**Tree Data Structure-**

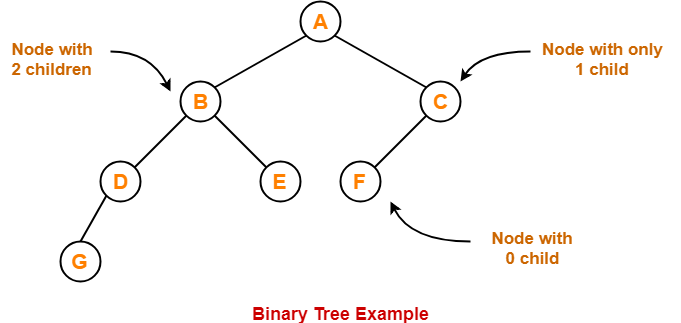
We have discussed-

* Tree is a non-linear data structure.
* In a tree data structure, a node can have any number of child nodes.

**Binary Tree-**

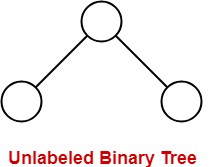
|  |
| --- |
| Binary tree is a special tree data structure in which each node can have at most 2 children.  Thus, in a binary tree,  Each node has either 0 child or 1 child or 2 children. |

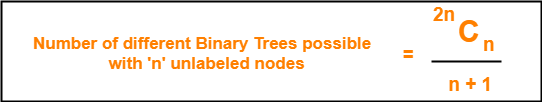
**Example-**



**Unlabeled Binary Tree-**

|  |
| --- |
| A binary tree is unlabeled if its nodes are not assigned any label. |





**Example-**

Consider we want to draw all the binary trees possible with 3 unlabeled nodes.

Using the above formula, we have-

Number of binary trees possible with 3 unlabeled nodes

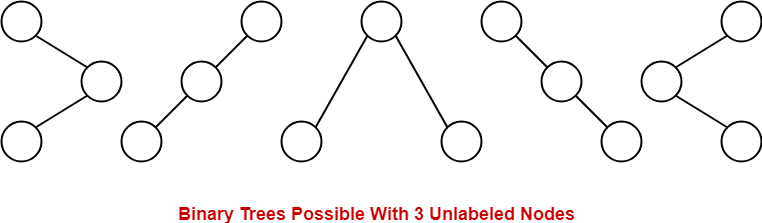
= 2 x 3C3 / (3 + 1)

= 6C3 / 4

= 5

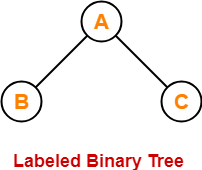
Thus,

* With 3 unlabeled nodes, 5 unlabeled binary trees are possible.
* These unlabeled binary trees are as follows-



**Labeled Binary Tree-**

|  |
| --- |
| A binary tree is labeled if all its nodes are assigned a label. |





**Example-**

Consider we want to draw all the binary trees possible with 3 labeled nodes.

Using the above formula, we have-

Number of binary trees possible with 3 labeled nodes

= { 2 x 3C3 / (3 + 1) } x 3!

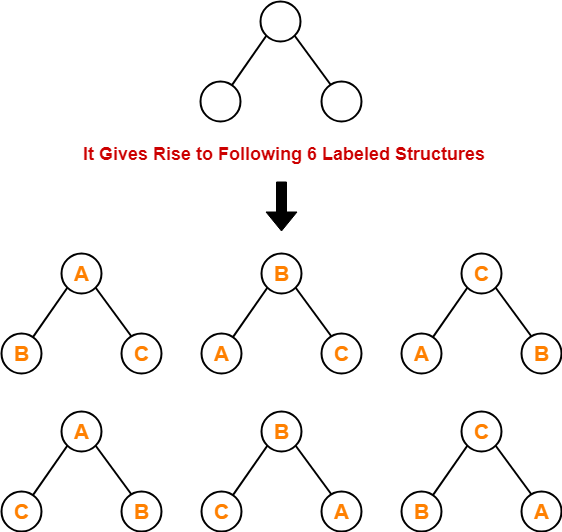
= { 6C3 / 4 } x 6

= 5 x 6

= 30

Thus,

* With 3 labeled nodes, 30 labeled binary trees are possible.
* Each unlabeled structure gives rise to 3! = 6 different labeled structures.

