UNIT 5:

Graph and Trees

What is Graph

In Mathematics, a graph is a pictorial representation of any data in an organised manner. The graph shows the relationship between variable quantities. In a graph theory, the graph represents the set of objects, that are related in some sense to each other. The objects are basically mathematical concepts, expressed by vertices or nodes and the relation between the pair of nodes, are expressed by edges.

Graph Theory is the study of points and lines. In Mathematics, it is a sub-field that deals with the study of graphs. It is a pictorial representation that represents the Mathematical truth. Graph theory is the study of relationship between the vertices (nodes) and edges (lines).

Formally, a graph is denoted as a pair G(V, E).

Where V represents the finite set vertices and E represents the finite set edges.

Therefore, we can say a graph includes non-empty set of vertices V and set of edges E.

Example

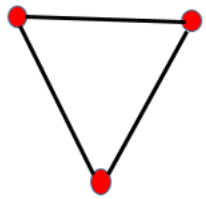
Suppose, a Graph G=(V,E), where

Vertices, V={a,b,c,d}

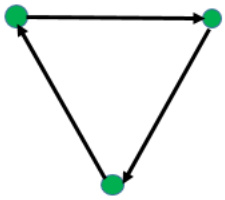
Edges, E={{a,b},{a,c},{b,c},{c,d}}

Types of Graph

The graphs are basically of two types, directed and undirected. It is best understood by the figure given below. The arrow in the figure indicates the direction.



Undirected Graph



Directed Graph

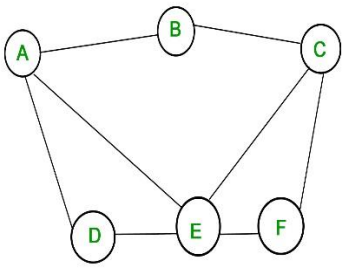
Directed Graph

In graph theory, a directed graph is a graph made up of a set of vertices connected by edges, in which the edges have a direction associated with them.

Undirected Graph

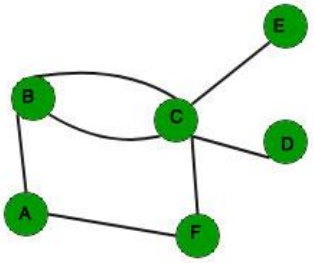
The undirected graph is defined as a graph where the set of nodes are connected together, in which all the edges are bidirectional. Sometimes, this type of graph is known as the undirected network.

Simple graph – A graph in which each edge connects two **different** vertices and where no two edges connect the same pair of vertices is called a simple graph. For example, Consider the following graph –



The above graph is a simple graph, since no vertex has a self-loop and no two vertices have more than one edge connecting them.
The edges are denoted by the vertices that they connect- {A,B}is the edge connecting vertices A and B.

2. Multigraph – A graph in which multiple edges may connect the same pair of vertices is called a multigraph.
Since there can be multiple edges between the same pair of vertices,
the **multiplicity** of edge tells the number of edges between two vertices.



The above graph is a multigraph since there are multiple edges between B and C. The multiplicity of the edge {B,C}is 2.

Other types of graphs

- Null Graph: A graph that does not have edges.
- Simple graph: A graph that is undirected and does not have any loops or multiple edges.
- Multigraph: A graph with multiple edges between the same set of vertices. It has loops formed.
- Connected graph: A graph where any two vertices are connected by a path.
- Disconnected graph: A graph where any two vertices or nodes are disconnected by a path.
- Cycle Graph: A graph that completes a cycle.
- Complete Graph: When each pair of vertices are connected by an edge then such graph is called a complete graph
- Planar graph: When no two edges of a graph intersect and are all the vertices and edges are drawn in a single plane, then such a graph is called a planar graph

Properties of Graph

- The starting point of the network is known as root.
- When the same types of nodes are connected to one another, then the graph is known as an assortative graph, else it is called a disassortative graph.
- A cycle graph is said to be a graph that has a single cycle.
- When all the pairs of nodes are connected by a single edge it forms a complete graph.
- A graph is said to be in symmetry when each pair of vertices or nodes are connected in the same direction or in the reverse direction.
- When a graph has a single graph, it is a path graph.

Graph Representations

In graph theory, a graph representation is a technique to store graph into the memory of computer.

To represent a graph, we just need the set of vertices, and for each vertex the neighbors of the vertex (vertices which is directly connected to it by an edge). If it is a weighted graph, then the weight will be associated with each edge.

There are different ways to optimally represent a graph, depending on the density of its edges, type of operations to be performed and ease of use.

1. Adjacency Matrix

- Adjacency matrix is a sequential representation.
- It is used to represent which nodes are adjacent to each other. i.e. is there any edge connecting nodes to a graph.
- In this representation, we have to construct a nXn matrix A. If there is any edge from a vertex i to vertex j, then the corresponding element of A, $a^{i,j} = 1$, otherwise $a^{i,j} = 0$.

