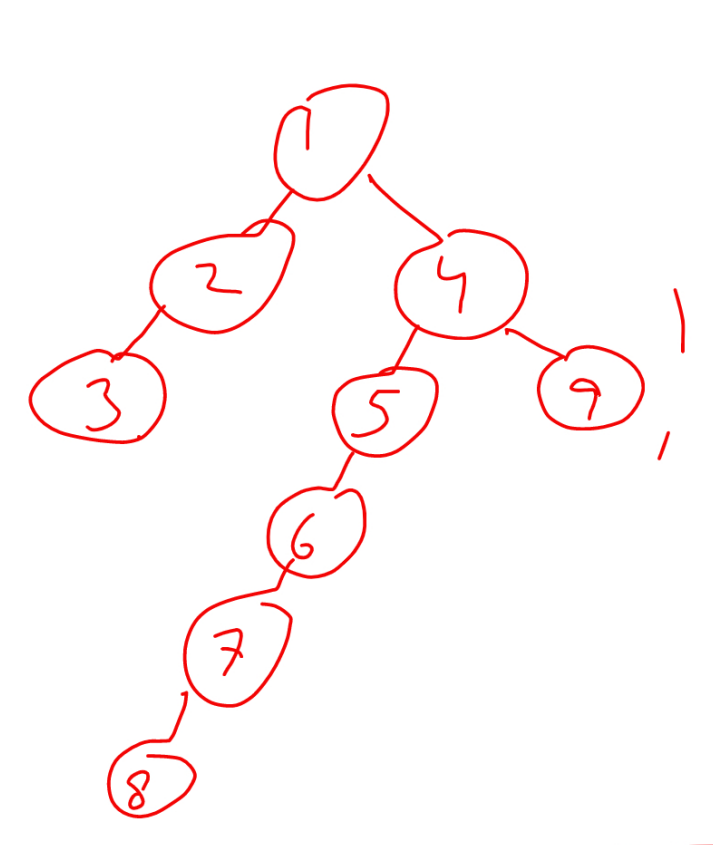


lh=2 rh=5

2+5+1=8

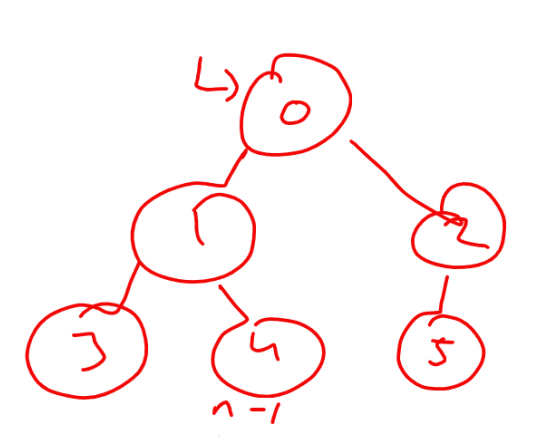
diam =

(2, 8, 6) = 8

---

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

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



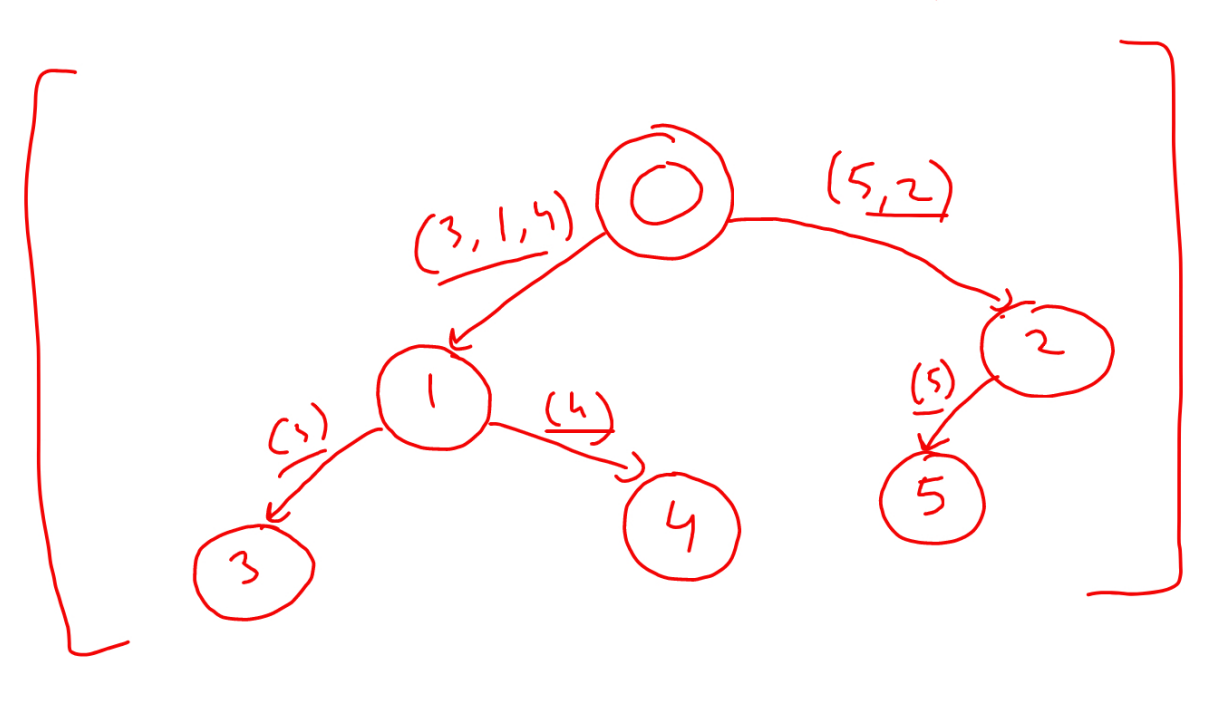
[4]