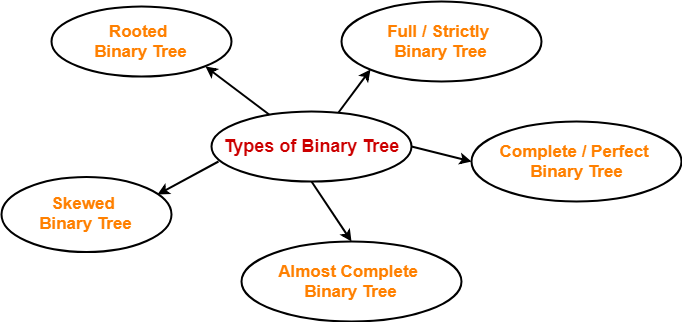


Similarly,

* Every other unlabeled structure gives rise to 6 different labeled structures.
* Thus, in total 30 different labeled binary trees are possible.

**Types of Binary Trees-**

Binary trees can be of the following types-



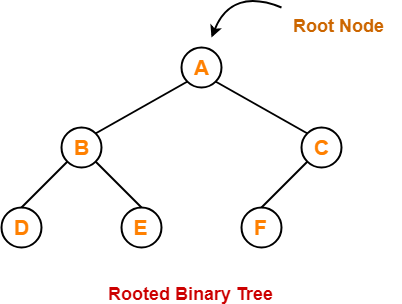
1. Rooted Binary Tree
2. Full / Strictly Binary Tree
3. Complete / Perfect Binary Tree
4. Almost Complete Binary Tree
5. Skewed Binary Tree

**1. Rooted Binary Tree-**

A **rooted binary tree** is a binary tree that satisfies the following 2 properties-

* It has a root node.
* Each node has at most 2 children.

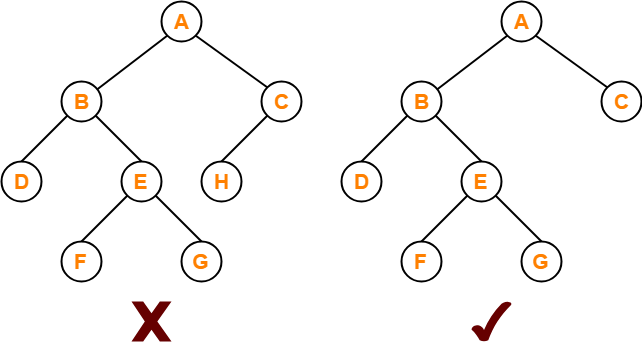
**Example-**



**2. Full / Strictly Binary Tree-**

* A binary tree in which every node has either 0 or 2 children is called as a **Full binary tree**.
* Full binary tree is also called as **Strictly binary tree**.

**Example-**



Here,

* First binary tree is not a full binary tree.
* This is because node C has only 1 child.

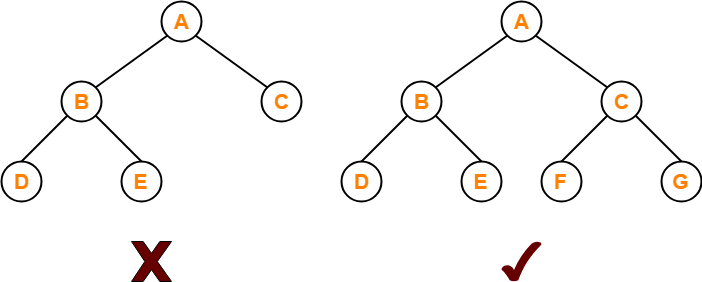
**3. Complete / Perfect Binary Tree-**

A **complete binary tree** is a binary tree that satisfies the following 2 properties-

* Every internal node has exactly 2 children.
* All the leaf nodes are at the same level.

Complete binary tree is also called as **Perfect binary tree**.

**Example-**



Here,

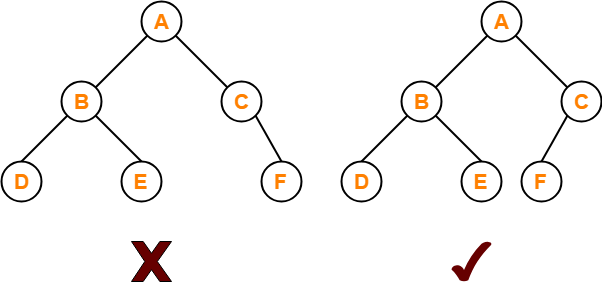
* First binary tree is not a complete binary tree.
* This is because all the leaf nodes are not at the same level.

**4. Almost Complete Binary Tree-**

An **almost complete binary tree** is a binary tree that satisfies the following 2 properties-

* All the levels are completely filled except possibly the last level.
* The last level must be strictly filled from left to right.

**Example-**



Here,

* First binary tree is not an almost complete binary tree.
* This is because the last level is not filled from left to right.