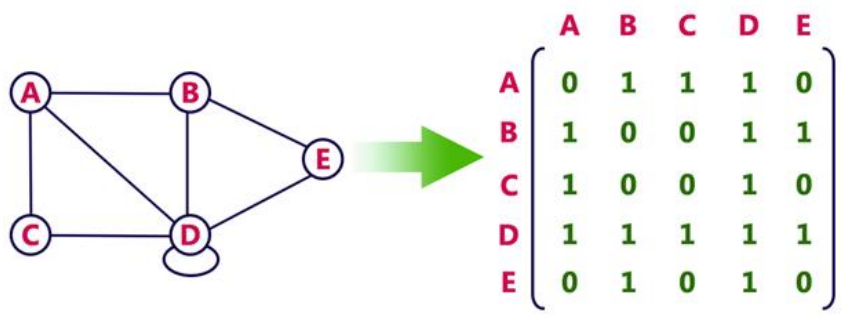
Note, even if the graph on 100 vertices contains only 1 edge, we still have to have a 100x100 matrix with lots of zeroes.

- If there is any weighted graph then instead of 1s and 0s, we can store the weight of the edge.

Example

Consider the following **undirected graph representation**:

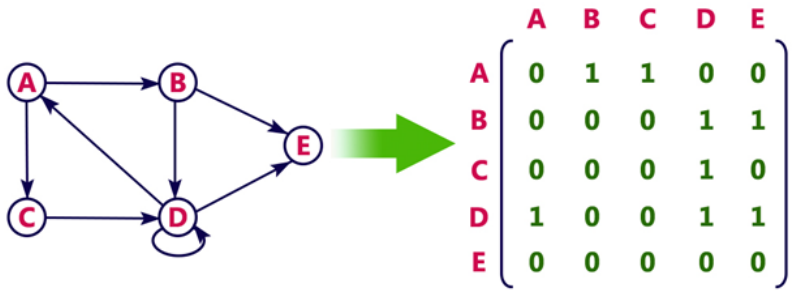
Undirected graph representation



	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	1	1
C	1	0	0	1	0
D	1	1	1	1	1
E	0	1	0	1	0

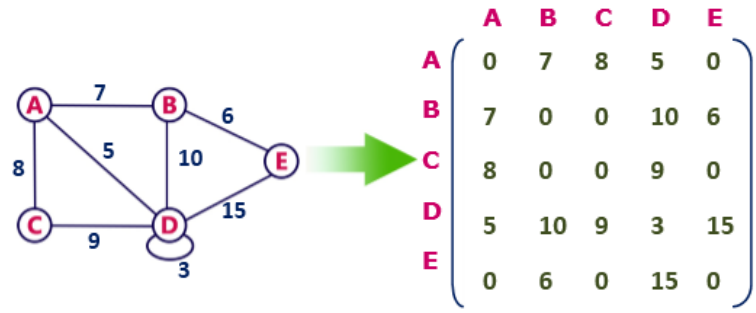
Directed graph represenation

See the directed graph representation:



In the above examples, 1 represents an edge from row vertex to column vertex, and 0 represents no edge from row vertex to column vertex.

Undirected weighted graph represenation



Pros: Representation is easier to implement and follow.

Cons: It takes a lot of space and time to visit all the neighbors of a vertex, we have to traverse all the vertices in the graph, which takes quite some time.

---

2. Incidence Matrix

In **Incidence matrix representation**, graph can be represented using a matrix of size:

Total number of vertices by total number of edges.

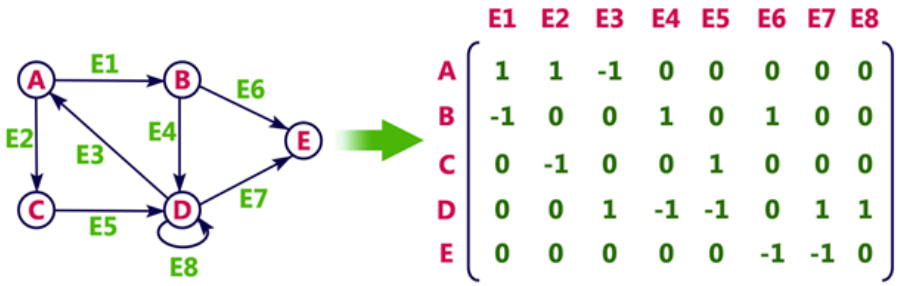
It means if a graph has 4 vertices and 6 edges, then it can be represented using a matrix of 4X6 class. In this matrix, columns represent edges and rows represent vertices.

This matrix is filled with either **0 or 1** or -1. Where,

- 0 is used to represent row edge which is not connected to column vertex.
- 1 is used to represent row edge which is connected as outgoing edge to column vertex.
- -1 is used to represent row edge which is connected as incoming edge to column vertex.

