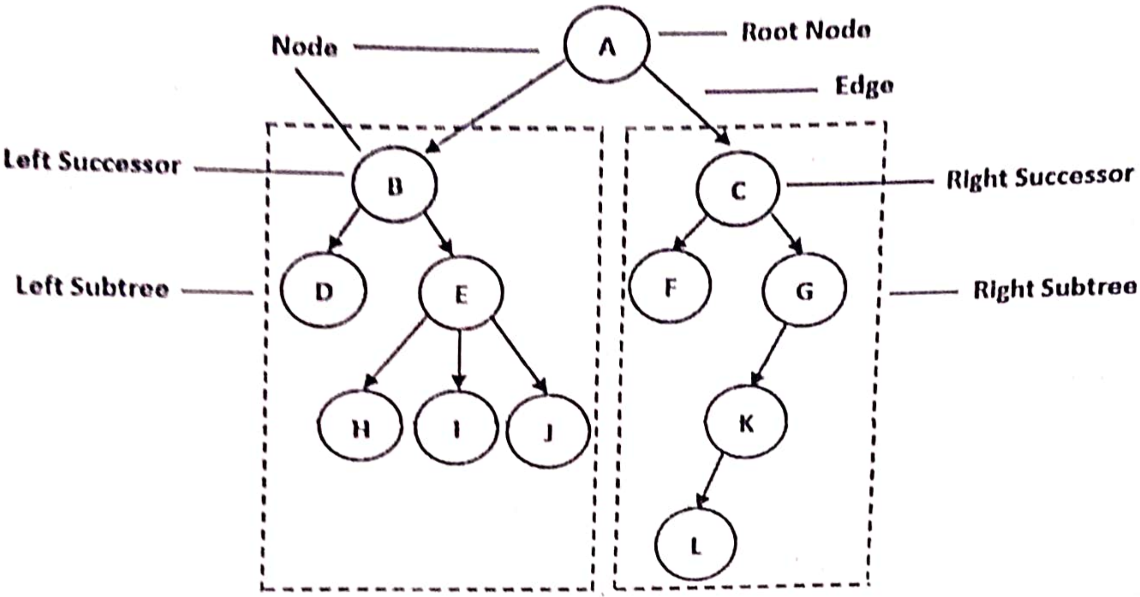
Reference: Data Structures With C By Seymour Lipschutz
For Educational Purpose Only. Not For Sale.

UNIT-V: Trees

Tree / Rooted tree graph:

- Tree is a non-linear Data Structure which represents/simulates the hierarchical relationship between various elements/ entities/objects.
- Tree is a graph that doesn't contain any cycle.
- A Tree is denoted by T. (Ex. T_{10} indicates a tree with 10 nodes)
- Examples: Records, Family trees and Table of contents.

Terminologies:



Node:

- Each element in a tree is called as node.
- Each node contains some data and links of other nodes (i.e. children).
- It is denoted by N, except root, root is denoted by R
- **Example:** In the above diagram A, B, C, D, E, F, G, H, I, J, K, L all are nodes

Root:

- The topmost (i.e. first) node in the tree is called as root node.
- The ROOT does not have any parent.
- It is denoted by R
- Example: In the above diagram node A is Root node.

Edge/ Link/ Reference/ Connection:

- The connection between one node to another.

Path:

- The sequence of nodes and edges connecting a node with a descendant.

Branch:

- A path ending with a terminal node is called as branch.

Left Child / Right Child:

- The child node is also called as Successor or Son.
- **Example:**
  In the above diagram:
  Node B is left child.
  Node C is right child.

Siblings or Brothers:

- Nodes with the same parent are called as siblings or brothers.
- **Example:**
  In the above diagram:
  Node B and Node C are brothers
  Node D and Node E are brothers
  Node F and Node G are brothers
  Node H, Node I, Node J are brothers

Ancestor:

- An ancestor of a node is any other node on the path from the node to the root.
- **Example:** In the above diagram ancestor of node H are node E, node B, node A

Descendant:

- An ancestor of a node is any other node on the path from the node to the root.
- A descendant is the inverse relationship of ancestor.
- **Example:** A node p is a descendant of a node q if and only if q is an ancestor of p.