**5. Skewed Binary Tree-**

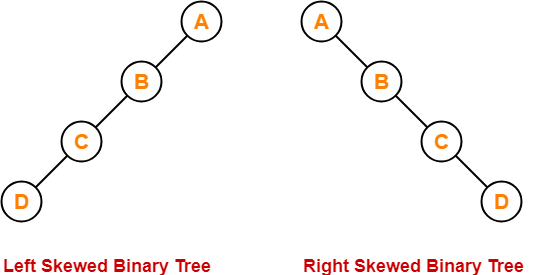
A **skewed binary tree** is a binary tree that satisfies the following 2 properties-

* All the nodes except one node has one and only one child.
* The remaining node has no child.

**OR**

A **skewed binary tree** is a binary tree of n nodes such that its depth is (n-1).

**Example-**



# [Tree Traversal | Binary Tree Traversal](https://www.gatevidyalay.com/tree-traversal-binary-tree-traversal/)

## ****Binary Tree-****

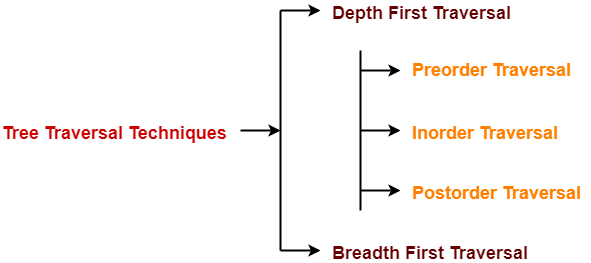
We have discussed-

* Binary tree is a special tree data structure.
* In a binary tree, each node can have at most 2 children.

## ****Tree Traversal-****

|  |
| --- |
| Tree Traversal refers to the process of visiting each node in a tree data structure exactly once. |

Various tree traversal techniques are-



## ****Depth First Traversal-****

The following three traversal techniques fall under Depth First Traversal-

1. Preorder Traversal
2. Inorder Traversal
3. Postorder Traversal

## ****1. Preorder Traversal-****

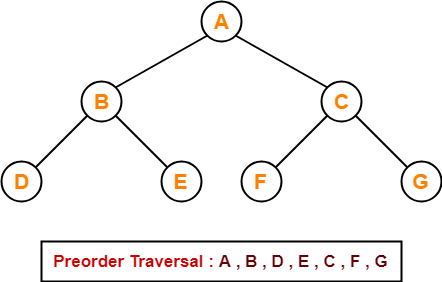
## ****Algorithm-****

1. Visit the root
2. Traverse the left sub tree i.e. call Preorder (left sub tree)
3. Traverse the right sub tree i.e. call Preorder (right sub tree)

**Root**→**Left**→**Right**

## ****Example-****

Consider the following example-



|  |
| --- |
| ****Preorder Traversal Shortcut****   Traverse the entire tree starting from the root node keeping yourself to the left.    https://www.gatevidyalay.com/wp-content/uploads/2018/07/Preorder-Traversal-Shortcut-1.png |

## ****Applications-****

* Preorder traversal is used to get the prefix expression of an expression tree.
* Preorder traversal is used to create a copy of the tree.

## ****2. Inorder Traversal-****

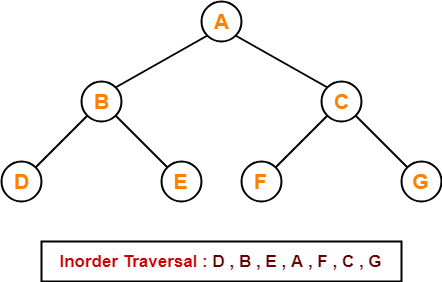
## ****Algorithm-****

1. Traverse the left sub tree i.e. call Inorder (left subtree)
2. Visit the root
3. Traverse the right subtree i.e. call Inorder (right subtree)

**Left**→**Root**→**Right**

## ****Example-****

Consider the following example-



|  |
| --- |
| ****Inorder Traversal Shortcut****   Keep a plane mirror horizontally at the bottom of the tree and take the projection of all the nodes.    https://www.gatevidyalay.com/wp-content/uploads/2018/07/Inorder-Traversal-Shortcut-1.png |

## ****Application-****

* Inorder traversal is used to get the infix expression of an expression tree.

