges are incidence with the same pair of vertices. Loops are represented using a column with exactly one entry equal to 1, corresponding to the vertex that is incident with this loop.

Isomorphism of Graphs

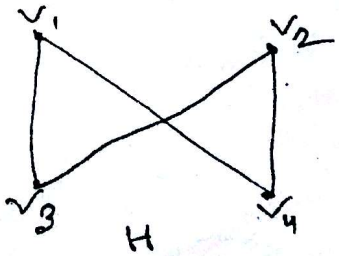
The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there is a one-to-one and onto function *f* from V_1 to V_2 with the property that a and b are adjacent in G_1 if and only if *f*(a) and *f*(b) are adjacent in G_2, for all a and b in V_1. Such a function *f* is called an *isomorphism*.

- The word *isomorphism* comes from the Greek roots *isos* for "equal" and *morphe* for "form".
- Isomorphism of simple graphs is an equivalence relation.

Example: Show that the following two graphs G = (V,E) and H = (W,F) are isomorphic.
Solution: The function *f* with *f*(u1) = v1, *f*(u2) = v4, *f*(u3) = v3 and *f*(u4) = v2 is a one-to-one correspondence between V and W. So, they are isomorphic.



G



H

Scanned by CamScanner

- It is often difficult to determine whether two simple graphs are isomorphic. Because there are n! possible one-to-one correspondences between the vertex sets of two simple graphs with n vertices.
- However, we can often show that two simple graphs are not isomorphic by showing that they do not share a property that isomorphic simple graphs must both have. Such a property is called ***an invariant*** *with respect to isomorphism of simple graphs*. For example, isomorphic simple graphs must have the same number of vertices. Furthermore, they must have the same number of edges. In addition, the degrees of the vertices in isomorphic simple graphs must be same.

Example- 9, 11: See Rosen [Page-561-62, 5th Edition].

- The best algorithms known for determining whether two graphs are isomorphic have exponential worst-case time complexity. However, linear average-case time complexity algorithms are known that solve this problem. And there is some hope that an algorithm with polynomial worst-case time complexity for determining whether two graphs are isomorphic can be found.

Graph Connectivity

- Let n be a nonnegative integer and G is an undirected graph. A *path of length n* from u to v in G is a sequence of n edges $e_1, \ldots, e_n$ of G such that $f(e_1) = \{x_0, x_1\}$, $f(e_2) = \{x_1, x_2\}, \ldots\ldots, f(e_n) = \{x_{n-1}, x_n\}$, where $x_0 = u$ and $x_n = v$. When the graph is simple, we denote this path by its vertex sequence $x_0, x_1, \ldots, x_n$.
- The path is a *circuit* if it begins and ends at the same vertex, that is $u = v$ and has length greater than zero. The *path* or *circuit* is said to *pass through* the vertices $x_1, x_2, \ldots, x_{n-1}$ or *traverse* the edges $e_1, e_2, \ldots, e_n$.
- A path or circuit is *simple* if it does not contain the same edge more than once.
- ❖ [SOS] A *simple path* is a path in which all the vertices are distinct. A *cycle* is a closed path in which all vertices are distinct except $v_0 = v_n$. A *trial* is a path where all the edges are distinct.
- In a graph, any circuit/cycle must have length 3 or more.

Example-1: See Rosen [page-567, 5th Edition].

- Let n be a nonnegative integer and G is a ***directed multigraph***. A *path of length n* from u to v in G is a sequence of n edges $e_1, \ldots, e_n$ of G such that $f(e_1) = (x_0, x_1)$, $f(e_2) = (x_1, x_2), \ldots\ldots, f(e_n) = (x_{n-1}, x_n)$, where $x_0 = u$ and $x_n = v$. When there are no multiple edges in the graph, this path is denoted by its vertex sequence $x_0, x_1, \ldots, x_n$.

Connectedness in Undirected Graph

An undirected graph is called *connected* if there is a path between every pair of distinct vertices of the graph.

Example-5: See Rosen [page-570, 5th Edition].

Theorem

There is a simple path between every pair of distinct vertices of a connected undirected graph.