Example

Consider the following directed graph representation.



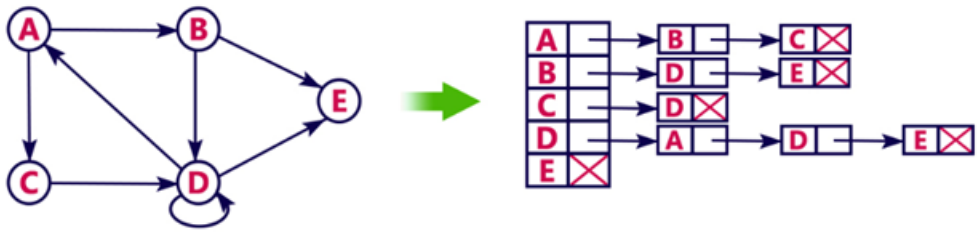
---

3. Adjacency List

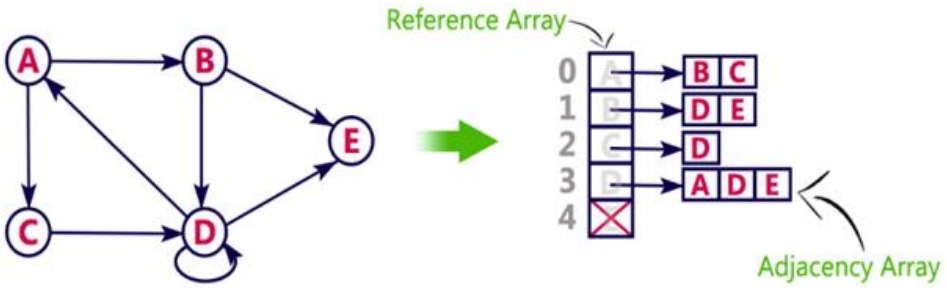
- Adjacency list is a linked representation.
- In this representation, for each vertex in the graph, we maintain the list of its neighbors. It means, every vertex of the graph contains list of its adjacent vertices.
- We have an array of vertices which is indexed by the vertex number and for each vertex v, the corresponding array element points to a **singly linked list** of neighbors of v.

Example

Let's see the following directed graph representation implemented using linked list:



We can also implement this representation using array as follows:



Pros:

- Adjacency list saves lot of space.
- We can easily insert or delete as we use linked list.
- Such kind of representation is easy to follow and clearly shows the adjacent nodes of node.

Cons:

- The adjacency list allows testing whether two vertices are adjacent to each other but it is slower to support this operation.

Representation of Graphs

There are two principal ways to represent a graph G with the matrix, i.e., adjacency matrix and incidence matrix representation.

(a)Representation of the Undirected Graph:

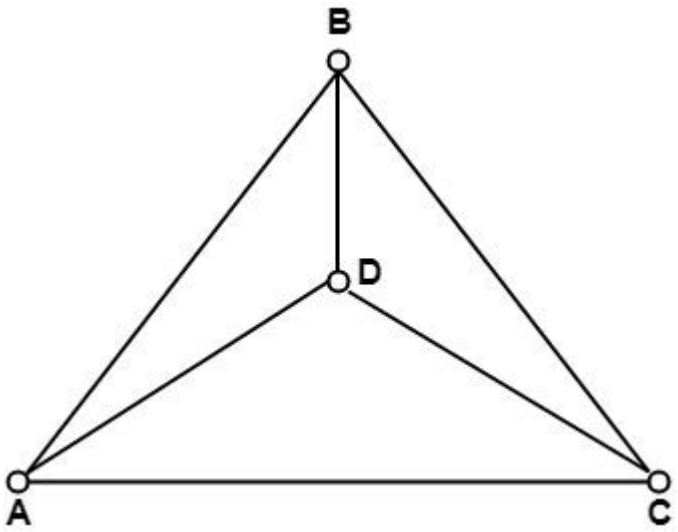
1. Adjacency Matrix Representation: If an Undirected Graph G consists of n vertices then the adjacency matrix of a graph is an n x n matrix A = [a_{ij}] and defined by

$$a_{ij} = \begin{bmatrix} 1, \text{if } \{V_i, V_j\} \text{is an edge i. e., } v_i \text{is adjacent to } v_j \\ 0, \text{if there is no edge between } v_i \text{ and } v_j \end{bmatrix}$$

If there exists an edge between vertex v_i and v_j, where i is a row and j is a column then the value of a_{ij}=1.

If there is no edge between vertex v_i and v_j, then value of a_{ij}=0.

