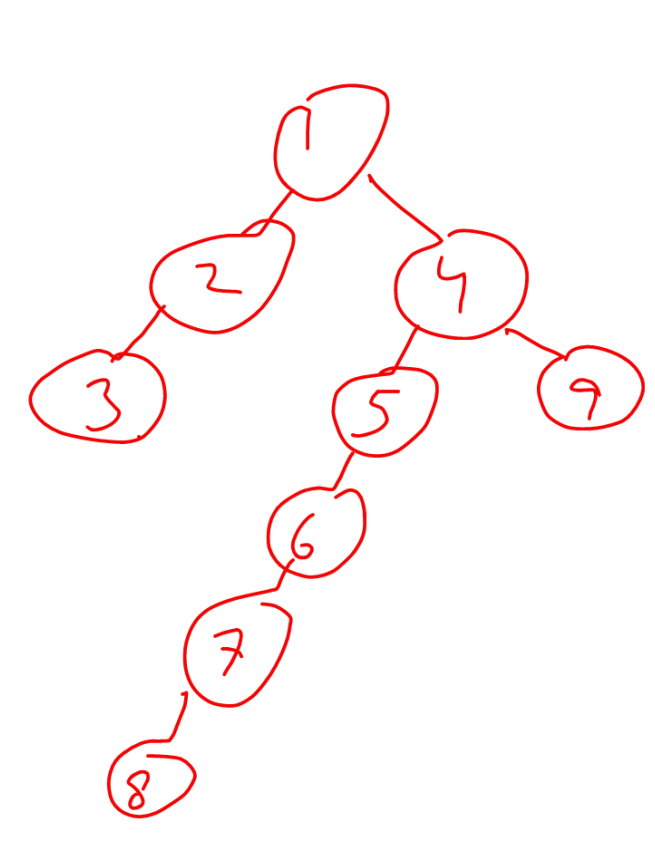
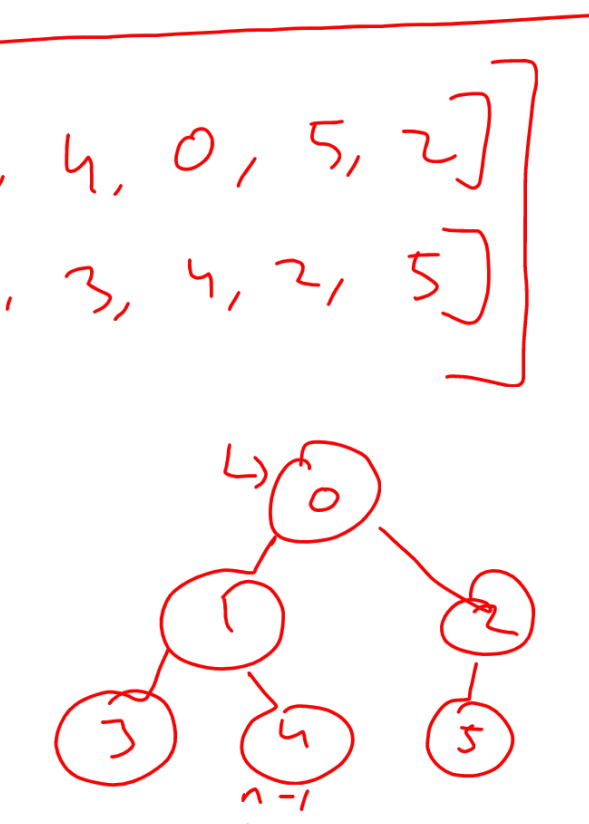


