Lab#12

Implementation of BST

Objective

After completing this lab, the students should be able to:

- Implement the binary search tree.
- Implement the traversing schemes in binary search tree-BST (in-order, pre-order and post order)

Theory

Binary Tree

Binary Tree is a special data structure used for data storage purposes. A binary tree has a special condition that each node can have a maximum of two children. A binary tree has the benefits of both an ordered array and a linked list as search is as quick as in a sorted array and insertion or deletion operation are as fast as in linked list.

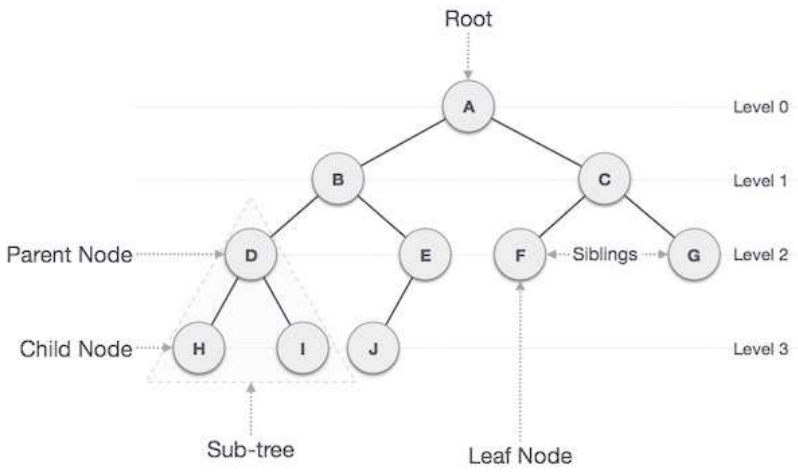


Figure 1 An example of BT

Binary Search Tree

BST is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys lesser than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- The left and right subtree each must also be a binary search tree.

Happy Coding 😊

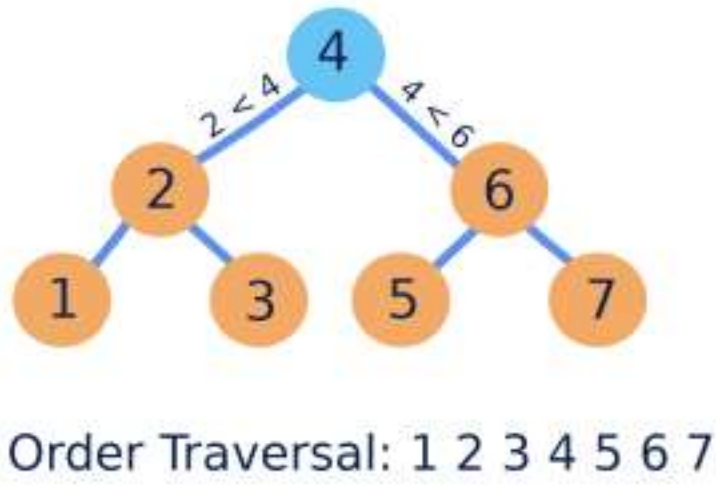


Figure 2 An example of BST

Difference between Binary Tree and Binary Search Tree:

BINARY TREE	**BINARY SEARCH TREE**
BINARY TREE is a nonlinear data structure where each node can have almost two child nodes	BINARY SEARCH TREE is a node based binary tree which further has right and left subtree that too are binary search trees.
BINARY TREE is unordered hence slower in process of insertion, deletion and searching.	Insertion, deletion, searching of an element is faster in BINARY SEARCH TREE than BINARY TREE due to the ordered characteristics
IN BINARY TREE there is no ordering in terms of how the nodes are arranged	IN BINARY SEARCH TREE the left subtree has elements less than the nodes element and the right subtree has elements greater than the nodes element.