Example: Find the adjacency matrix M_A of graph G shown in Fig:



Solution: Since graph G consist of four vertices. Therefore, the adjacency matrix wills a 4 x 4 matrix. The adjacency matrix is as follows in fig:

$$M_A = \begin{array}{c|cccc} & A & B & C & D \\ A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 1 & 1 \\ C & 1 & 1 & 0 & 1 \\ D & 1 & 1 & 1 & 0 \end{array}$$

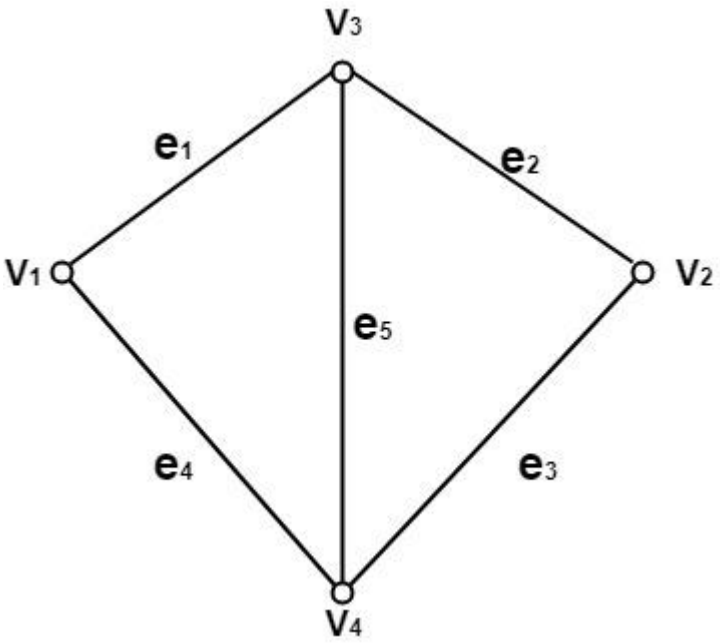
2. Incidence Matrix Representation: If an Undirected Graph G consists of n vertices and m edges, then the incidence matrix is an n x m matrix C = [c_{ij}] and defined by

$$c_{ij} = \begin{bmatrix} 1, \text{if the vertex } V_i \text{ incident by edge } e_j \\ 0, \text{otherwise} \end{bmatrix}$$

There is a row for every vertex and a column for every edge in the incident matrix.

The number of ones in an incidence matrix of the undirected graph (without loops) is equal to the sum of the degrees of all the vertices in a graph.

Example: Consider the undirected graph G as shown in fig. Find its incidence matrix M_I.



Solution: The undirected graph consists of four vertices and five edges. Therefore, the incidence matrix is an 4 x 5 matrix, which is shown in Fig:

$$M_I = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

General Trees

A graph which has no cycle is called an acyclic graph. A tree is an acyclic graph or graph having no cycles.

A tree or general trees is defined as a non-empty finite set of elements called vertices or nodes having the property that each node can have minimum degree 1 and maximum degree n. It can be partitioned into n+1 disjoint subsets such that the first subset contains the root of the tree and remaining n subsets includes the elements of the n subtree.

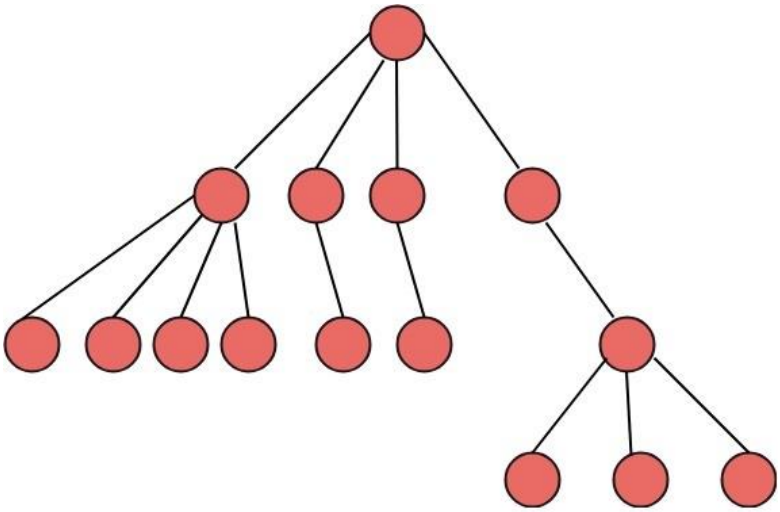
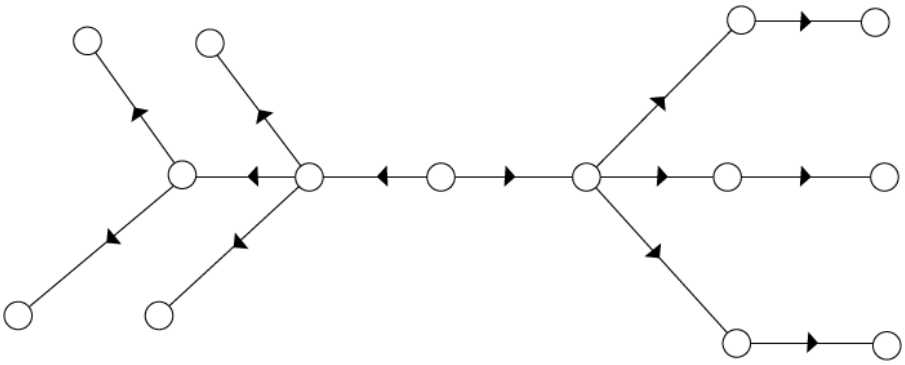


