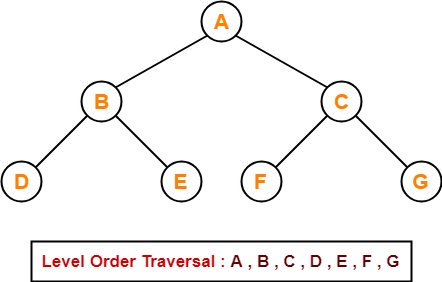
## ****Breadth First Traversal-****

* 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-****



In order🡪 Left -🡪 ROOT 🡪RIGHT

Pre order 🡪 ROOT🡪 Left -🡪 RIGHT

Post order🡪 Left 🡪 RIGHT🡪 ROOT

 .

A🡪B🡪C🡪D🡪E🡪F🡪G

## ****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.

# [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.

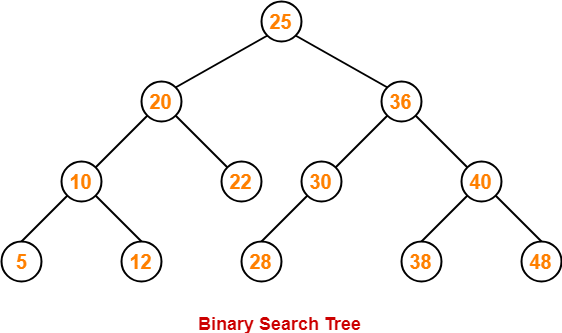
## ****Binary Search Tree-****

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

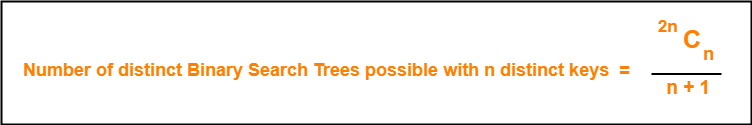
In a binary search tree (BST), each node contains-

* 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

= FAC 6/((FAC 3 \* FAC(6-3))/4

=6 x 5 x 4 x fac 3/ (fac 3 x fac 3)/4

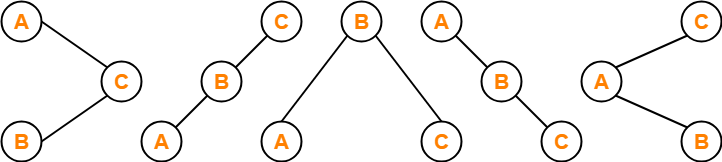
=6 x 5 x 4 / fac ¾

= 6 x 5 x 4/ 3 x 2 x ¼

= 5

= 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-

50

20 70

### **10 40 60 90**

### **100**

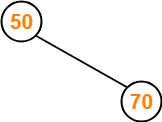
### 

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

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

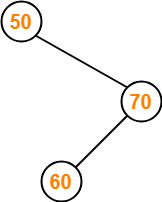
### ****Insert 70-****

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



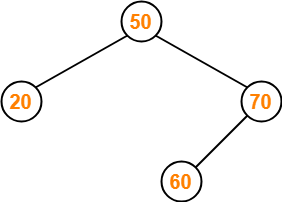
### ****Insert 60-****

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



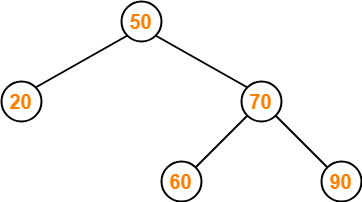
### ****Insert 20-****

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



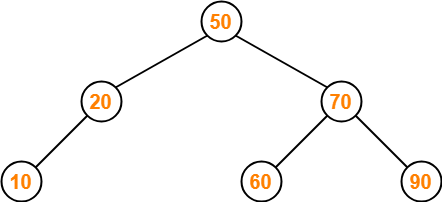
### ****Insert 90-****

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



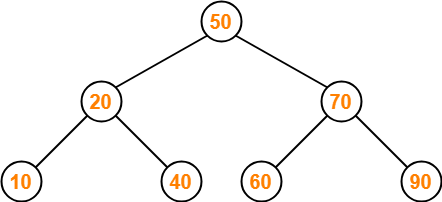
### ****Insert 10-****

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



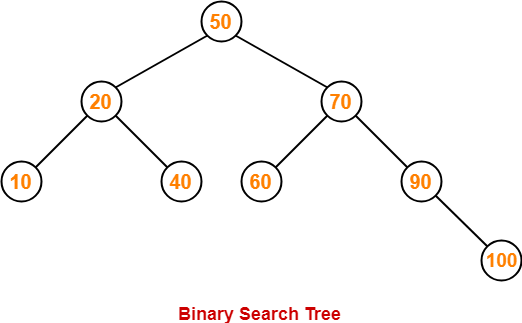
### ****Insert 40-****

* 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.