## ****3. Postorder Traversal-****

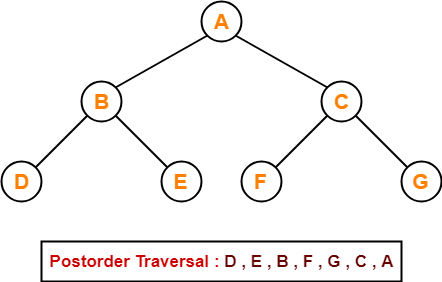
## ****Algorithm-****

1. Traverse the left sub tree i.e. call Postorder (left sub tree)
2. Traverse the right subtree i.e. call Postorder (right sub tree)
3. Visit the root

**Left**→**Right**→**Root**

## ****Example-****

Consider the following example-



|  |
| --- |
| ****Postorder Traversal Shortcut****   Pluck all the leftmost leaf nodes one by one.    https://www.gatevidyalay.com/wp-content/uploads/2018/07/Postorder-Traversal-Shortcut-1.png |

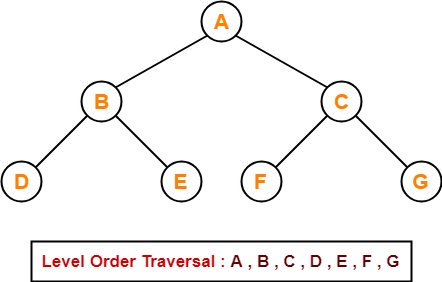
## ****Applications-****

* Postorder traversal is used to get the postfix expression of an expression tree.
* Postorder traversal is used to delete the tree.
* This is because it deletes the children first and then it deletes the parent.

## ****Breadth First Traversal-****

* Breadth First Traversal of a tree prints all the nodes of a tree level by level.
* Breadth First Traversal is also called as **Level Order Traversal**.

## ****Example-****



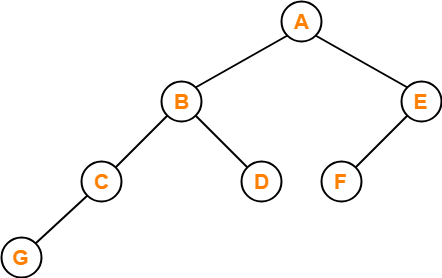
## ****Application-****

* Level order traversal is used to print the data in the same order as stored in the array representation of a complete binary tree

## ****PRACTICE PROBLEMS BASED ON TREE TRAVERSAL-****

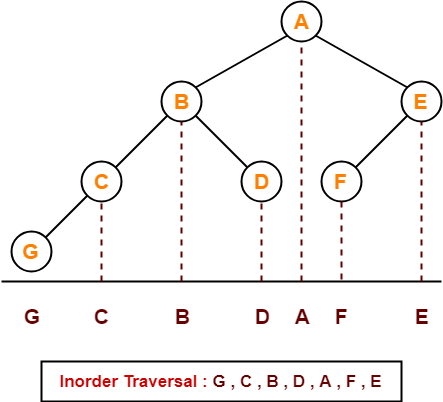
## ****Problem-01:****

If the binary tree in figure is traversed in inorder, then the order in which the nodes will be visited is \_\_\_\_?



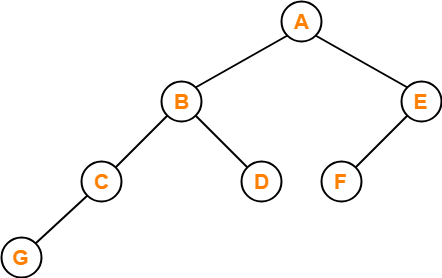
## ****Solution-****

The inorder traversal will be performed as-



## ****Problem-02:****

Which of the following sequences denotes the postorder traversal sequence of the tree shown in figure?



1. FEGCBDBA
2. GCBDAFE
3. GCDBFEA
4. FDEGCBA

## ****Solution-****

Perform the postorder traversal by plucking all the leftmost leaf nodes one by one.

Then,

Postorder Traversal : G , C , D , B , F , E , A

Thus, Option (C) is correct.

## ****Problem-03:****

Let LAST POST, LASTIN, and LASTPRE denote the last vertex visited in a postorder, inorder, and preorder traversal respectively of a complete binary tree. Which of the following is always true?

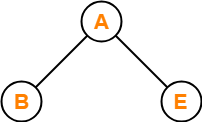
1. LASTIN = LASTPOST
2. LASTIN = LASTPRE
3. LASTPRE = LASTPOST
4. None of these

Let FIRSTPOST, FIRSTIN, FIRSTPRE denote the FIRST vertex visited in a postorder, inorder and preorder traversal respectively of a complete binary tree. Which of the following is always true?

1. FIRSTIN = FIRSTPOST
2. FIRSTIN = FIRSTPRE
3. FIRSTPRE = FIRSTPOST
4. None of these

## ****Solution-****

Consider the following complete binary tree-



Preorder Traversal : A ,B , E

Inorder Traversal : B , A , E

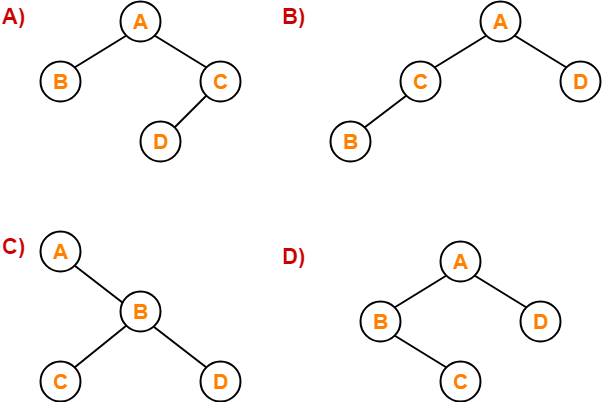
Postorder Traversal : B , E , A

Clearly, LASTIN = LASTPRE.

