Certainly! To construct a Binary Search Tree (BST) from a given pre-order traversal, you can follow the steps outlined below: 1. Define the `Node` class, representing a single node in the BST, similar to the previous code example. 2. Create a function, let's call it `constructBST`, that takes the pre-order traversal array, its start and end indices, and returns the root of the constructed BST. 3. In the `constructBST` function, if the start index is greater than the end index, return `nullptr`. 4. Create a new `Node` with the value at the start index of the pre-order traversal array. This node will be the root of the current subtree. 5. Find the index `i` where the first element greater than the root's value occurs in the pre-order traversal array (between start and end indices). 6. Recursively call `constructBST` for the left subtree, using the range from start+1 to i-1, and assign the returned node as the left child of the current root. 7. Recursively call `constructBST` for the right subtree, using the range from i to end, and assign the returned node as the right child of the current root. 8. Return the current root node. 9. Finally, call the `constructBST` function with the pre-order traversal array and appropriate start and end indices to construct the BST. Here's an implementation of the above steps in C++: ```cpp #include #include using namespace std; // Node class class Node { public: int data; Node* left; Node* right; Node(int value) { data = value; left = nullptr; right = nullptr; } }; // Function to construct BST from pre-order traversal Node* constructBST(vector& preorder, int start, int end) { if (start > end) { return nullptr; } Node* root = new Node(preorder[start]); // Find the index where the first element greater than the root's value occurs int i; for (i = start; i <= end; i++) { if (preorder[i] > root->data) { break; } } root->left = constructBST(preorder, start + 1, i - 1); root->right = constructBST(preorder, i, end); return root; } // Function to perform in-order traversal of the BST void inorderTraversal(Node* root) { if (root != nullptr) { inorderTraversal(root->left); cout << root->data << " "; inorderTraversal(root->right); } } // Example usage int main() { vector preorder = {10, 6, 3, 8, 15, 20}; Node* root = constructBST(preorder, 0, preorder.size() - 1); cout << "In-order traversal of the constructed BST: "; inorderTraversal(root); cout << endl; return 0; } ``` In this example, we have the pre-order traversal `10, 6, 3, 8, 15, 20`. The `constructBST` function is called with this traversal array and the start and end indices (`0` and `preorder.size() - 1`, respectively). It constructs the corresponding BST and returns the root. Finally, the `inorderTraversal` function is called to perform an in-order traversal of the constructed BST and display the result.