Binary Search Trees

Sergios Anestis Kefalidis
Konstantinos Nikoletos
Kostas Plas
& Manolis Koubarakis

What’s a Binary Search Tree (BST)?

A binary search tree is a tree-like data structure that is made up of nodes. Each node has a value, and at most two child nodes, known as the left and right child.

Properties:

1. The *left child* of a node has a value **less** than the node's value, while the *right child* has a value **greater** than or equal to the node's value.
2. The binary search tree is **sorted**, making it easy to search for a specific value within it.

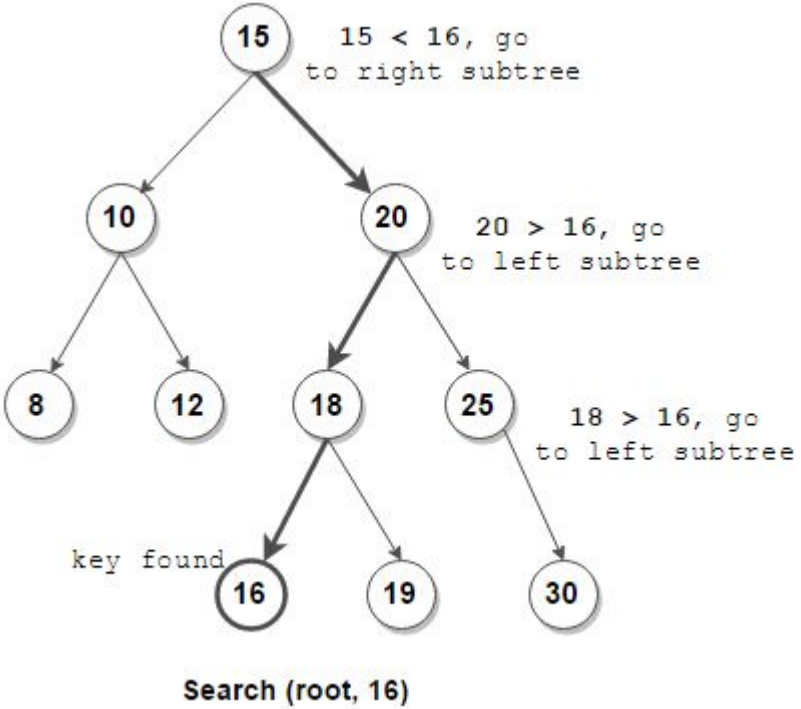
Search in a BST

Searching for a value involves starting at the root node, comparing the value with the node's value, and moving left or right depending on whether the value is greater or less than the current node's value.

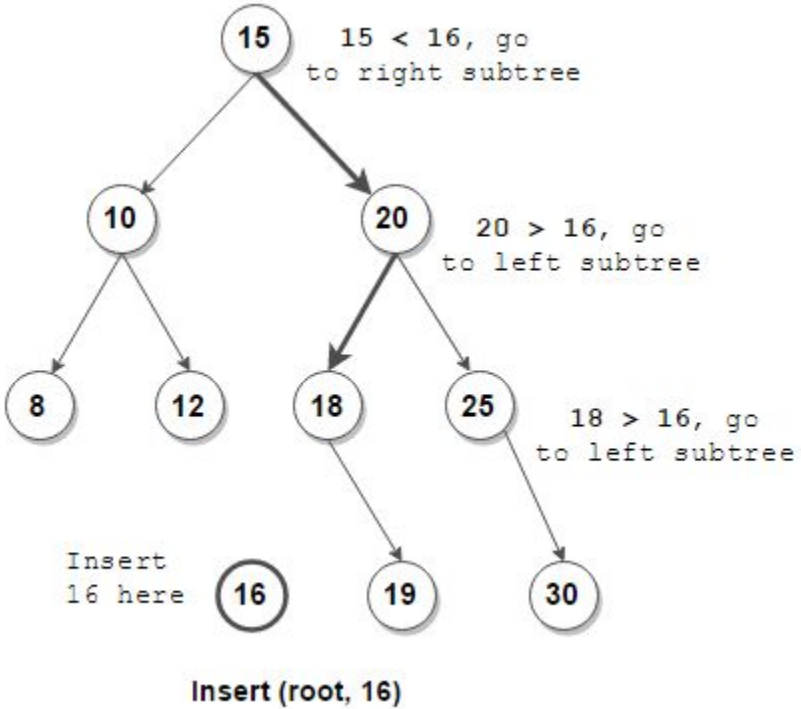
[Go to bst.c and fill in the code]

Time complexity in in the

- average case: **O(logn)**, why?
- worst case: **O(n)**



Insertion in a BST



Insertion in a binary search tree involves adding a new node to the tree in a position that maintains the BST properties

Time complexity in in the

- average case: **O(logn)**, why?
- worst case: **O(n)**

Can you describe how the tree would look like? How can we avoid the worst case complexity?

Delete from a BST

Deleting a node from a binary search tree involves removing the node while still maintaining the BST's properties. How?? (check next slide)

[Go to bst.c and fill in the code]

Deleting a node from a binary search tree involves **two** main cases:

1. The node being deleted has no children (it is a **leaf node**). In this case, simply remove the node from the tree.
2. The node being deleted has **one** or **two children**.

Time complexity in in the

- average case: **O(logn)**
- worst case: **O(n)**

Preserve BST properties with rotations

Rotations in a BST are a set of operations used to balance the tree and maintain its properties (check slide 2 for these properties). There are two types of rotations:

- **Left rotation:** This operation involves moving the right child of a node up to its position and moving the original node down to the left of its former right child.
- **Right rotation:** This operation involves moving the left child of a node up to its position and moving the original node down to the right of its former left child.

How to traverse a BST?

Three main methods of traversing a BST:

1. **Inorder traversal:** In this method, we first visit the left child of a node, then the node itself, and then the right child. This traversal method visits the nodes of the tree in ascending order. [Go to bst.c and fill in the code]
2. **Preorder traversal:** In this method, we first visit the node itself, then the left child, and then the right child. This traversal method is useful for creating a copy of a tree or a prefix expression. [Go to bst.c and fill in the code]
3. **Postorder traversal:** In this method, we first visit the left child, then the right child, and then the node itself. This traversal method is useful for deleting a tree or a postfix expression. [Go to bst.c and fill in the code]