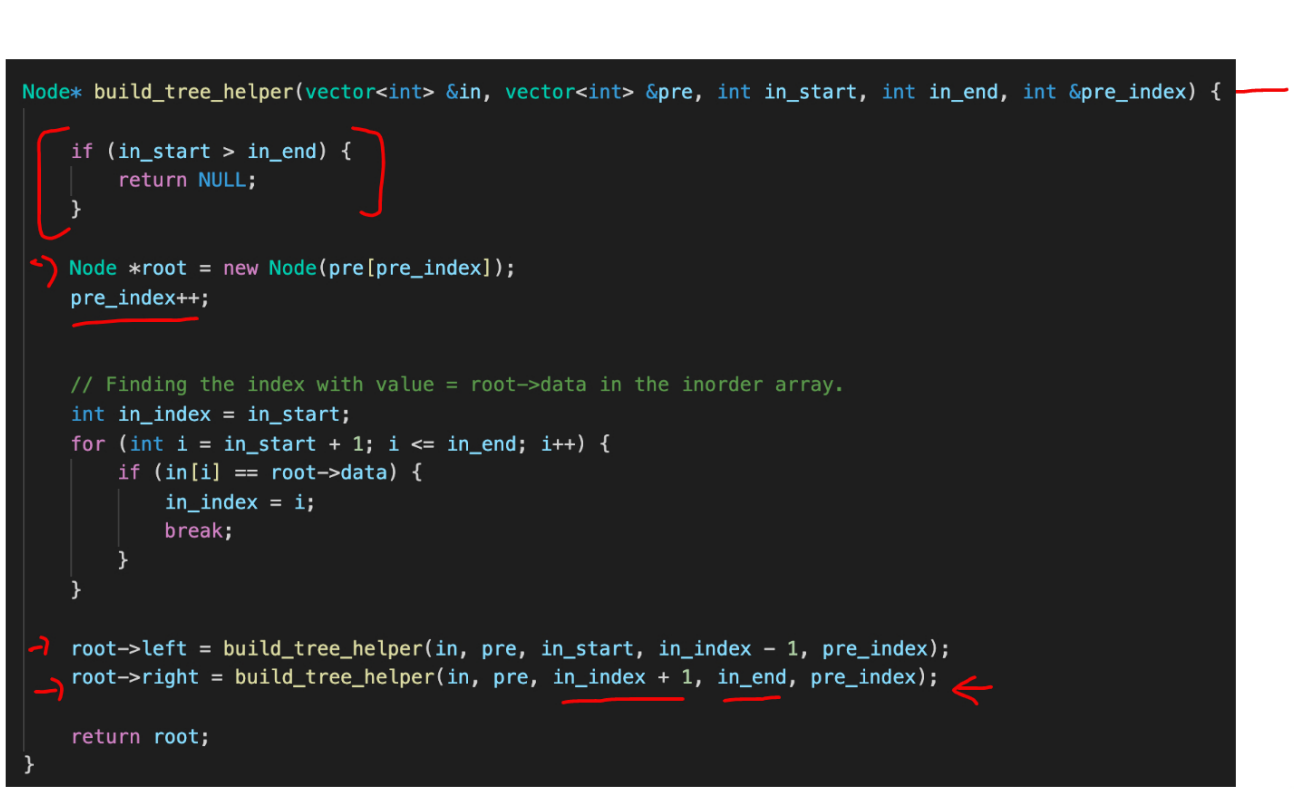
in: [3, 1, 4, 0, 5, 2]