lh=2 rh=5
2+5+1=8
diam=
(2, 8, 6)=8

---

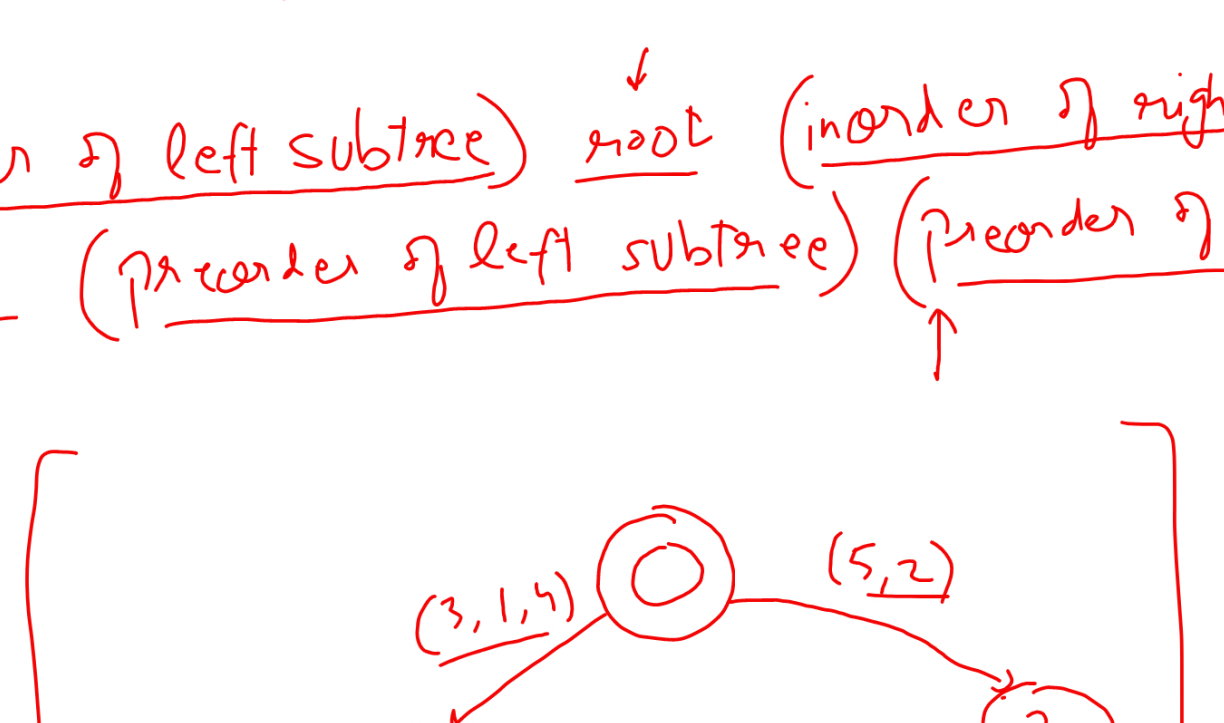
in [] = [3, 1, 4, 0, 5, 2]
pre [] = [0, 1, 3, 4, 2, 5]

