**Binary Search Tree (Revision)**

Binary search tree is a data structure that quickly allows us to maintain a sorted list of numbers. It is called a binary tree because each tree node has a maximum of two children. It is called a search tree because it can be used to search for the presence of a number in O(log(n)) time.

The properties that separate a binary search tree from a regular binary tree is:

1. All nodes of left subtree are less than the root node
2. All nodes of right subtree are more than the root node
3. Both subtrees of each node are also BSTs i.e. they have the above two properties

**Search Operation**

The algorithm depends on the property of BST that if each left subtree has values below root and each right subtree has values above the root.

If the value is below the root, then the value is not in the right subtree; we need to only search in the left subtree and if the value is above the root, then the value is not in the left subtree; we need to only search in the right subtree.

**Algorithm**

**Text

Description automatically generated**

**Insert Operation**

Inserting a value in the correct position is similar to searching because we try to maintain the rule that the left subtree is lesser than root and the right subtree is larger than root. We keep going to either right subtree or left subtree depending on the value and when we reach a point left or right subtree is null, we put the new node there.

**Algorithm**

Text

Description automatically generated

**Deletion Operation**

There are three cases for deleting a node from a binary search tree.

**Case I**

In the first case, the node to be deleted is the leaf node. In such a case, simply delete the node from the tree.

**Case II**

In the second case, the node to be deleted lies has a single child node. In such a case follow the steps below:

* Replace that node with its child node.
* Remove the child node from its original position.

**Case III**

In the third case, the node to be deleted has two children. In such a case follow the steps below:

1. Get the inorder successor of that node.
2. Replace the node with the inorder successor.
3. Remove the inorder successor from its original position.

**Binary Search Tree Complexities**

**Time Complexity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Best Case Complexity** | **Average Case Complexity** | **Worst Case Complexity** |
| **Search** | O(log n) | O(log n) | O(n) |
| **Insertion** | O(log n) | O(log n) | O(n) |
| **Deletion** | O(log n) | O(log n) | O(n) |

**Binary Search Tree Applications**

* In multilevel indexing in the database
* For dynamic sorting
* For managing virtual memory areas in Unix kernel

**Exam Questions**

1. Icon

   Description automatically generated**(a) Draw the binary search tree that is created if the following numbers are inserted in the tree in the given order. 12 15 3 35 21 42 14**

**A picture containing icon

Description automatically generated(b)** **Draw a balanced binary search tree containing the same numbers given in part (a).**

1. **TRUE/FALSE: State “True” or “False” and explain briefly.**
2. **An abstract class can implement an interface.**

True. There’s nothing about to forbid it.

1. **Searching for an element in a binary search tree with N nodes takes O(log N).**

False. If the tree isn’t roughly balanced (or non-bushy, or not like a linked list), then it could be O(N).

1. **If a method M of class A throws an exception and does not catch it, the exception will be caught by the parent class of A.**

False. It might be caught by the calling method, but that doesn’t have to

be in the parent class.

1. **If we have two algorithms A1 and A2, and A1 takes time O(N) while A2 is O(N3), then A1 always runs faster than A2, for any input.**

False. For very large N, A1 will be faster than A2, but perhaps not for smaller N.

1. **How many non-null links are there in a doubly linked list with N nodes?**

2N-2. Each node has two outgoing links, but the last ones in each direction are null.

1. **How many non-null links are there in a binary tree with N nodes?**

N-1. There’s one incoming to each node except the root.

1. **The integers 7, 1, 12, 8, 3, 0, –1, 9 are inserted in that order into an initially empty binary search tree. Draw the tree after the last insertion.**

**Diagram

Description automatically generated with medium confidence**