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)
3
4
2
(5)
5

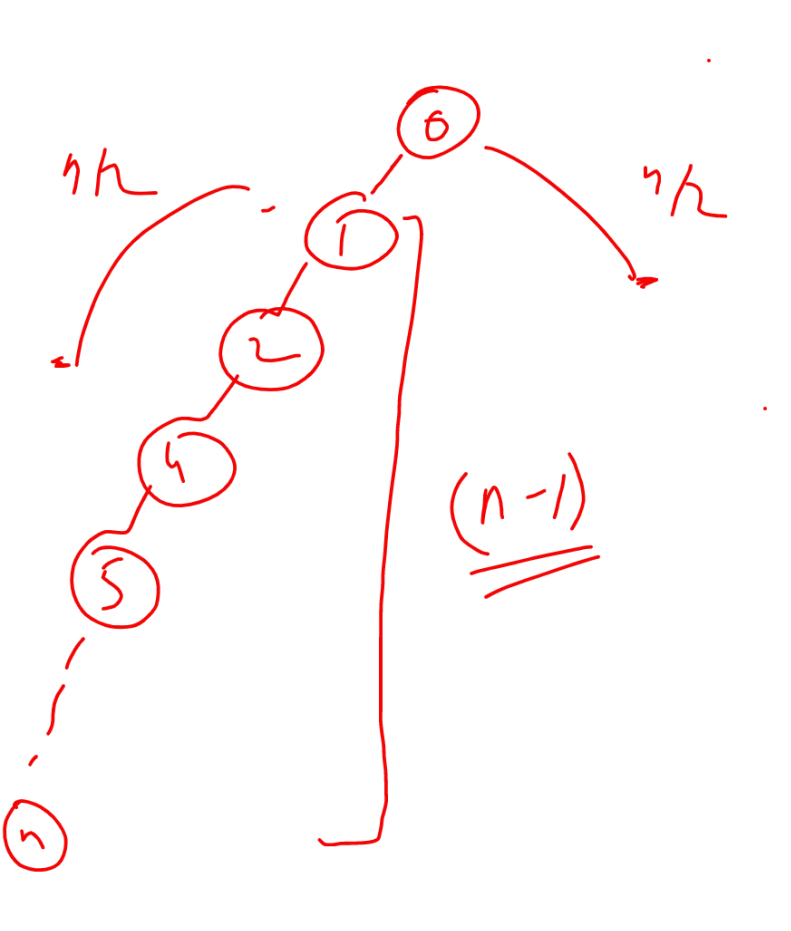


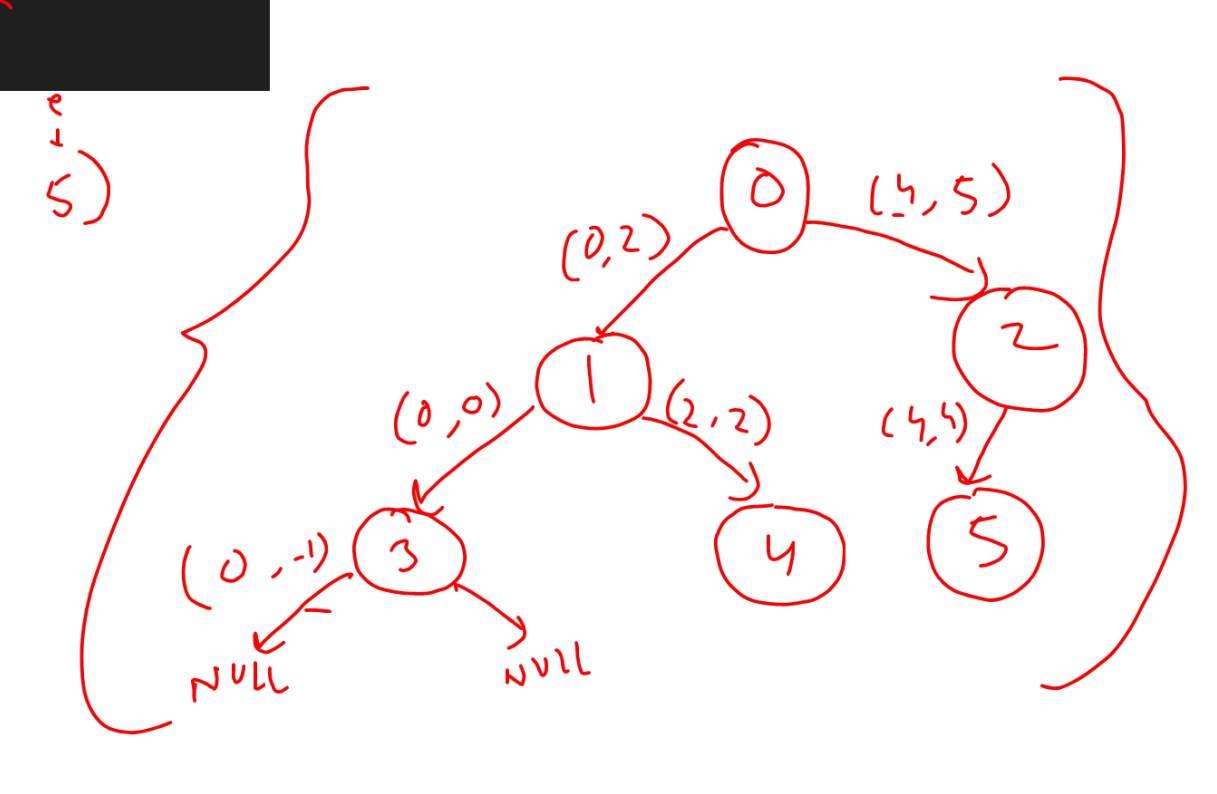
```
Node* build_tree_helper(vector<int> &in, vector<int> &pre, int in_start, int in_end, int &pre_index) {
    if (in_start > in_end) {
        return NULL;
    }
    Node *root = new Node(pre[pre_index]);
    pre_index++;

    // 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;
}
```