50

20 70

### **10 40 60 90**

### **100**

### 

## ****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

50

15 62

5 20 58 91

3 8 37 60

24

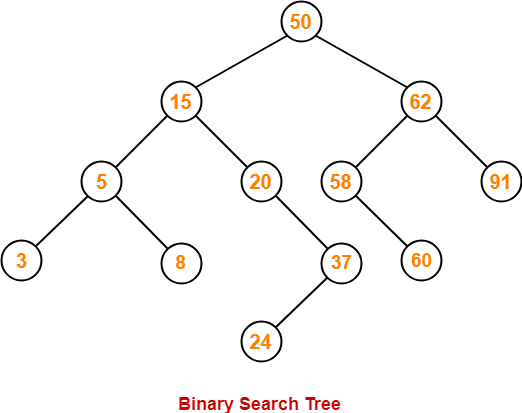
The number of nodes in the left subtree and right subtree of the root respectively is \_\_\_\_\_.

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-



50

15 62

5 20 58 91

3 8 37 60

24

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.

## ****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-****

 2n C n /n+1

= 8 C 4 /5

= fac 8 / (fac 4 x fac(8-4))/5

= 8 x 7 x 6 x5 x fac 4 / (fac 4 x fac4))/5

= 8 x 7 x 6 x5 / (fac4))/5

= 2 x 7

=14

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

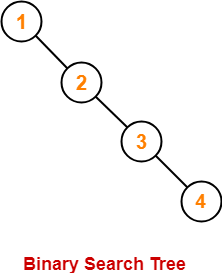
## ****Solution-****

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-****

.

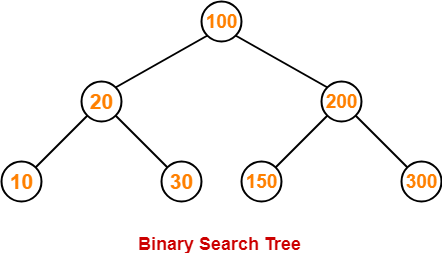
Binary search tree (BST) is a special kind of binary tree where each node contains-

* 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.