Fig:General Trees

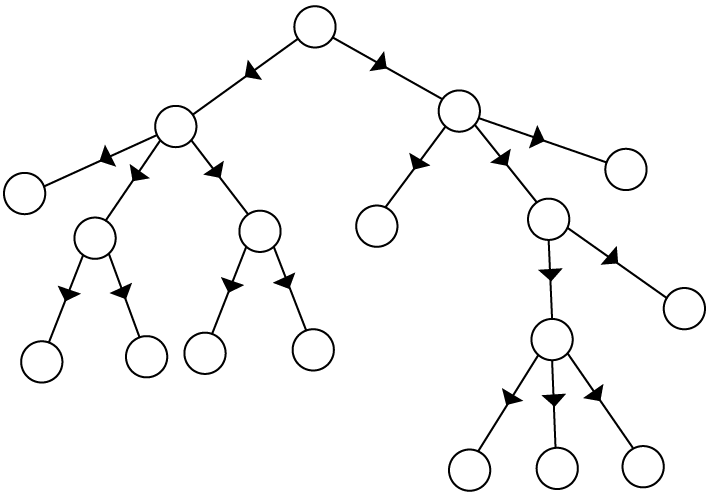
Directed Trees:

A directed tree is an acyclic directed graph. It has one node with indegree 1, while all other nodes have indegree 1 as shown in fig:



Directed Trees

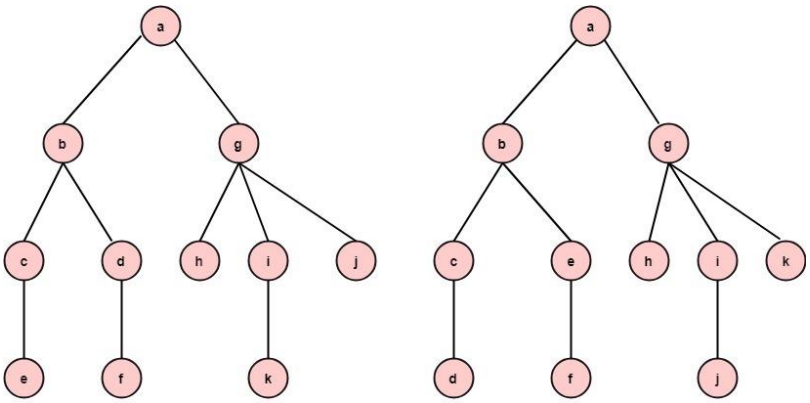
The node which has outdegree 0 is called an external node or a terminal node or a leaf. The nodes which have outdegree greater than or equal to one are called internal node.



Ordered Trees:

If in a tree at each level, an ordering is defined, then such a tree is called an ordered tree.

Example: The trees shown in the figures represent the same tree but have different orders.



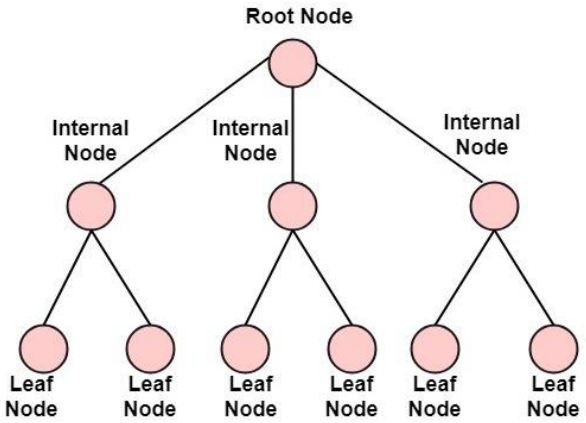
Properties of Trees:

1. There is only one path between each pair of vertices of a tree.
2. If a graph G there is one and only one path between each pair of vertices G is a tree.
3. A tree T with n vertices has n-1 edges.
4. A graph is a tree if and only if it a minimal connected.

Rooted Trees:

If a directed tree has exactly one node or vertex called root whose incoming degrees is 0 and all other vertices have incoming degree one, then the tree is called rooted tree.