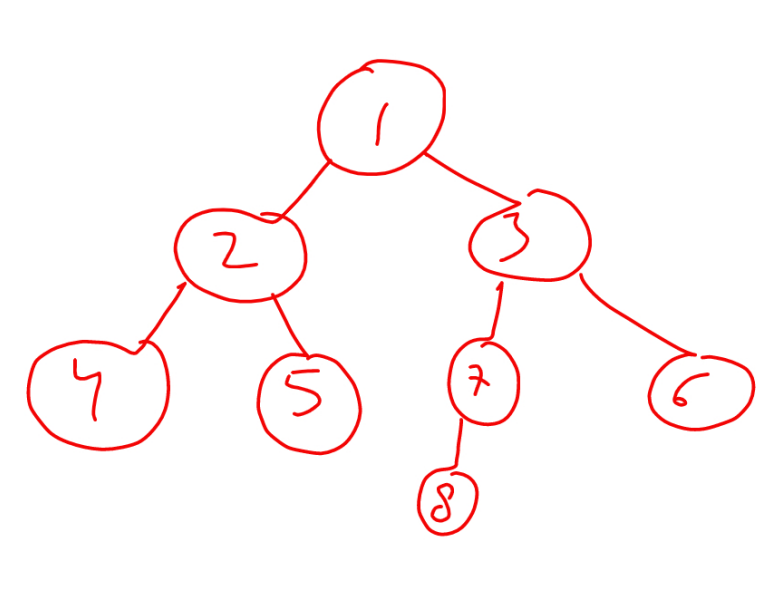
in: [3, 1, 4, 0, 5, 2]

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

(0, 5)


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


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

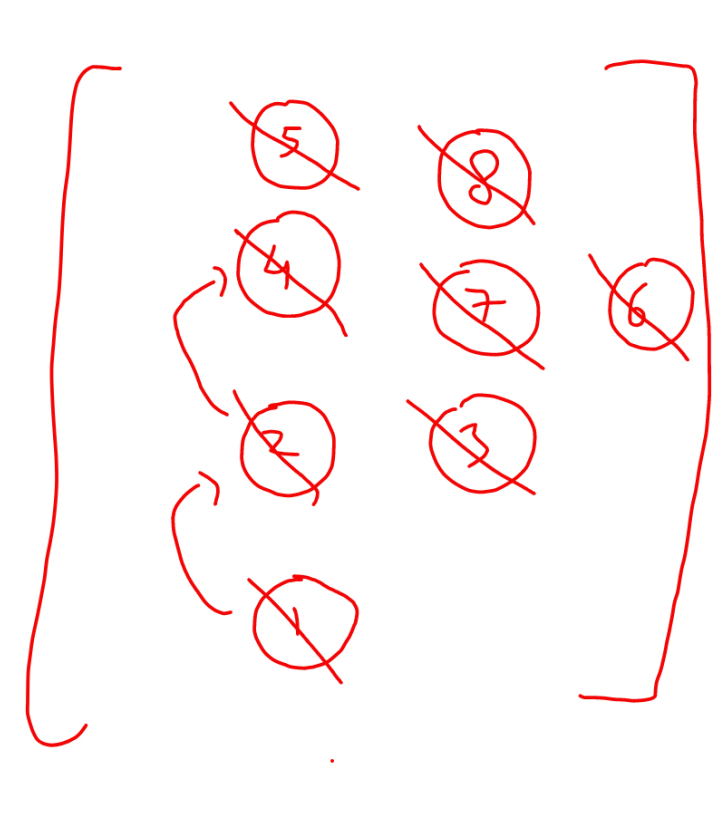


[4, 2, 5, 1, 3, 6]

→ left

→ current ⇐

→ right



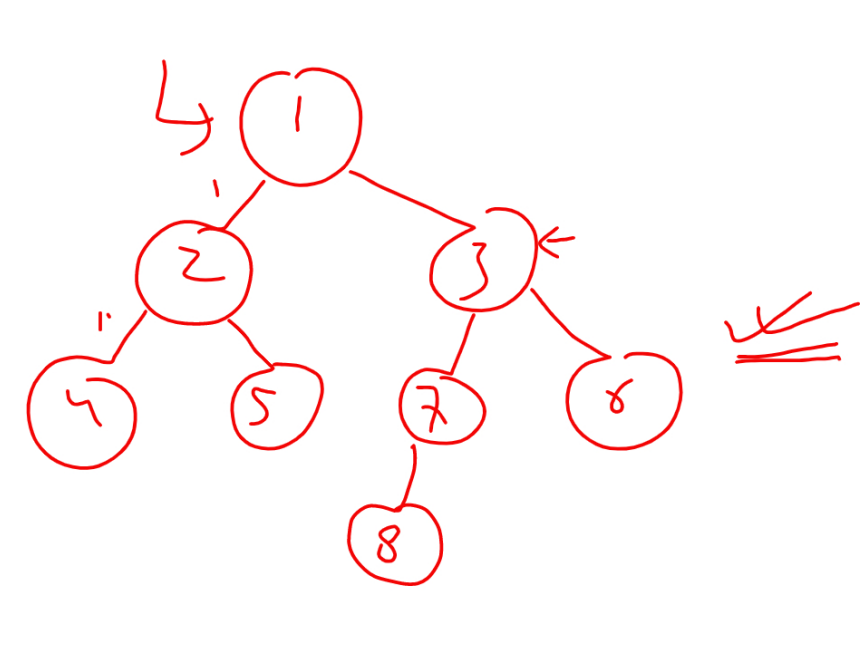
[4, 2, 5, 1, 8, 7, 3, 6] ✓

Stack <Node*>

push left (node) ✓

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

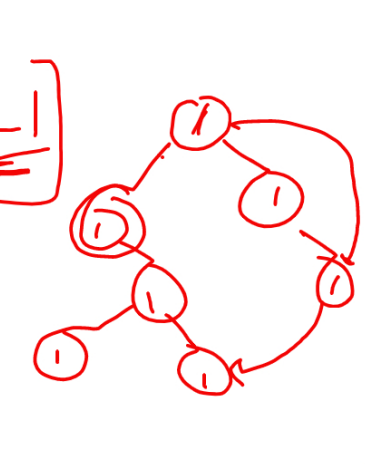
~~6~~
~~8~~
~~7~~
~~3~~
~~5~~
~~4~~
~~2~~
~~1~~



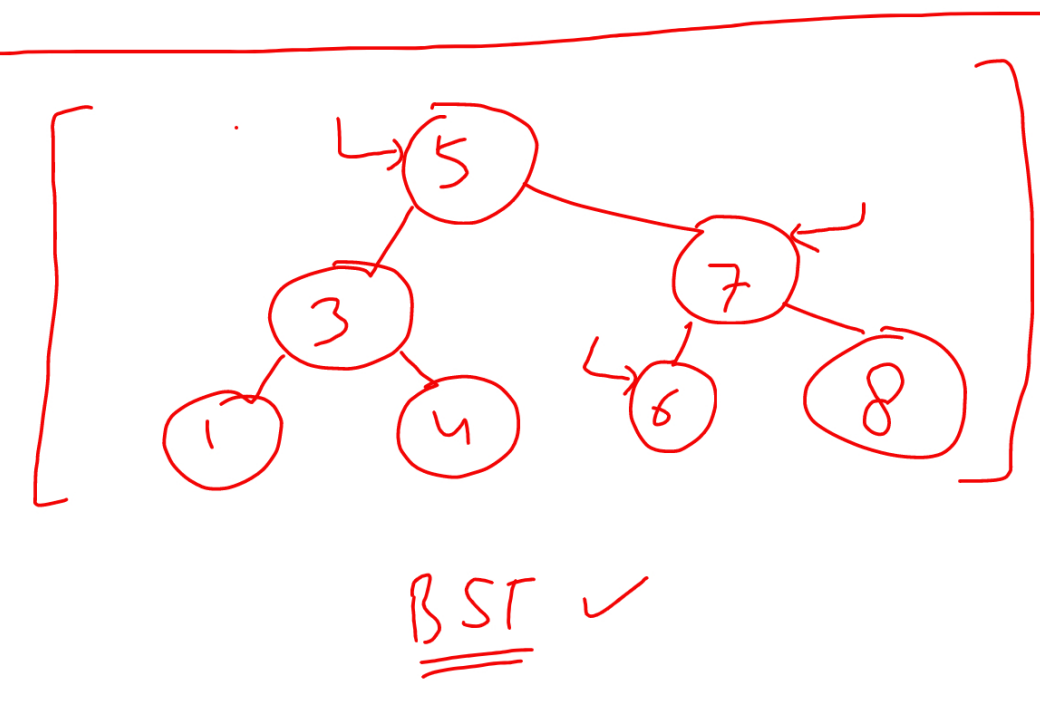
[4, 2, 5, 1, 8, 7, 3, 6] ✓

N

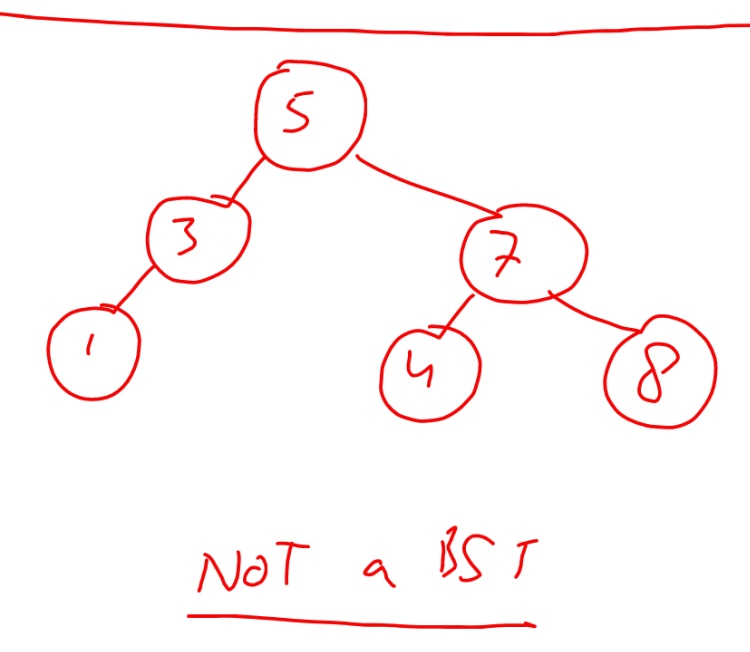
N-1



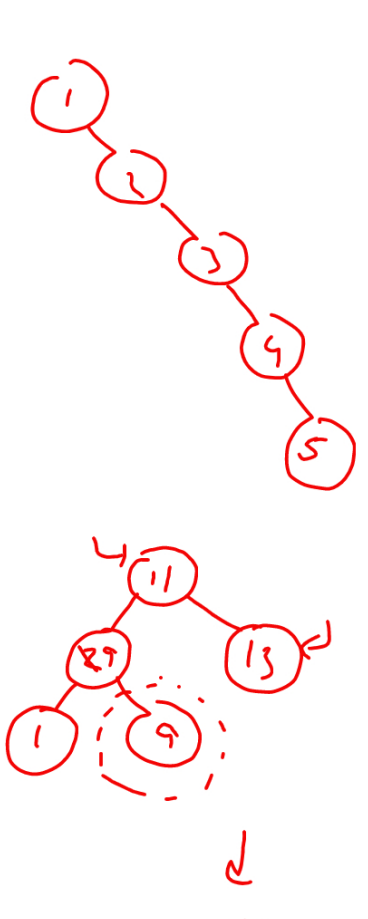
(LCA)



BST ✓

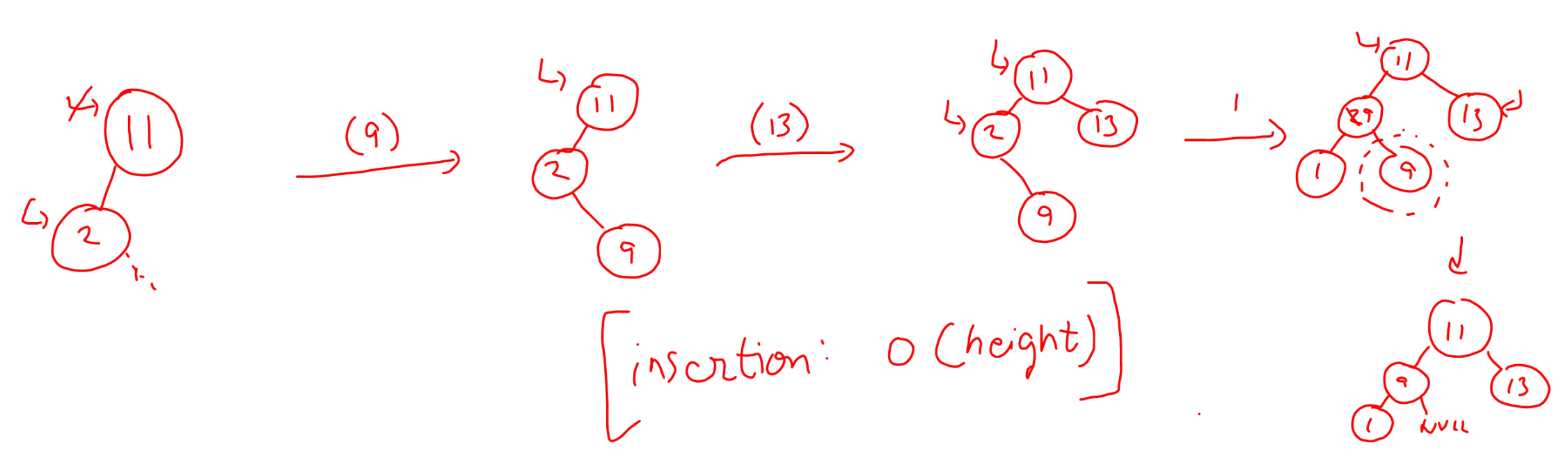


NOT a BST

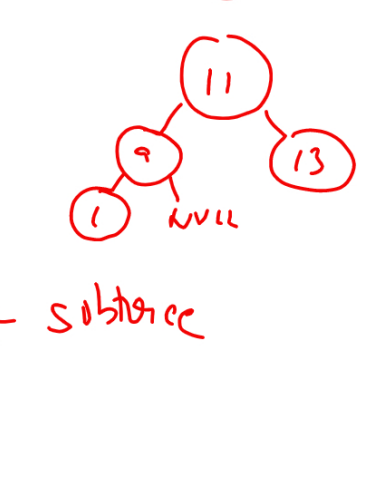


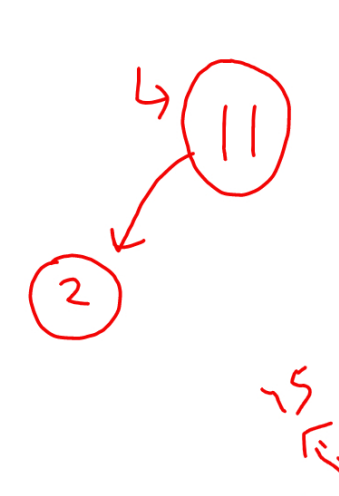
Search 6

[O(height)] ✓


11
2
(9)
11
2
9
(13)
11
2
13
9
11
13
9

[insertion: O(height)]





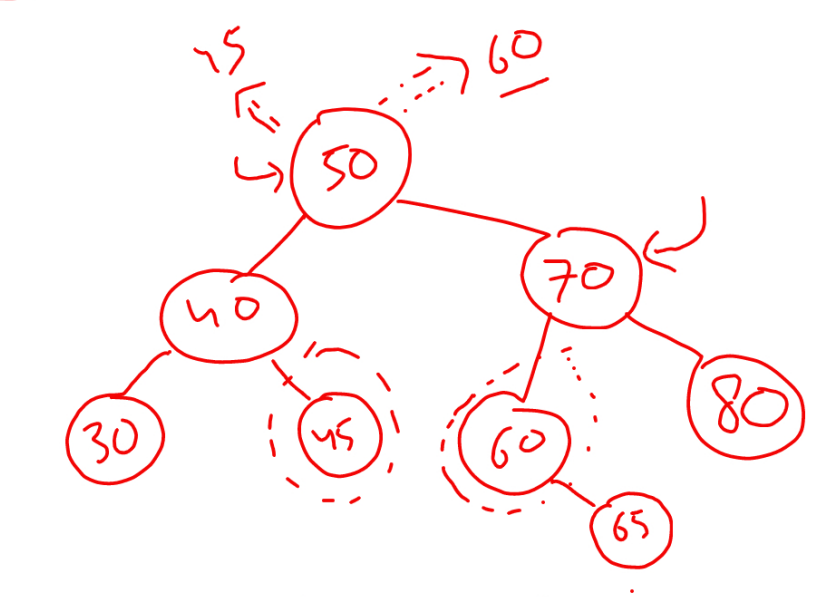
(2)