Consider the following binary search tree-



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

Inorder: LEFT🡪ROOT🡪RIGHT

10🡪20🡪30🡪100🡪150🡪200🡪300

Preorder: ROOT🡪 LEFT🡪 RIGHT

100🡪20🡪10🡪30🡪200🡪150🡪300

Postorder: LEFT 🡪RIGHT🡪ROOT

10🡪30🡪20🡪150🡪300🡪200🡪100

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

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

Preorder: ROOT🡪 LEFT🡪 RIGHT

100🡪20🡪10🡪30🡪200🡪150🡪300

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

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

Inorder: LEFT🡪ROOT🡪RIGHT

10🡪20🡪30🡪100🡪150🡪200🡪300

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

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

Postorder: LEFT 🡪RIGHT🡪ROOT

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

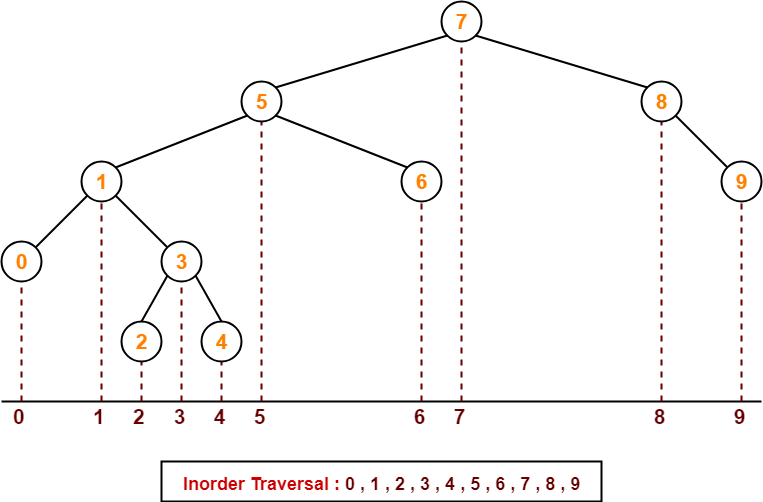
## ****Solution-****

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

Preorder: Root🡪Left🡪Right

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,

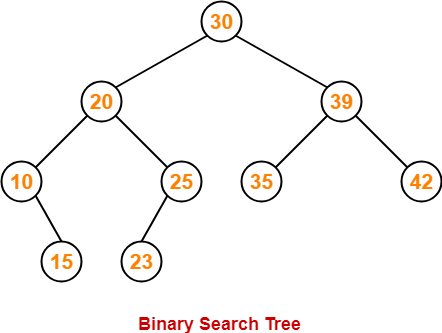
* We are provided with the preorder traversal sequence.
* We write the inorder traversal sequence by arranging all the numbers in ascending order.

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 Insertion | BST Deletion](https://www.gatevidyalay.com/binary-search-tree-insertion-bst-deletion/)

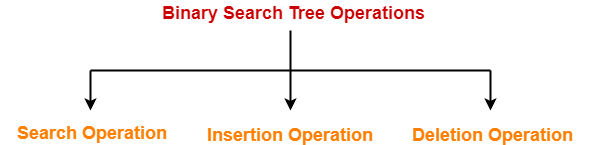
## ****Binary Search Tree-****

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.

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

Commonly performed operations on binary search tree are-



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

## ****1. Search Operation-****

|  |
| --- |
| Search Operation is performed to search a particular element in the Binary Search Tree. |

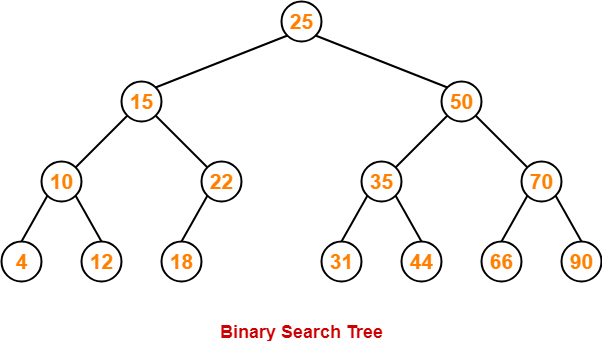
## ****Rules-****

For searching a given key in the BST,

* Compare the key with the value of root node.
* If the key is present at the root node, then return the root node.
* If the key is greater than the root node value, then recur for the root node’s right subtree.
* If the key is smaller than the root node value, then recur for the root node’s left subtree.

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

Consider key = 45 has to be searched in the given BST-



* We start our search from the root node 25.
* As 45 > 25, so we search in 25’s right subtree.
* As 45 < 50, so we search in 50’s left subtree.
* As 45 > 35, so we search in 35’s right subtree.
* As 45 > 44, so we search in 44’s right subtree but 44 has no subtrees.
* So, we conclude that 45 is not present in the above BST.

## ****2. Insertion Operation-****

|  |
| --- |
| Insertion Operation is performed to insert an element in the Binary Search Tree. |

## ****Rules-****

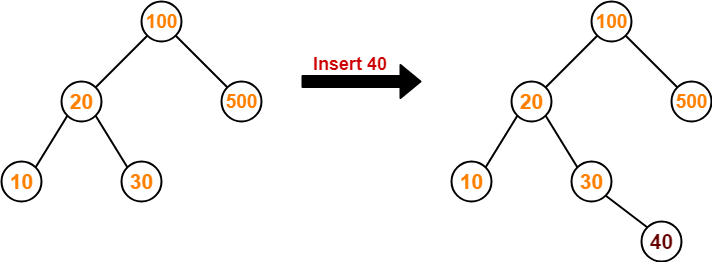
The insertion of a new key always takes place as the child of some leaf node.

For finding out the suitable leaf node,

* Search the key to be inserted from the root node till some leaf node is reached.
* Once a leaf node is reached, insert the key as child of that leaf node.

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

Consider the following example where key = 40 is inserted in the given BST-



* We start searching for value 40 from the root node 100.
* As 40 < 100, so we search in 100’s left subtree.
* As 40 > 20, so we search in 20’s right subtree.
* As 40 > 30, so we add 40 to 30’s right subtree.

