Traversing a Binary Search Tree

inorder, preorder and postorder traversal (Reading time: 2 minutes)

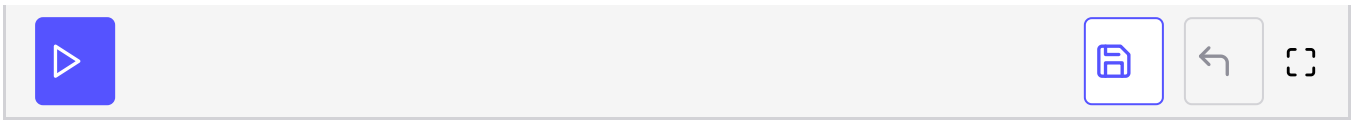
There are three ways to traverse (go through) a binary search tree:

Inorder

- Left subtree
- Root node
- Right subtree

The inorder way is important if you want to flatten the tree back into its original sequence.

```
inOrder(node) {
   if (node) {
      this.inOrder(node.left);
      console.log(node.data);
      this.inOrder(node.right);
   }
}
```



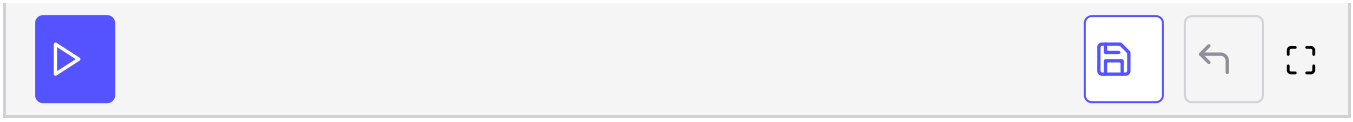
Preorder

- Root node
- Left subtree
- Right subtree

The preorder way is important if you need to inspect roots before inspecting the leaves.

```
preOrder(node) {
   if (node) {
      console.log(node.data);
      this.preOrder(node.left);
```

```
      this.preOrder(node.right);
   }
}
```

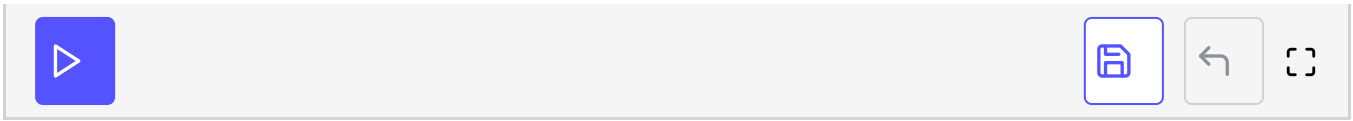


Postorder

- Left subtree
- Right subtree
- Root node

The postorder way is important if you want to delete an entire tree, or simply want to inspect the leaves before inspecting the nodes. If you deleted the root node, you wouldn't be able to delete the nodes in the right subtree!

```
postOrder(node) {
   if (node) {
      this.postOrder(node.left);
      this.postOrder(node.right);
      console.log(node.data);
   }
}
```



Now, lets move on to the breadt-first and depth-first traversals of a BST.