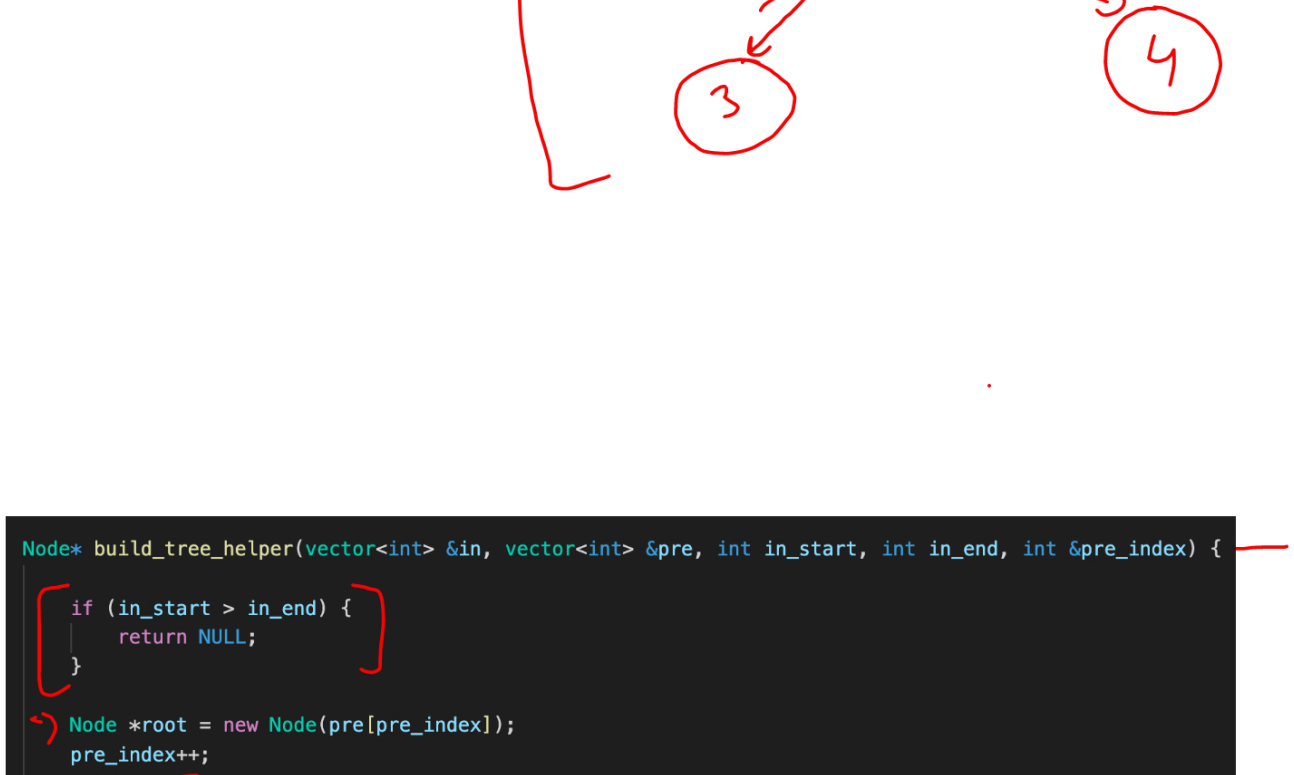


[4]

in: [3, 1, 4, 0, 5, 2]
pre: [0, 1, 3, 4, 2, 5]

[in : (inorder of left subtree) root (inorder of right subtree)]
[pre : root (preorder of left subtree) (preorder of right subtree)]