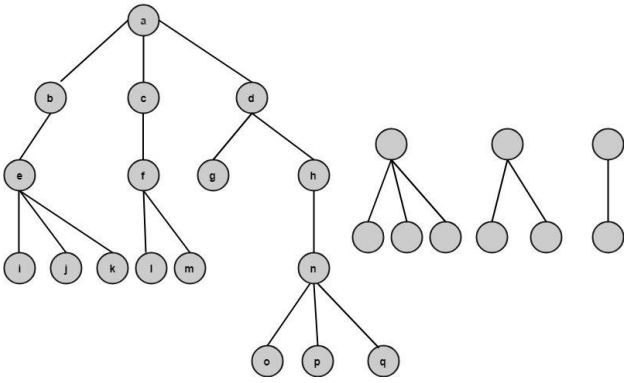
Note: 1. A tree with no nodes is a rooted tree (the empty tree)
2. A single node with no children is a rooted tree.



Path length of a Vertex:

The path length of a vertex in a rooted tree is defined to be the number of edges in the path from the root to the vertex.

Example: Find the path lengths of the nodes b, f, l, q as shown in fig:



Solution: The path length of node b is one.
The path length of node f is two.
The path length of node l is three
The path length of the node q is four.

Binary Trees:

If the outdegree of every node is less than or equal to 2, in a directed tree than the tree is called a binary tree. A tree consisting of the nodes (empty tree) is also a binary tree. A binary tree is shown in fig:

Basic Terminology:

Root: A binary tree has a unique node called the root of the tree.

Left Child: The node to the left of the root is called its left child.

Right Child: The node to the right of the root is called its right child.

Parent: A node having a left child or right child or both are called the parent of the nodes.

Siblings: Two nodes having the same parent are called siblings.

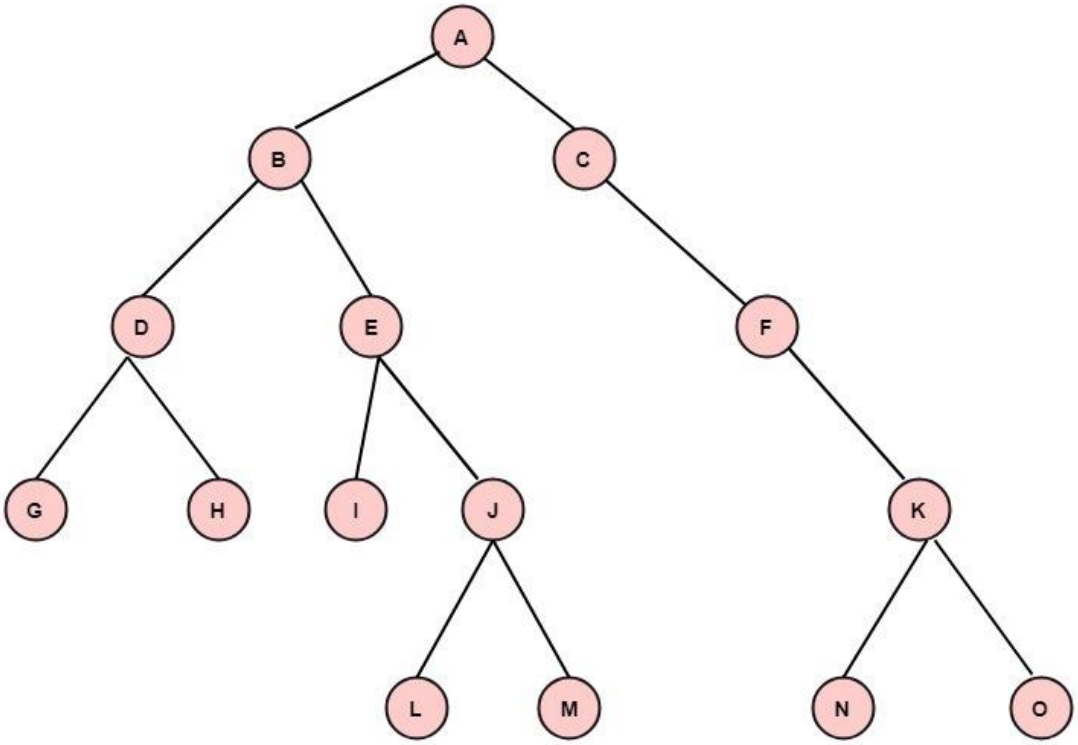
Leaf: A node with no children is called a leaf. The number of leaves in a binary tree can vary from one (minimum) to half the number of vertices (maximum) in a tree.

Descendant: A node is called descendant of another node if it is the child of the node or child of some other descendant of that node. All the nodes in the tree are descendants of the root.

Left Subtree: The subtree whose root is the left child of some node is called the left subtree of that node.

Example: For the tree as shown in fig:

- Which node is the root?
- Which nodes are leaves?
- Name the parent node of each node



Solution: (i) The node A is the root node.
(ii) The nodes G, H, I, L, M, N, O are leaves.
(iii)

Nodes	**Parent**
B, C	A
D, E	B
F	C
G, H	D
I, J	E
K	F
L, M	J
N, O	K

Right Subtree: The subtree whose root is the right child of some node is called the right subtree of that node.