Thus, Option (B) is correct.

## ****Problem-04:****

Which of the following binary trees has its inorder and preorder traversals as BCAD and ABCD respectively-



## ****Solution-****

Option (D) is correct.

# [Binary Search Tree | Example | Construction](https://www.gatevidyalay.com/binary-search-trees-data-structures/)

## ****Binary Tree-****

We have discussed-

* Binary tree is a special tree data structure.
* In a binary tree, each node can have at most 2 children.
* In a binary tree, nodes may be arranged in any random order.

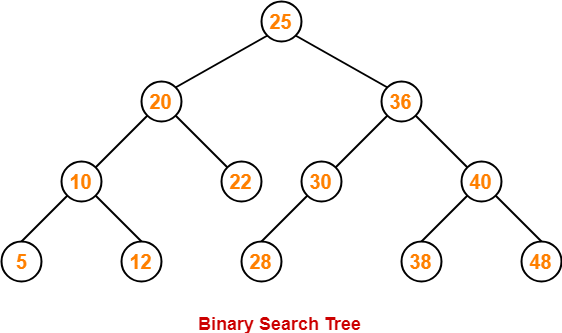
## ****Binary Search Tree-****

|  |
| --- |
| Binary Search Tree is a special kind of binary tree in which nodes are arranged in a specific order. |

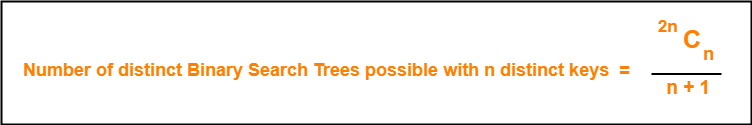
In a binary search tree (BST), each node contains-

* Only smaller values in its left sub tree
* Only larger values in its right sub tree

## ****Example-****



## ****Number of Binary Search Trees-****



## ****Example-****

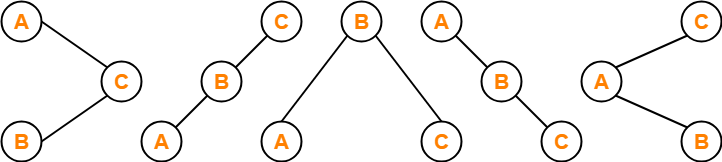
Number of distinct binary search trees possible with 3 distinct keys

= 2×3C3 / 3+1

= 6C3 / 4

= 5

If three distinct keys are A, B and C, then 5 distinct binary search trees are-



## ****Binary Search Tree Construction-****

Let us understand the construction of a binary search tree using the following example-

## ****Example-****

Construct a Binary Search Tree (BST) for the following sequence of numbers-

50, 70, 60, 20, 90, 10, 40, 100

When elements are given in a sequence,

* Always consider the first element as the root node.
* Consider the given elements and insert them in the BST one by one.

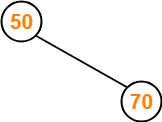
The binary search tree will be constructed as explained below-

### ****Insert 50-****

https://www.gatevidyalay.com/wp-content/uploads/2018/07/Binary-Search-Tree-Construction-Step-01.png

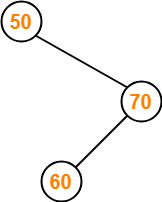
### ****Insert 70-****

* As 70 > 50, so insert 70 to the right of 50.



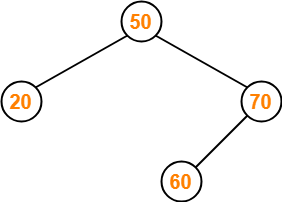
### ****Insert 60-****

* As 60 > 50, so insert 60 to the right of 50.
* As 60 < 70, so insert 60 to the left of 70.



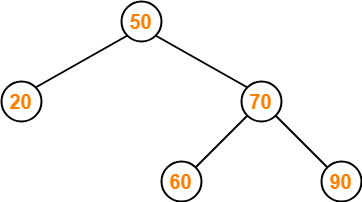
### ****Insert 20-****

* As 20 < 50, so insert 20 to the left of 50.