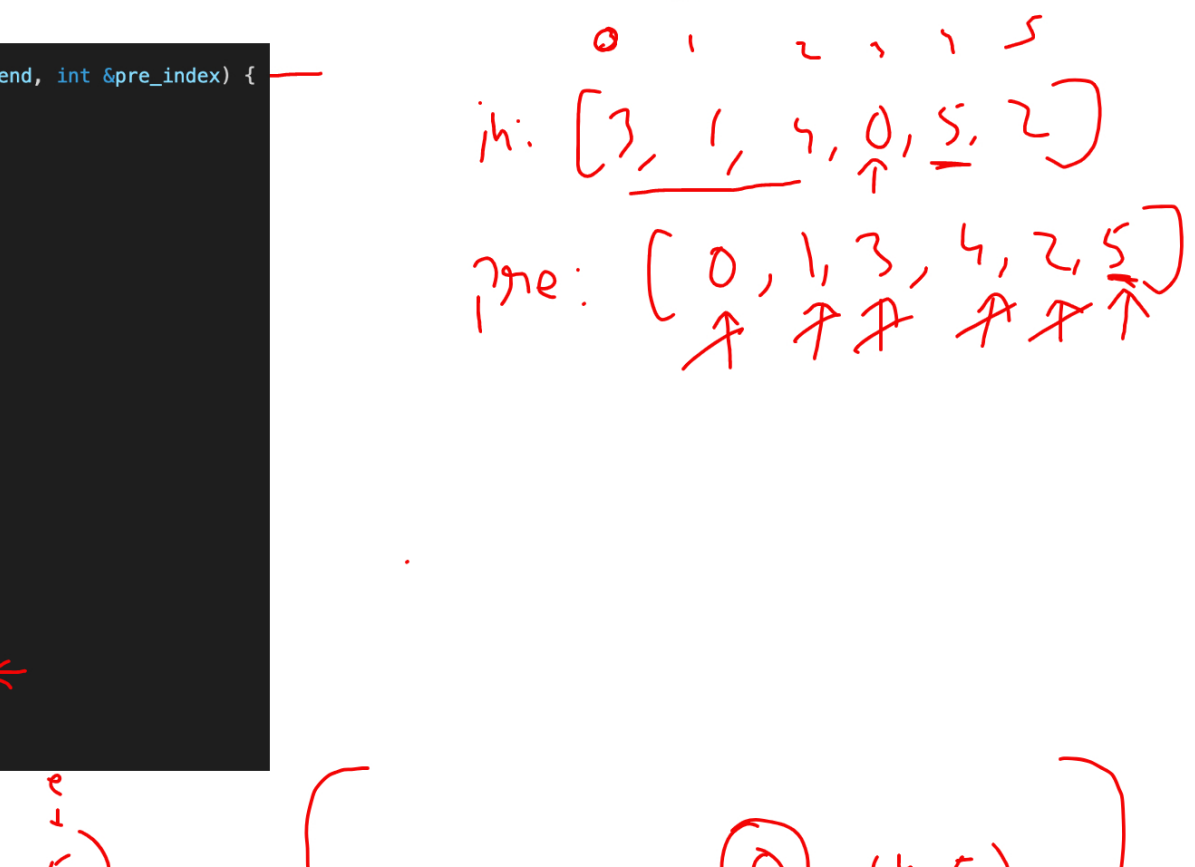

(3, 1, 4)
0
(5, 2)
1
(3)
(4)
2
(5)
3
4
5

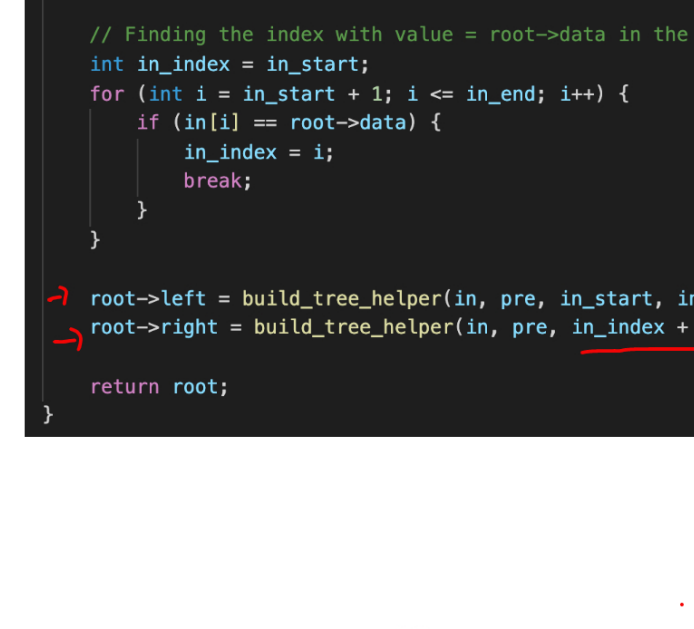


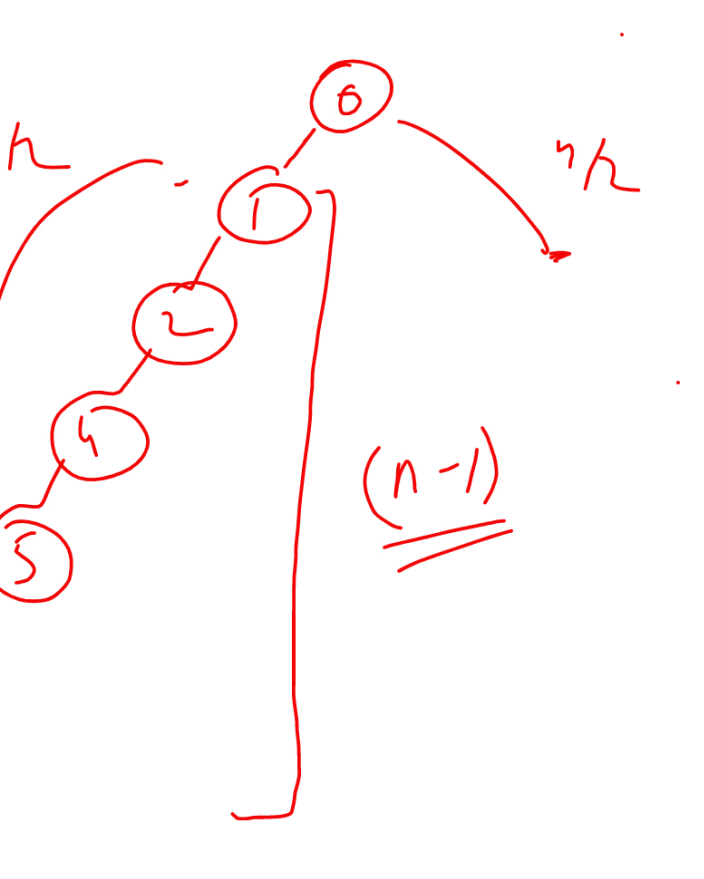
```
Node* build_tree_helper(vector<int> &in, vector<int> &pre, int in_start, int in_end, int &pre_index) {
    if (in_start > in_end) {
        return NULL;
    }
    Node *root = new Node(pre[pre_index]);
    pre_index++;

    // Finding the index with value = root->data in the inorder array.
    int in_index = in_start;
    for (int i = in_start + 1; i <= in_end; i++) {
        if (in[i] == root->data) {
            in_index = i;
            break;
        }
    }

    root->left = build_tree_helper(in, pre, in_start, in_index - 1, pre_index);
    root->right = build_tree_helper(in, pre, in_index + 1, in_end, pre_index);

    return root;
}
```

s e
(0, 5)