Scanned by CamScanner

Connected Component

A graph that is not connected is the union of two or more connected subgraphs, each pair of which has no vertex in common. These disjoint connected subgraph are called the *connected component* of the graph.

Exmple-6: See Rosen [page-570, 5th Edition].

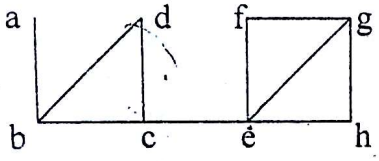
Cut vertices

Sometimes the removal of a vertex and all edges incident with it produces a subgraph with more connected components than in the original graph. Such vertices are called *cut vertices* (or *articulation points*). The removal of a cut vertex from a connected graph produces a subgraph that is not connected.

Cut edges

An edge whose removal produces a graph with more connected components than in the original graph is called a *cut edge* or *bridge*.

Example: Find the cut vertices and cut edges of the graph G in the following Fig.



The cut vertices of G are b, c and e.
The cut edges are {a, b}and {c, e}.

Distance and Diameter

Let G be a connected graph. The *distance* between vertices u and v in G, written *d(u,v)* is the length of the shortest path between u and v. The *diameter* of G, written *diam(g)*, is the maximum distance between any two points in G.

Example: In Fig-a, d(A,F) = 2 and diam(G) = 3 and in Fig-b, d(A,F) = 3 and diam(G) = 4.

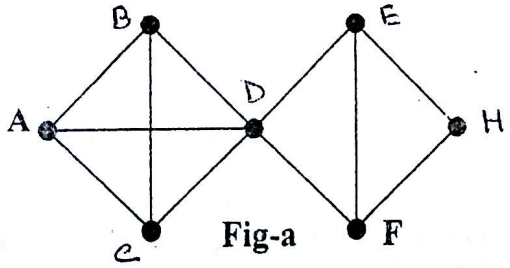


Fig-a

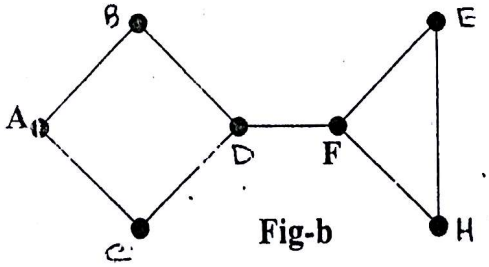


Fig-b

Connectedness in Directed Graph

Strongly Connected

A directed graph is strongly connected if there is a path from a to b and b to a whenever a and b are vertices in the graph.

Weakly Connected

A directed graph is weakly connected if there is a path between any two vertices in the underlying undirected graph.

Example-9: See Rosen [page-572, 5th Edition].

Scanned by CamScanner

Strongly connected components

The subgraphs of a directed graph G that are strongly connected but not contained in larger strongly connected subgraphs, that is, the maximal strongly connected subgraphs, are called the strongly connected components or strong components of G.

Example-10: See Rosen [page-572, 5th Edition].

Paths and Isomorphism

A useful isomorphic invariant for simple graphs is the existence of a simple circuit of length k, where k is a positive integer greater than 2.

Example-12: See Rosen [page-573, 5th Edition].

Counting paths between vertices

The numbers of paths between two vertices in a graph can be determine using its adjacency matrix.

Theorem

Let G be a graph with adjacency matrix **A** with respect to the ordering $v_1, v_2 \ldots, v_n$ (with directed or undirected edges, with multiple edges and loops allowed). The number of different paths of length r from v_i to v_j, where r is a positive integer, equals the (i, j) th entry of $\mathbf{A}^r$.

Example-14: See Rosen [page-574, 5th Edition].

Mir@cle Point
আলেয়া ফটোগ্রাফির নীচে
কেয়ারীর সামনে, কলেজ রোড, চকবাজার, চট্টগ্রাম।
IIUC এর সকল লেকচার শীট ও পূর্ববর্তী
সেমিস্টারের প্রশ্ন পাওয়া যায়।
মোবাইল : 01775943269

Scanned by CamScanner