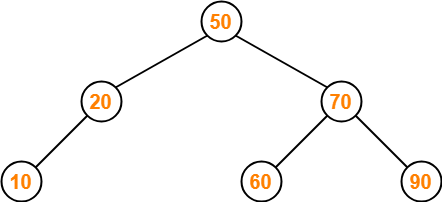
### ****Insert 90-****

* As 90 > 50, so insert 90 to the right of 50.
* As 90 > 70, so insert 90 to the right of 70.



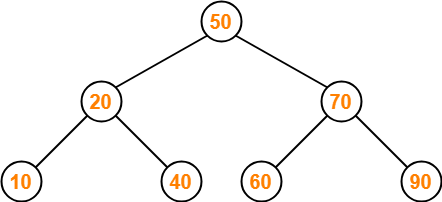
### ****Insert 10-****

* As 10 < 50, so insert 10 to the left of 50.
* As 10 < 20, so insert 10 to the left of 20.



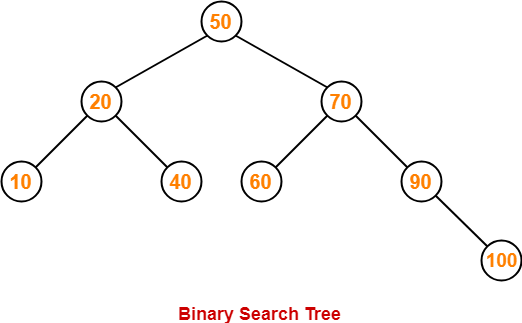
### ****Insert 40-****

* As 40 < 50, so insert 40 to the left of 50.
* As 40 > 20, so insert 40 to the right of 20.



### ****Insert 100-****

* As 100 > 50, so insert 100 to the right of 50.
* As 100 > 70, so insert 100 to the right of 70.
* As 100 > 90, so insert 100 to the right of 90.



This is the required Binary Search Tree.

## ****PRACTICE PROBLEMS BASED ON BINARY SEARCH TREES-****

## ****Problem-01:****

A binary search tree is generated by inserting in order of the following integers-

50, 15, 62, 5, 20, 58, 91, 3, 8, 37, 60, 24

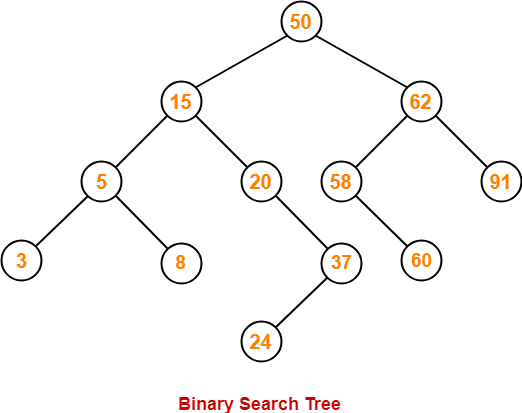
The number of nodes in the left subtree and right subtree of the root respectively is \_\_\_\_\_.

1. (4, 7)
2. (7, 4)
3. (8, 3)
4. (3, 8)

## ****Solution-****

Using the above discussed steps, we will construct the binary search tree.

The resultant binary search tree will be-



Clearly,

* Number of nodes in the left subtree of the root = 7
* Number of nodes in the right subtree of the root = 4

Thus, Option (B) is correct.

 NOVEMBER 07, 2023-IT

## ****Problem-02:****

How many distinct binary search trees can be constructed out of 4 distinct keys?

1. 5
2. 14
3. 24
4. 35

## ****Solution-****

Number of distinct binary search trees possible with 4 distinct keys

= 2nCn / n+1

= 2×4C4 / 4+1

= 8C4 / 5

= 14

Thus, Option (B) is correct.

## ****Problem-03:****

The numbers 1, 2, …, n are inserted in a binary search tree in some order. In the resulting tree, the right subtree of the root contains p nodes. The first number to be inserted in the tree must be-

1. p
2. p+1
3. n-p
4. n-p+1

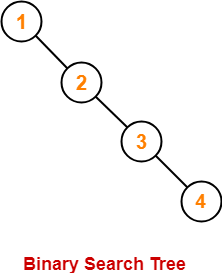
## ****Solution-****

Let n = 4 and p = 3.

Then, given options reduce to-

1. 3
2. 4
3. 1
4. 2

Our binary search tree will be as shown-



Clearly, first inserted number = 1.

Thus, Option (C) is correct.

## ****Problem-04:****

We are given a set of n distinct elements and an unlabeled binary tree with n nodes. In how many ways can we populate the tree with given set so that it becomes a binary search tree?

1. 0
2. 1
3. n!
4. C(2n, n) / n+1

## ****Solution-****

Option (B) is correct.