Subtree:

- The tree which is formed by the child of a node is called as subtree.
- In binary tree, the tree which is formed by left child is called as left subtree.
- In binary tree, the tree which is formed by right child is called as right subtree

Degree of a node:

- The number of children of a node is called as degree of that node.
- Nodes that have degree zero are called leaf or terminal nodes.
- **Example:**
  In the above diagram:

Degree of Node A = 2	Degree of Node G = 1
Degree of Node B = 2	Degree of Node H = 0
Degree of Node C = 2	Degree of Node I = 0
Degree of Node D = 0	Degree of Node J = 0
Degree of Node E = 3	Degree of Node K = 1
Degree of Node F = 0	Degree of Node L = 0

Degree of Tree:

- The maximum (i.e. greatest) degree of a node is called as degree of that tree.
- **Example:** In the above diagram E have maximum childs (i.e. 3) so, the degree of tree is 3.

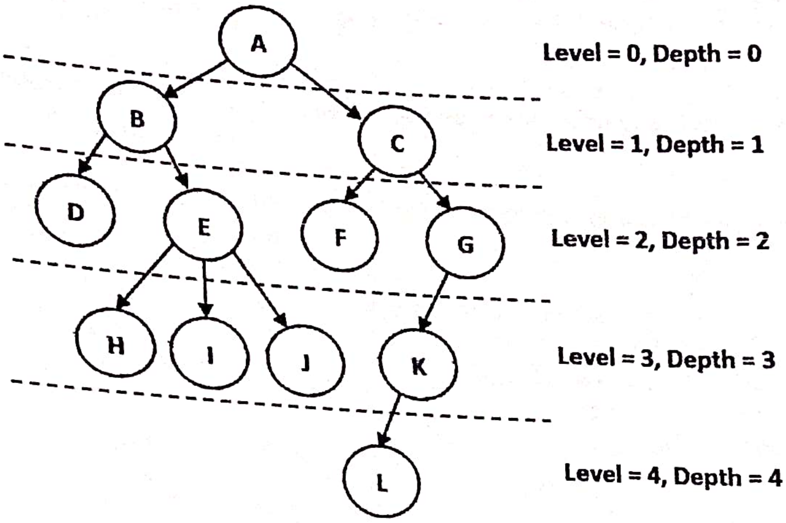
Terminal Node / Leaf Node/ External Node/ Outer Node:

- Any node which does not have any child is called as terminal node or leaf node or external node or outer node.
  OR
- A node with degree 0 (i.e. zero) is called as terminal node.
- **Example:**
  In the above diagram Node D, Node F, Node H, Node I, Node J and Node L are terminal nodes.

[redacted]

Non-Terminal Node/ Parent Node/ Internal Node/ Predecessor:

- Any node which has at least one child is called as non-terminal node.
  OR
- Any node whose degree is greater than zero, is called as non-terminal node.
- **Example:**
  In the above diagram Node A, Node B, Node C, Node E, Node G and Node K are Non-Terminal nodes.



Depth of a node:

- The depth of a node is the number of edges from the node to the tree's root node.
- **Example:**
  In above diagram:

Depth of Node A = 0	Depth of Node G = 2
Depth of Node B = 1	Depth of Node H = 3
Depth of Node C = 1	Depth of Node I = 3
Depth of Node D = 2	Depth of Node J = 3
Depth of Node E = 2	Depth of Node K = 3
Depth of Node F = 2	Depth of Node L = 4

Level of a node:

- The number of edges between the node and root is called as level of that node.
- The root of the tree has level 0
- The level of any other node in tree is +1 than the level of its parent node.
- **Example:**
  In the above diagram:

Level of Node A = 0	Level of Node G = 2
Level of Node B = 1	Level of Node H = 3
Level of Node C = 1	Level of Node I = 3
Level of Node D = 2	Level of Node J = 3
Level of Node E = 2	Level of Node K = 3
Level of Node F = 2	Level of Node L = 4

Height of Tree / Depth of Tree/ Height of Root:

- The depth or height of a Tree is the maximum number of nodes in the longest branch.
- The height of tree = Maximum (i.e. Greatest) Level index + 1

4

- The height of tree = total number of levels
- **Example:**
  In the above diagram:
  1. The height of tree = Greatest level index + 1 = (4 + 1) = 5
  2. The height of tree = total number of levels = (0 to 4) = 5

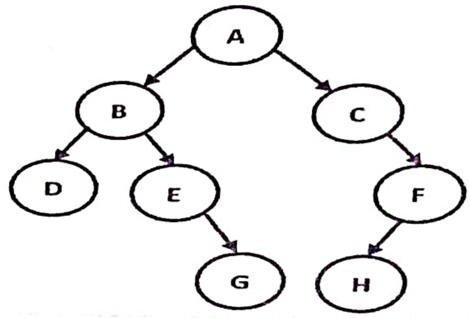
Intermediate Nodes:
- While traversing the tree the nodes which came in between the root node and terminal node are called as intermediate nodes.