## ****3. Deletion Operation-****

|  |
| --- |
| Deletion Operation is performed to delete a particular element from the Binary Search Tree. |

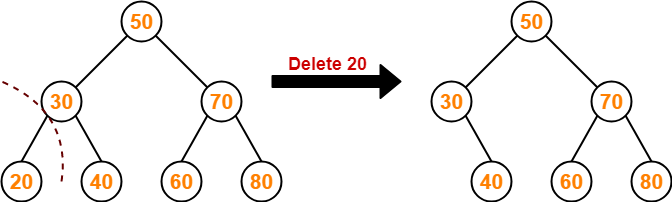
When it comes to deleting a node from the binary search tree, following three cases are possible-

### ****Case-01: Deletion Of A Node Having No Child (Leaf Node)-****

Just remove / disconnect the leaf node that is to deleted from the tree.

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

Consider the following example where node with value = 20 is deleted from the BST-

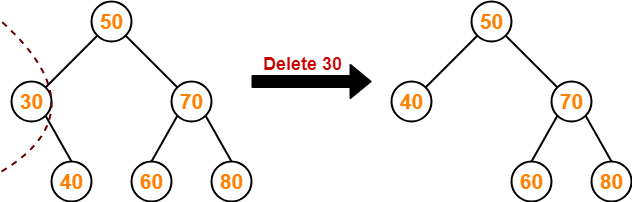


### ****Case-02: Deletion Of A Node Having Only One Child-****

Just make the child of the deleting node, the child of its grandparent.

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

Consider the following example where node with value = 30 is deleted from the BST-



### ****Case-02: Deletion Of A Node Having Two Children-****

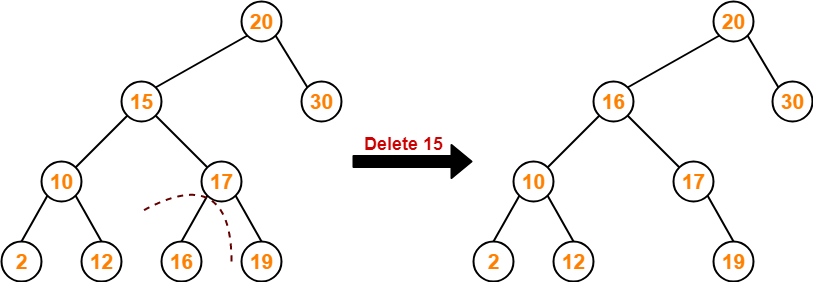
A node with two children may be deleted from the BST in the following two ways-

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

* Visit to the right subtree of the deleting node.
* Pluck the least value element called as inorder successor.
* Replace the deleting element with its inorder successor.

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

Consider the following example where node with value = 15 is deleted from the BST-

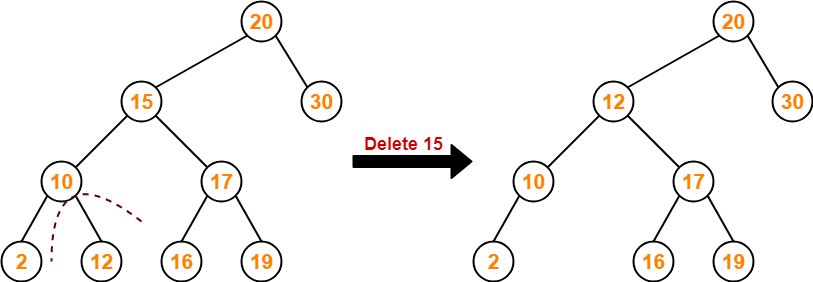


### ****Method-02:****

* Visit to the left subtree of the deleting node.
* Pluck the greatest value element called as inorder predecessor.
* Replace the deleting element with its inorder predecessor.

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

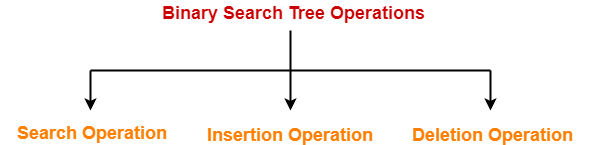
Consider the following example where node with value = 15 is deleted from the BST-



# [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

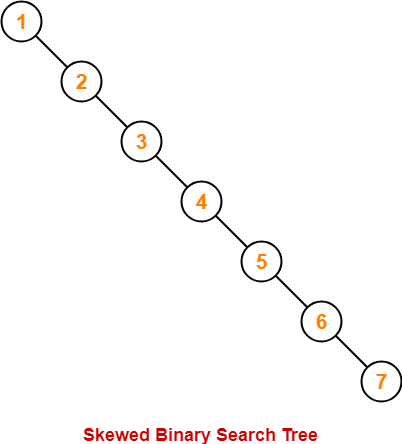
Now, let us discuss the worst case and best case.

### ****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).