# [Binary Search Tree Traversal | BST Traversal](https://www.gatevidyalay.com/binary-search-tree-traversal-bst-traversal/)

## ****Binary Search Tree-****

Binary search tree (BST) is a special kind of binary tree where each node contains-

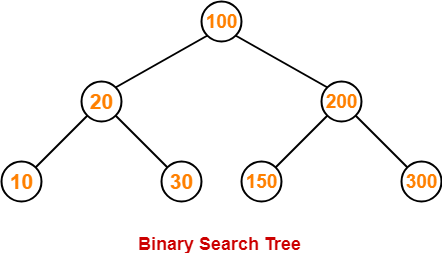
* Only larger values in its right subtree.
* Only smaller values in its left subtree.

## ****BST Traversal-****

* A binary search tree is traversed in exactly the same way a binary tree is traversed.
* In other words, BST traversal is same as binary tree traversal.

## ****Example-****

Consider the following binary search tree-



Now, let us write the traversal sequences for this binary search tree-

### ****Preorder Traversal-****

100 , 20 , 10 , 30 , 200 , 150 , 300

### ****Inorder Traversal-****

10 , 20 , 30 , 100 , 150 , 200 , 300

### ****Postorder Traversal-****

10 , 30 , 20 , 150 , 300 , 200 , 100

## ****Important Notes-****

## ****Note-01:****

* Inorder traversal of a binary search tree always yields all the nodes in increasing order.

## ****Note-02:****

Unlike [**Binary Trees**](https://www.gatevidyalay.com/binary-tree-types-of-trees-in-data-structure/),

* A binary search tree can be constructed using only preorder or only postorder traversal result.
* This is because inorder traversal can be obtained by sorting the given result in increasing order.

## ****PRACTICE PROBLEMS BASED ON BST TRAVERSAL-****

## ****Problem-01:****

Suppose the numbers 7 , 5 , 1 , 8 , 3 , 6 , 0 , 9 , 4 , 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers.

What is the inorder traversal sequence of the resultant tree?

1. 7 , 5 , 1 , 0 , 3 , 2 , 4 , 6 , 8 , 9
2. 0 , 2 , 4 , 3 , 1 , 6 , 5 , 9 , 8 , 7
3. 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9
4. 9 , 8 , 6 , 4 , 2 , 3 , 0 , 1 , 5 , 7

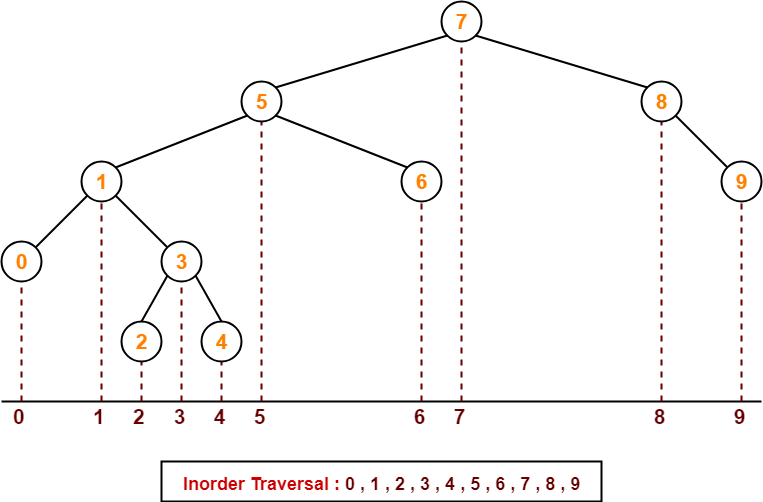
## ****Solution-****

This given problem may be solved in the following two ways-

### ****Method-01:****

* We construct a binary search tree for the given elements.
* We write the inorder traversal sequence from the binary search tree so obtained.

Following these steps, we have-



Thus, Option (C) is correct.

### ****Method-02:****

We know, inorder traversal of a binary search tree always yields all the nodes in increasing order.

Using this result,

* We arrange all the given elements in increasing order.
* Then, we report the sequence so obtained as inorder traversal sequence.

|  |
| --- |
| **Inorder Traversal :**  **0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9** |

Thus, Option (C) is correct.

## ****Problem-02:****

The preorder traversal sequence of a binary search tree is-

30 , 20 , 10 , 15 , 25 , 23 , 39 , 35 , 42

What one of the following is the postorder traversal sequence of the same tree?