Null Tree / Empty Tree:
- A tree which does not contain any node, not even Root node, such a tree is called as null tree or empty tree.

General Tree:
- A tree in which any node can have zero or more childs is called as general tree.

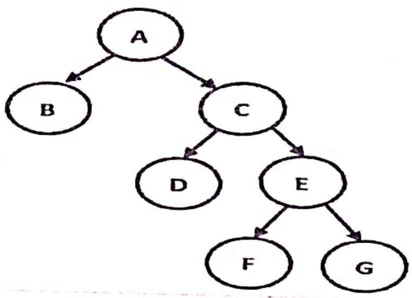
Binary Tree:
- A tree is said to be binary, if any node in that tree has 0, 1 or 2 childs.
- That means no node has more than two children.
- If the node has only one child then the child may be the left child or the child may be the right child.
- That means the node in a binary tree can have only left child or only right child.

Example:

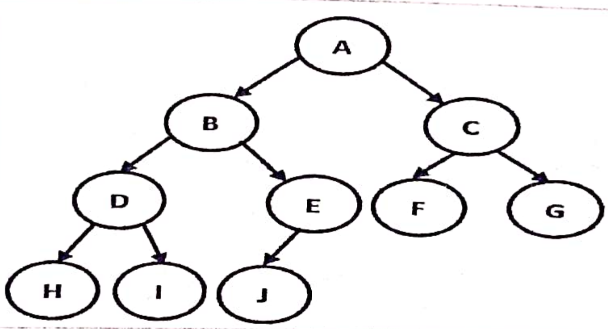


Full Binary Tree:
- A Binary Tree is a full binary tree if every node has 0 or 2 children.
- That means a full binary tree is a binary tree in which all nodes except leaf nodes have two children.
- 1 child is not allowed.

Example:

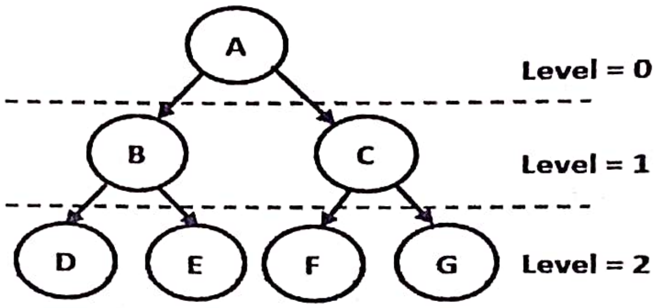


Complete Binary Tree:
- A Binary Tree is said to be a Complete Binary Tree if all the levels are completely filled except possibly the last level and all the nodes in the last level are placed as far left as possible.



Perfect Binary Tree:

- A Binary tree is said to be a Perfect Binary Tree, in which all the internal nodes have two children and all leaf nodes are at the same level.
- That means each level contains the maximum number of nodes i.e. every level is completely full of nodes.
- Maximum number of node in a level in a binary tree = 2 ^ level_index
  Example:
  Index of Root level is 0 so, it contains $2^0 = 1$ node.
- In a Perfect Binary Tree, the number of leaf nodes is the number of internal nodes + 1
  Example:
  In the following diagram, internal nodes are A, B and C. That means number of internal nodes = 3.
  So, the number of leaf nodes will be 3 + 1 = 4
- A Perfect Binary Tree has (2 ^ height of tree) - 1 nodes.
  Example:
  In the following diagram height of tree = greatest level index + 1 = 3
  So, total number of nodes = (2^3) - 1 = 8 - 1 = 7.