Level of a Node: The level of a node is its distance from the root. The level of root is defined as zero. The level of all other nodes is one more than its parent node. The maximum number of nodes at any level N is 2^N.

Depth or Height of a tree: The depth or height of a tree is defined as the maximum number of nodes in a branch of a tree. This is more than the maximum level of the tree, i.e., the depth of root is one. The maximum number of nodes in a binary tree of depth d is 2^d-1, where d ≥1.

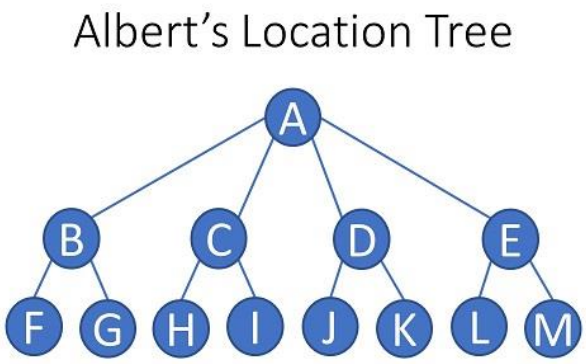
External Nodes: The nodes which have no children are called external nodes or terminal nodes.

Internal Nodes: The nodes which have one or more than one children are called internal nodes or non-terminal nodes.

Linear structures are easy to search. This lesson looks at the slightly trickier problem of searching a tree structure. Three algorithms are used to make sure that all nodes are searched.

Searching the Tree Structure

We were supposed to go out to eat tonight, but Albert lost his keys again! I asked Albert to write down the places he has been today. Instead of a list, Albert drew his travels as tree structure. He then began to ponder about in what order to start the search so he wouldn't miss anywhere. While he worked, I found his keys in the door of the room he had just entered. For fun, let's see how Albert would have found the keys his way.



Search Methods

The diagram that Albert drew was a **tree** structure. The tree is composed of items, called **nodes**, with connecting lines called **edges**. An end node, connected by only one edge, is called a **leaf**. Trees are often used in discrete math to organize information and make decisions. Albert's tree

happens to be a **search tree**. That means that data has been organized based on some criteria for ease of access. Because each fork of the tree has at most two branches, this is called a **binary search tree**.

Albert has three different methods or algorithms to help him search his tree:

Pre-order

Method 1 starts at the root and always bears left. You only move to the right when you have exhausted all the nodes to your left. Tracing the tree in this order gives Albert A-B-F-G-C-H-I-D-J-K-E-L-M

The real name for this method is **pre-order** traversal or enumeration.

Post-order

Method 2 starts at the left-most end node (or leaf) and works from the tips upward. You trim up the tree from the leaves to the root. You still bear left and work to the right, but with this scheme, you move from the bottom-up. Following this method, Albert got F-G-B-H-I-C-J-K-D-L-M-E-A

This is called **post-order** traversal or enumeration.

This is the most complicated, so let's follow the logic for clarity:

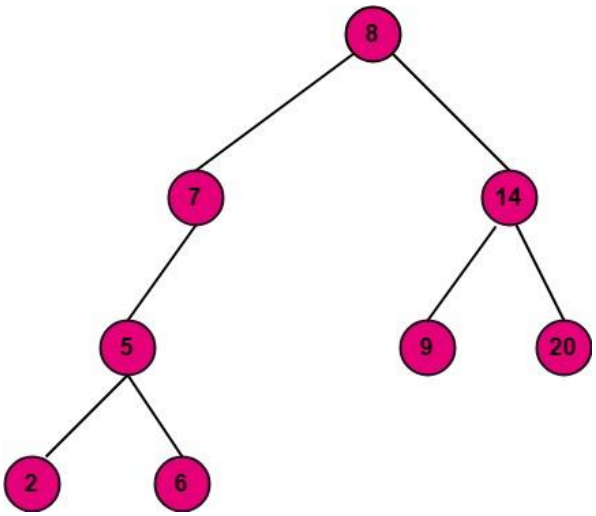
- Start with trimming off Leaf Node F
- Leaf Node G is the next leaf, so we trim off Node G
- Now, without Node F and Node G, Node B is a leaf
- Trim off Node B
- Looking at the remaining tree, Node H is the next leaf node
- Continue for remaining branches
- Finally, Node A becomes the last leaf

Binary Search Trees

Binary search trees have the property that the node to the left contains a smaller value than the node pointing to it and the node to the right contains a larger value than the node pointing to it.

It is not necessary that a node in a 'Binary Search Tree' point to the nodes whose value immediately precede and follow it.

Example: The tree shown in fig is a binary search tree.