1. 10 , 20 , 15 , 23 , 25 , 35 , 42 , 39 , 30
2. 15 , 10 , 25 , 23 , 20 , 42 , 35 , 39 , 30
3. 15 , 20 , 10 , 23 , 25 , 42 , 35 , 39 , 30
4. 15 , 10 , 23 , 25 , 20 , 35 , 42 , 39 , 30

## ****Solution-****

In this question,

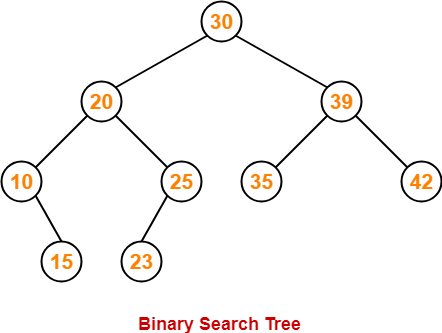
* We are provided with the preorder traversal sequence.
* We write the inorder traversal sequence by arranging all the numbers in ascending order.

Then-

* Postorder Traversal : 30 , 20 , 10 , 15 , 25 , 23 , 39 , 35 , 42
* Inorder Traversal : 10 , 15 , 20 , 23 , 25 , 30 , 35 , 39 , 42

Now,

* We draw a binary search tree using these traversal results.
* The binary search tree so obtained is as shown-



Now, we write the postorder traversal sequence-

|  |
| --- |
| **Postorder Traversal :**  **15 , 10 , 23 , 25, 20, 35, 42, 39, 30** |

Thus, Option (D) is correct.

# [Binary Search Tree Traversal | BST Traversal](https://www.gatevidyalay.com/binary-search-tree-traversal-bst-traversal/)

## ****Binary Search Tree-****

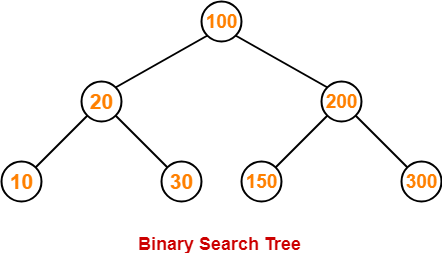
Binary search tree (BST) is a special kind of binary tree where each node contains-

* Only larger values in its right subtree.
* Only smaller values in its left subtree.

## ****BST Traversal-****

* A binary search tree is traversed in exactly the same way a binary tree is traversed.
* In other words, BST traversal is same as binary tree traversal.

Consider the following binary search tree-



Now, let us write the traversal sequences for this binary search tree-

### ****Preorder Traversal-****

100 , 20 , 10 , 30 , 200 , 150 , 300

**Inorder Traversal-**

  10 , 20 , 30 , 100 , 150 , 200 , 300

**Postorder Traversal-**

  10 , 30 , 20 , 150 , 300 , 200 , 100

**Important Notes-**

**Note-01:**

 Inorder traversal of a binary search tree always yields all the nodes in increasing order.

## ****Note-02:****

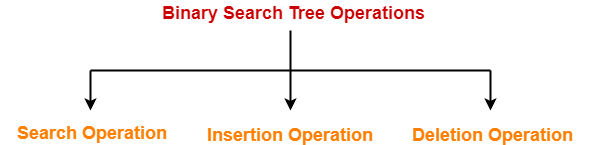
Unlike [**Binary Trees**](https://www.gatevidyalay.com/binary-tree-types-of-trees-in-data-structure/),

* A binary search tree can be constructed using only preorder or only postorder traversal result.
* This is because inorder traversal can be obtained by sorting the given result in increasing order.

# [Time Complexity of Binary Search Tree](https://www.gatevidyalay.com/time-complexity-of-bst-binary-search-tree/)

## ****Binary Search Tree-****

Commonly performed operations on binary search tree are-



1. Search Operation
2. Insertion Operation
3. Deletion Operation

## ****Time Complexity-****

* Time complexity of all BST Operations = O(h).
* Here, h = Height of binary search tree

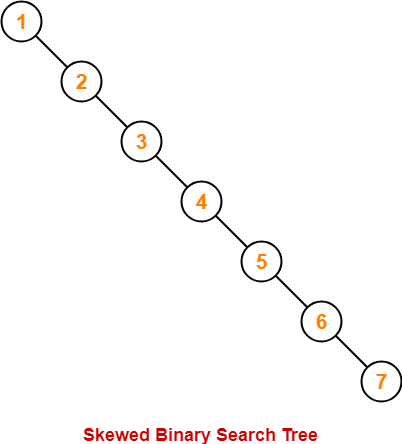
Now, let us discuss the worst case and best case.

### ****Worst Case-****

In worst case,

* The binary search tree is a skewed binary search tree.
* Height of the binary search tree becomes n.
* So, Time complexity of BST Operations = O(n).

In this case, binary search tree is as good as unordered list with no benefits.



### ****Best Case-****

In best case,

* The binary search tree is a balanced binary search tree.
* Height of the binary search tree becomes log(n).
* So, Time complexity of BST Operations = O(logn).