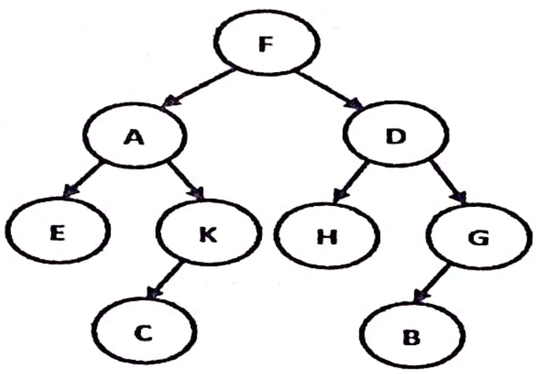
Representing Binary Trees In Memory:

The binary tree can be represented in memory in two ways as follows:

1. Sequential (i.e. Array) Representation of Binary Trees
2. Linked Representation of Binary Trees

Sequential (i.e. Array) Representation of Binary Trees:

- This representation uses a single dimensional array with name TREE
- The size of the array will be (2 ^ height of tree) - 1 nodes
- The root is stored at first position in the array
- The left child is stored at: parent_node_index * 2
- The right child is stored at: (parent_node_index * 2) + 1
- NULL (Φ) is stored in the array element if left or right child is not available
- **Example:**



6

Note:

- We assumed array indexes begins with 1.
- The height of the tree = Total number of nodes in the longest branch
- Branch: Path ending with terminal node is called as branch.
- The Longest branches in above tree are as follows:
  1. F -> A -> K -> C Total number of nodes = 4
  2. F -> D -> G -> B Total number of nodes = 4
- So, the height of the above tree = 4
- Total number of possible node in above tree = (2 ^ height of tree) – 1 = (2 ^ 4) – 1 = 16 – 1 = 15 nodes.
- The size of the array TREE will be 15.

F	A	D	E	K	H	G	NULL
TREE[1]	TREE[2]	TREE[3]	TREE[4]	TREE[5]	TREE[6]	TREE[7]	TREE[8]

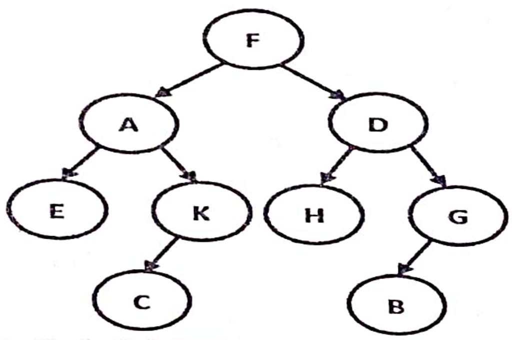
NULL	C	NULL	NULL	NULL	B	NULL
TREE[9]	TREE[10]	TREE[11]	TREE[12]	TREE[13]	TREE[14]	TREE[15]

Note:

- If a node does not have left and/or right child then the corresponding array element contains NULL. So, the sequential representation is suitable only for Perfect Binary Tree and Complete Binary Tree.

Linked Representation of Binary Tree:

- To represent a binary tree we can maintain Doubly Linked List.
- Here, the node in the doubly linked list is divided into three parts. These are INFO, LEFT and RIGHT.
- INFO is a regular variable which contains the value of node.
- LEFT is a pointer which points to the left child
- RIGHT is a pointer which points to the right child
- ROOT is a pointer variable which points to the root of the binary tree.
- If a node does not have left and /or right child then the corresponding LEFT and/or RIGHT pointer will contain NULL.
- **Example:**



ROOT (Pointer)
5

Note:

- The nodes will be stored at different addresses because the node in linked list is created dynamically so, any node can store at any available location in the memory.

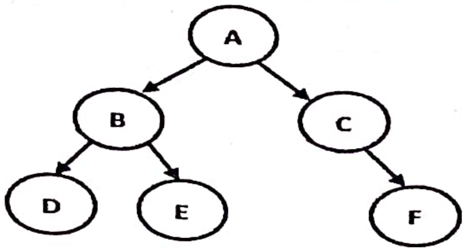
ADDRESS	INFO	LEFT	RIGHT
1	G	3	NULL
2	A	6	9
3	B	NULL	NULL
4	C	NULL	NULL
5	F	2	7
6	E	NULL	NULL
7	D	8	1
8	H	NULL	NULL
9	K	4	NULL

Traversing Binary Tree:

- A binary tree can be traverse in three different ways as follows:

Traversing Method	Description
Preorder **(Root, Left, Right)**	1. Process the root 2. Traverse the left subtree of root in preorder 3. Traverse the right subtree of root in preorder
Inorder **(Left, Root, Right)**	1. Traverse the left subtree of root in inorder 2. Process the root 3. Traverse the right subtree of root in inorder
Postorder **(Left, Right, Root)**	1. Traverse the left subtree of root in postorder 2. Traverse the right subtree of root in postorder 3. Process the root

Example-1: Traverse the following binary tree in Preorder, Inorder and Postorder.