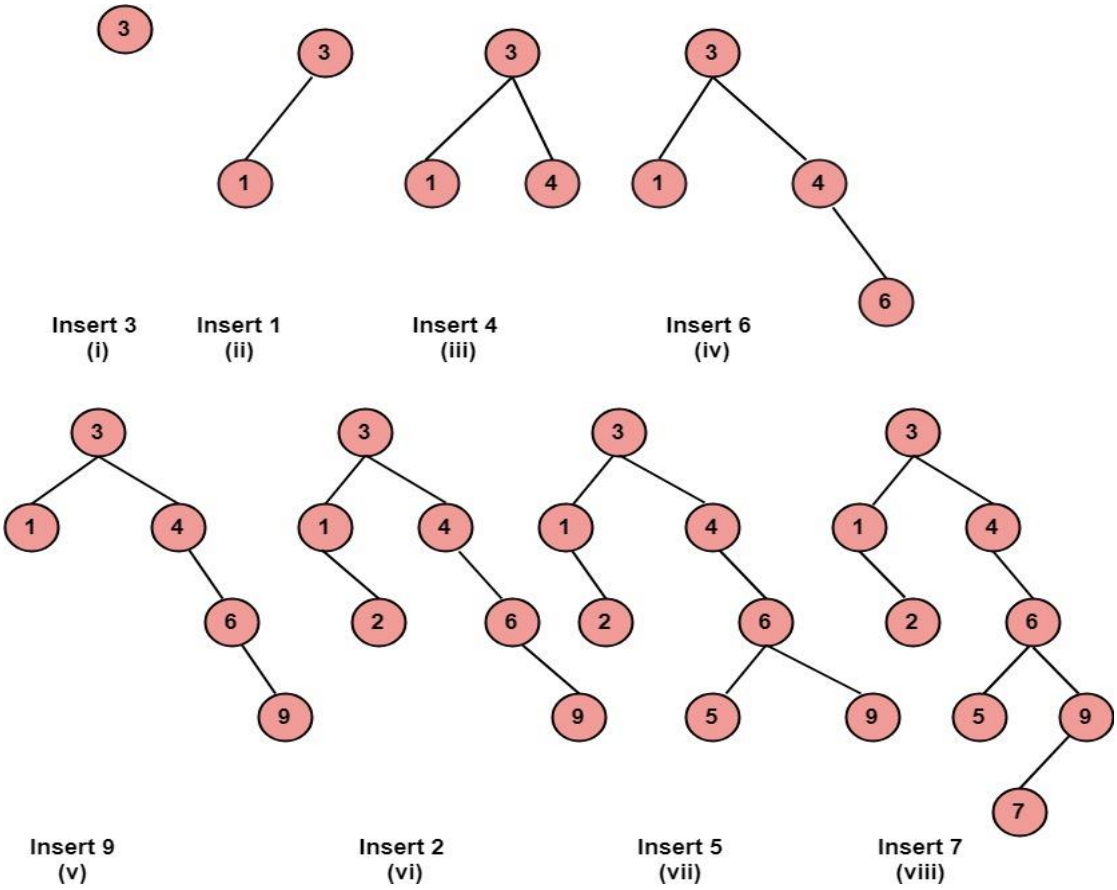


Inserting into a Binary Search Tree: Consider a binary tree T. Suppose we have given an ITEM of information to insert in T. The ITEM is inserted as a leaf in the tree. The following steps explain a procedure to insert an ITEM in the binary search tree T.

1. Compare the ITEM with the root node.
2. If ITEM>ROOT NODE, proceed to the right child, and it becomes a root node for the right subtree.
3. If ITEM<ROOT NODE, proceed to the left child.
4. Repeat the above steps until we meet a node which has no left and right subtree.
5. Now if the ITEM is greater than the node, then the ITEM is inserted as the right child, and if the ITEM is less than the node, then the ITEM is inserted as the left child.

Example: Show the binary search tree after inserting 3, 1,4,6,9,2,5,7 into an initially empty binary search tree.

Solution: The insertion of the above nodes in the empty binary search tree is shown in fig:



Deletion in a Binary Search Tree: Consider a binary tree T. Suppose we want to delete a given ITEM from binary search tree. To delete an ITEM from a binary search tree we have three cases, depending upon the number of children of the deleted node.

1. **Deleted Node has no children:** Deleting a node which has no children is very simple, as replace the node with null.
2. **Deleted Node has Only one child:** Replace the value of a deleted node with the only child.
3. **Deletion node has only two children:** In this case, replace the deleted node with the node that is closest in the value to the deleted node. To find the nearest value, we move once to the left and then to the right as far as possible. This node is called the immediate predecessor. Now replace the value of the deleted node with the immediate predecessor and then delete the replaced node by using case1 or case2.

Example: Show that the binary tree shown in fig (viii) after deleting the root node.

Solution: To delete the root node, first replace the root node with the closest elements of the root. For this, first, move one step left and then to the right as far as possible to the node.Then delete the replaced node. The tree after deletion shown in fig: