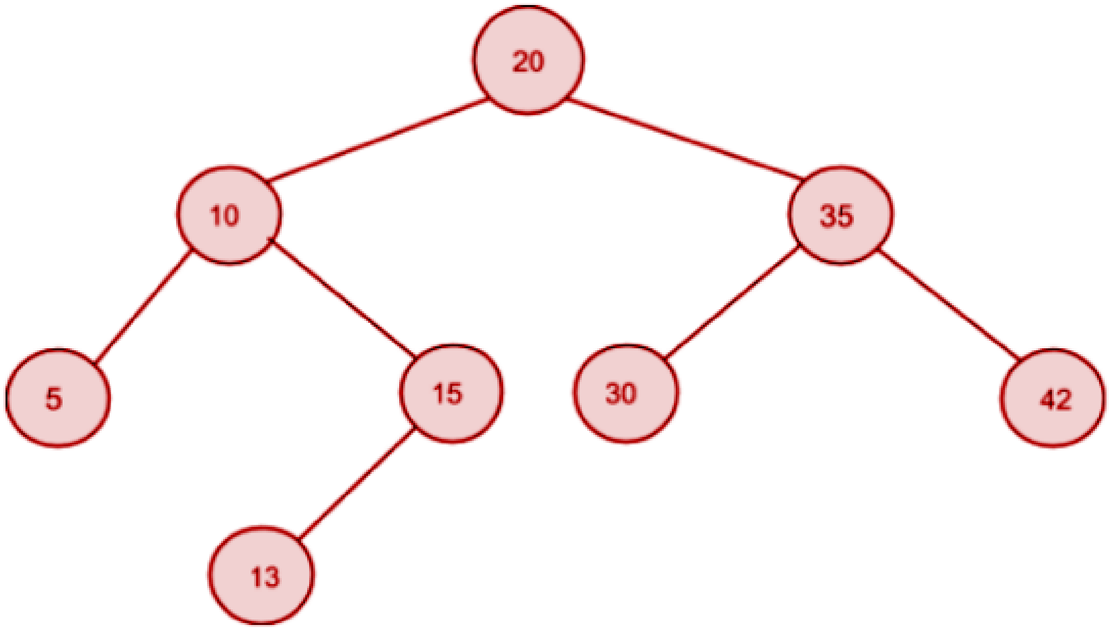
Construction

Construction from preorder traversal

Problem Statement: You are given the pre-order traversal of the array and you need to construct the original BST from it.

Example : For the array : [20, 10, 5, 15, 13, 35, 30, 42], the unique BST is :



The idea is based on the fact that the value of each node in a BST is greater than the value of all nodes in its left subtree and less than the value of all nodes in its right subtree.
So, instead of explicitly searching for indices that split the left and the right subtree (as in the previous approach), we will pass the information about the valid range of values for a node and its children in recursion itself.

- We will maintain 'minVal' and 'maxVal' (initialized to a minimum and maximum possible value, respectively) to set a range for every node.
- We will also maintain 'preorderIndex' to keep track of the index in the PREORDER array.
- We initialize 'currVal' to PREORDER[preorderIndex].

    - If ‘currVal’ doesn’t fall in the range [minVal: maxVal], we return NULL.
    - Else, we construct ‘node’ initialized to ‘currVal’ and increment ‘preorderIndex’.
- We will then set the range for the left subtree ([minVal: currVal-1]) and the right subtree ([currVal+1: maxVal]) and recursively construct them.

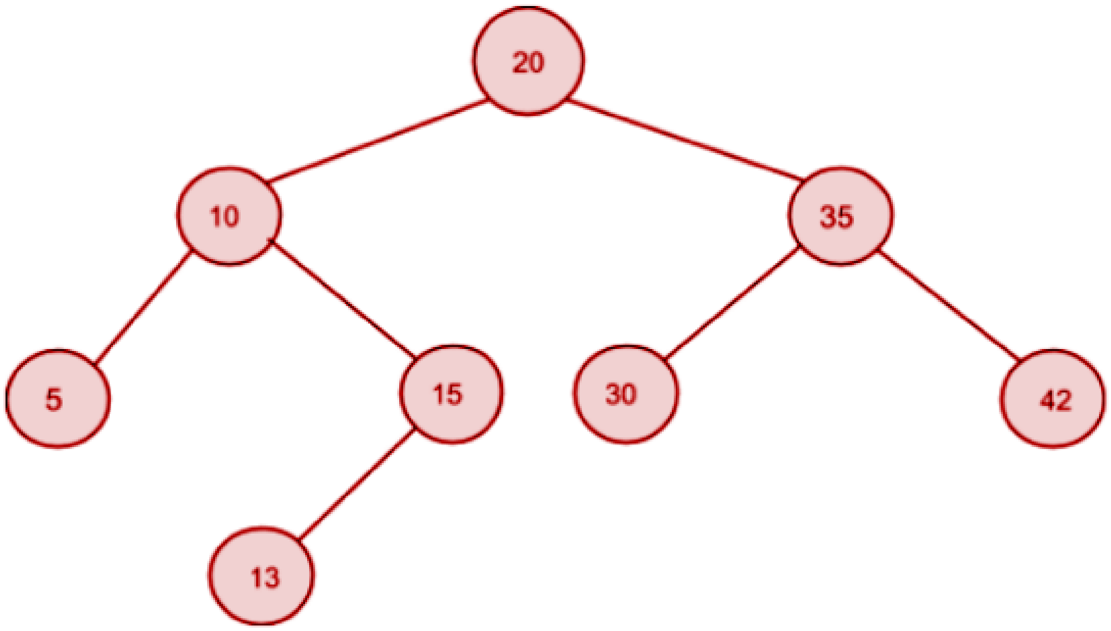
```
function preorderToBST(preorder, preorderIndex, minVal, maxVal)

        //  base case to reach end of the array
        if preorderIndex == len(preorder)
                return Null

        //  current value of node
        currVal = preorder[preorderIndex]

        //  value outside the present range
        if currVal < minVal or currVal > maxVal
                return Null


        root = TreeNode(currVal)

        preorderIndex += 1


        //  Recursively construct left subtree
        root.left = preorderToBST(preorder, preorderIndex, minVal, currVal - 1)
        //  Recursively construct right subtree
        root.right = preorderToBST(preorder, preorderIndex, currVal + 1, maxVal)

        return root
```

Time Complexity: O(N), where N is the number of nodes in the BST.
We are traversing through all the nodes of the tree in a preorder fashion.
Space Complexity: O(N), where N is the number of nodes in the BST.
O(H) recursion stack space is used by the algorithm. In the worst case, H will become N (in the case of skew trees). Thus, space complexity is O(N).

Construction from postorder traversal

Problem Statement: You are given the post-order traversal of the array and you need to construct the original BST from it.

Example : For the array : [5, 13, 15, 10, 30, 42, 35, 20], the unique BST is :



The idea is similar to the last time, with the only difference being that here we maintain a postorderIndex that starts from the end of the array and moves towards the beginning when we call the recursive function for the right subtree and then to the left subtree.

```
function postorderToBST(postorder, postorderIndex, minVal, maxVal)

        //  base case to reach beginning of the array
        if postorderIndex == -1
                return Null

        //  current value of node
        currVal = postorder[postorderIndex]

        //  value outside the postsent range
        if currVal < minVal or currVal > maxVal
                return Null


        root = TreeNode(currVal)

        postorderIndex -= 1
```

```
//   Recursively construct right subtree
root.right =postorderToBST(postorder,postorderIndex, currVal+1, maxVal)

//   Recursively construct left subtree
root.left = postorderToBST(postorder, postorderIndex, minVal, currVal - 1)

return root
```

Time Complexity: O(N), where N is the number of nodes in the BST.
We are traversing through all the nodes of the tree in a preorder fashion.
Space Complexity: O(N), where N is the number of nodes in the BST.
O(H) recursion stack space is used by the algorithm. In the worst case, H will become N (in the case of skew trees). Thus, space complexity is O(N).