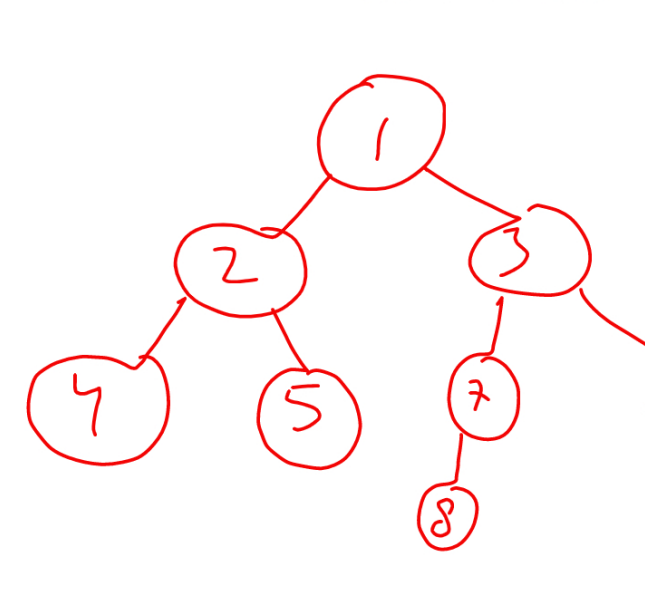
in: [3, 1, 4, 0, 5, 2]
pre: [0, 1, 3, 4, 2, 5]


0
(0, 2)
(3, 5)
1
(0, 0)
(2, 2)
2
(4, 5)
(0, -1)
3
4
5
NULL
NULL


n/2
n/2
0
1
2
4
5
n
(n-1)

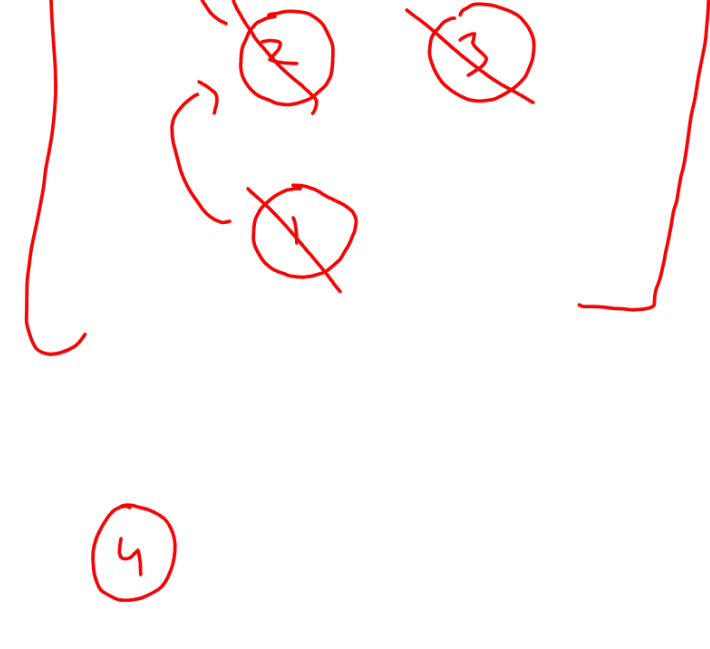


[4, 2, 5, 1, 3, 6]
→ left
→ current
→ right



[4, 2, 5, 1, 8, 7, 3, 6] ✓

Stack <Node*>
push left (node) ✓



~~5~~
~~4~~
~~2~~
~~1~~

~~6~~
~~8~~
~~7~~
~~3~~
~~5~~
~~4~~
~~2~~
~~1~~

[4, 2, 5, 1, 8, 7, 3, 6] ✓