Basic Operations

1) Insertion/ add node.
2) Traversing
    a. Inorder
    b. Preorder
    c. Postorder

Happy Coding 😊

Preorder Traversing

Algorithm

To traverse a nonempty binary tree:

1. Visit the root.
2. If the left subtree is nonempty, do a preorder traversal on it.
3. If the right subtree is nonempty, do a preorder traversal on i

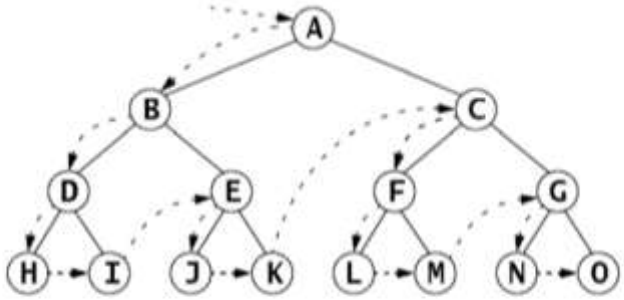


Figure 11.15 The preorder traversal of a binary tree

The nodes are visited in the order **A, B, D, H, I, E, J, K, C, F, L, M, G, N, O.**

Postorder Traversing

Algorithm

To traverse a nonempty binary tree:

1. If the left subtree is nonempty, do a postorder traversal on it.
2. If the right subtree is nonempty, do a postorder traversal on it.
3. Visit the root.

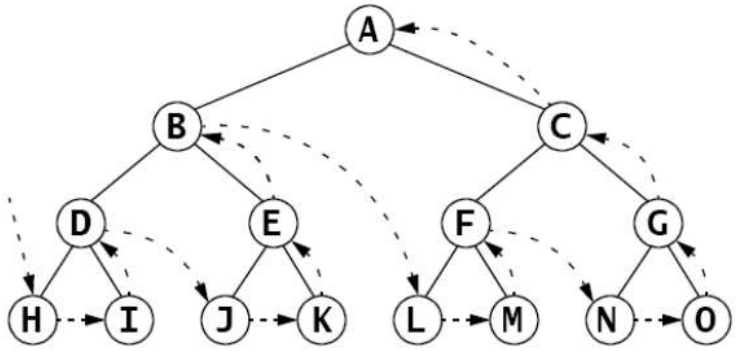


Figure 11.17 The postorder traversal of a binary tree

The nodes are visited in the order **H, I, D, J, K, E, B, L, M, F, N, O, G, C, A.**

Inorder Traversing

Algorithm

To traverse a nonempty binary tree:

1. If the left subtree is nonempty, do a preorder traversal on it.
2. Visit the root.
3. If the right subtree is nonempty, do a preorder traversal on it.

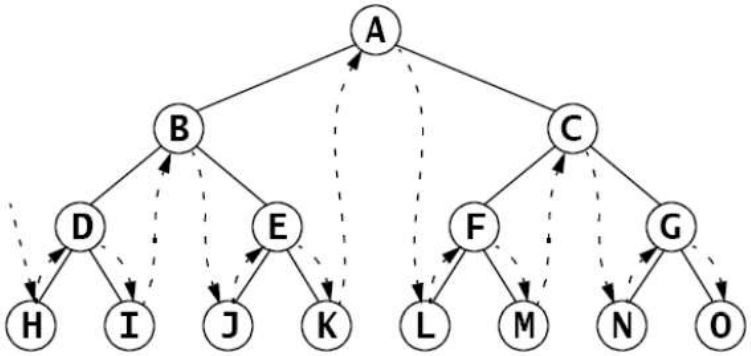


Figure 11.18 The inorder traversal of a binary tree

The nodes are visited in the order **H, D, I, B, J, E, K, A, L, F, M, C, N, G, O.**

Code Example of BST

GitHub:

Let's us visualize it: https://visualgo.net/en/bst?slide=1

Happy Coding 😊

Exercise

Task # 1: Create a Binary Search Tree for the following data.

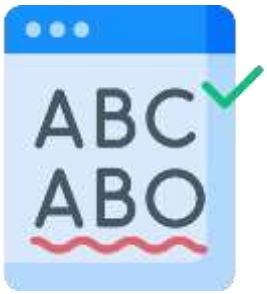
6,8,22,3,7,5,12,10,9,20,35,40,42

Task # 2: Create a Binary Search Tree for the following data.

A B C D E F G H I J K L M N O P

[Optional]Bonus Question for Champs 😊

Implement spell checker using BST.



Happy Coding 😊