Answer:

Preorder Traversal Sequence: (Root, Left, Right)

A	B	D	E	C	F

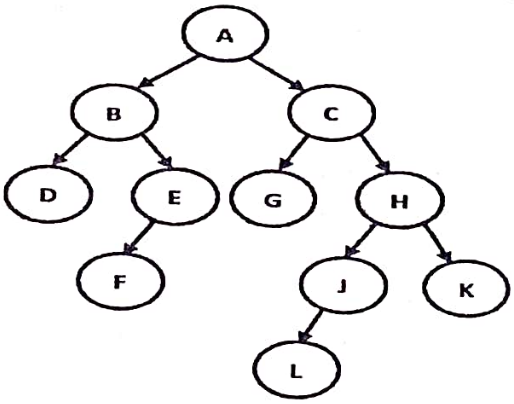
Inorder Traversal Sequence: (Left, Root, Right)

D	B	E	A	C	F

Postorder Traversal Sequence: (Left, Right, Root)

D	E	B	F	C	A

Example-2: Traverse the following binary tree in Preorder, Inorder and Postorder traversal sequence.



Answer:

Preorder Traversal Sequence: (Root, Left, Right)

A	B	D	E	F	C	G	H	J	L	K

Inorder Traversal Sequence: (Left, Root, Right)

D	B	F	E	A	G	C	L	J	H	K

Postorder Traversal Sequence: (Left, Right, Root)

D	F	E	B	G	L	J	K	H	C	A

Constructing a Binary Tree:

- To derive or construct binary tree from the traversal string, the ***inorder*** traversal is necessarily required without inorder traversal it is not possible to derive the tree.
- **Steps:**
  1. Identify the Root from the given preorder or postorder traversal sequence
  2. Check the given inorder traversal sequence and decide the nodes in the left and right subtree
  3. Apply the same process for to form left and right subtree

Example: Construct a binary tree by using following inorder and Preorder expressions

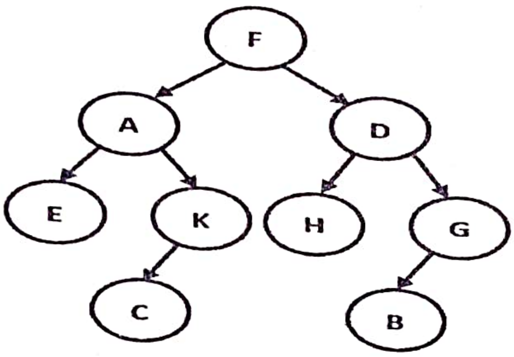
Inorder sequence: (Left, Root, Right)

E	A	C	K	F	H	D	B	G

Preorder sequence: (Root, Left, Right)

F	A	E	K	C	D	H	G	B

Answer:



Example-2: Construct a binary tree y using the following inorder and postorder traversal strings.

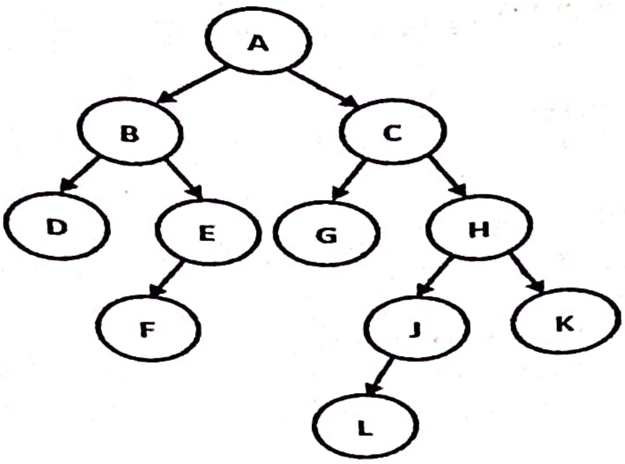
Postorder Traversal Sequence: (Left, Right, Root)

D	F	E	B	G	L	J	K	H	C	A

Inorder Traversal Sequence: (Left, Root, Right)

D	B	F	E	A	G	C	L	J	H	K

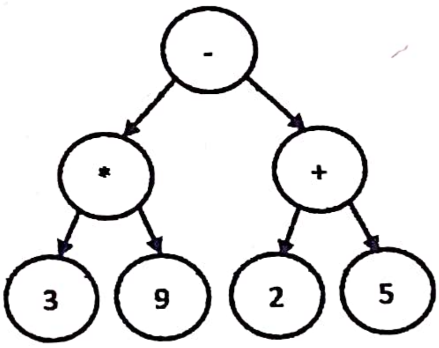
Answer:



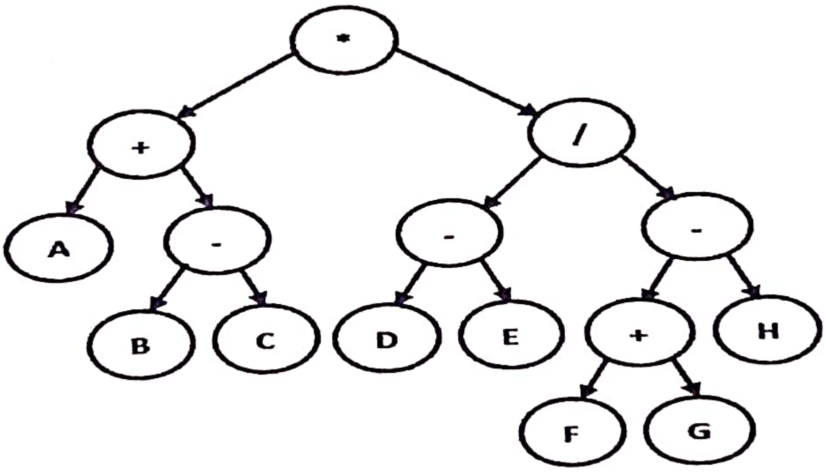
Application of Trees: Expression Tree

- Trees are used in compilers and interpreters.
- Trees can be used to represent arithmetic expressions.

Example-1: Consider the following algebraic expression
3 * 9 – (2 + 5)
Draw the binary tree for the above algebraic expression
Answer:



Example-2: Consider the following algebraic expression
(A + (B – C)) * ((D – E) / (F + G – H))
Draw the binary tree for the above algebraic expression and give the preorder, inorder and postorder traversal of the tree
Answer:



[redacted]

[redacted]

Preorder Traversal Sequence: (Root, Left, Right)

*	+	A	-	B	C	/	-	D	E	-	+	F	G	H

Inorder Traversal Sequence: (Left, Root, Right)

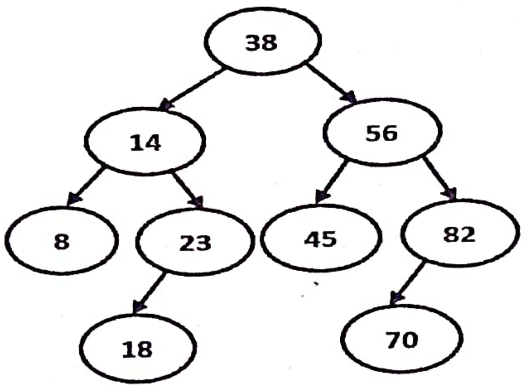
A	+	B	-	C	*	D	-	E	/	F	+	G	-	H

Postorder Traversal Sequence: (Left, Right, Root)

A	B	C	-	+	D	E	-	F	G	+	H	-	/	*

Binary Search Tree / Binary Sorted Tree/ BST:

- A binary tree is called as Binary Search Tree if each node in the tree has the following property:
- The value of node is greater than every value in the left subtree of that node and less than every value in the right subtree of that node.
- If the value is equals to the value of root or parent node then it will be placed in the right side of parent or root.
- Example:



Note:

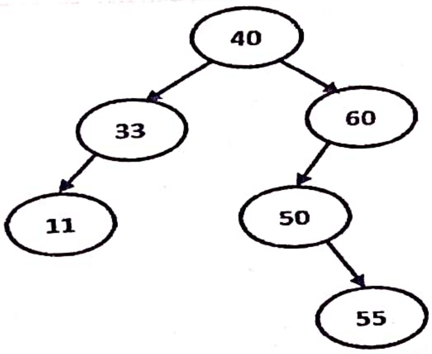
- In inorder traversal sequence of Binary Search Tree the elements will be arranged in ascending order.
- Inorder traversal sequence of above binary search tree: (Left, Root, Right)

8	14	18	23	38	45	56	70	82

Example: Construct a binary search tree by using the following numbers.
40, 60, 50, 33, 55, 11

Answer:

Consider the first value as root.



[redacted]

[redacted]