smallest node in the right subtree

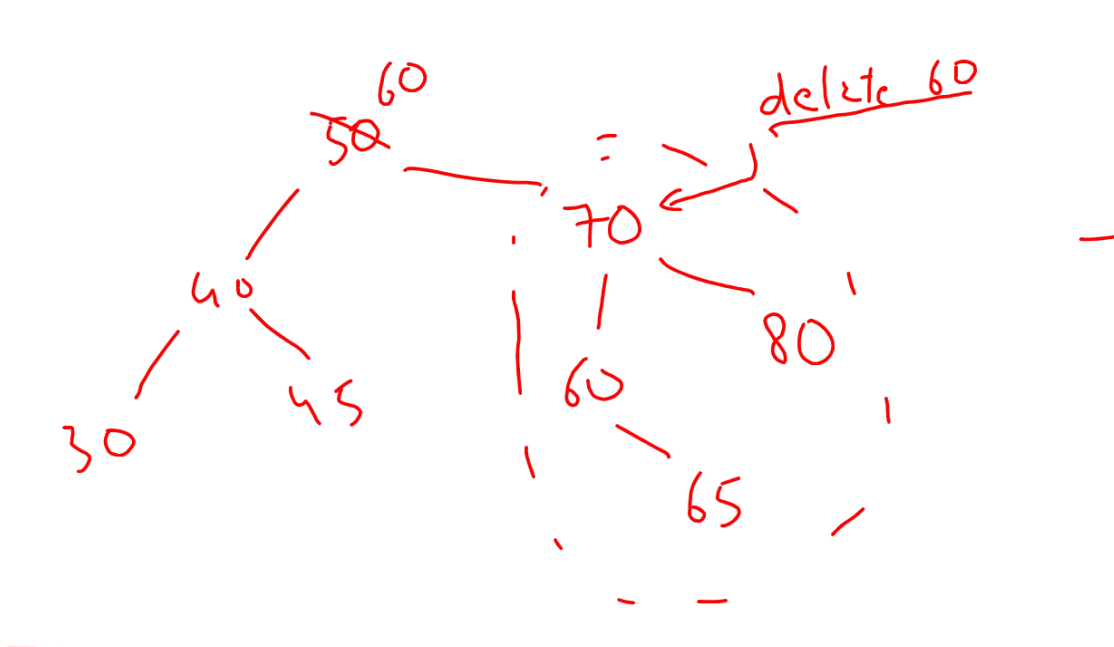
inorder successor → 60

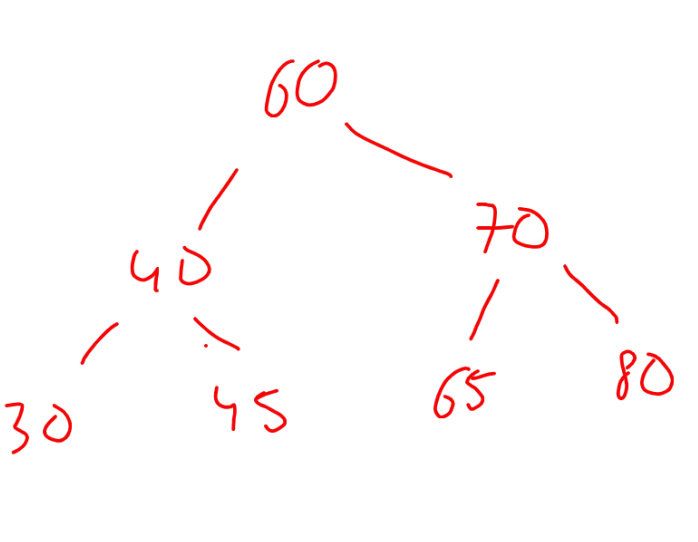
inorder predecessor → 45

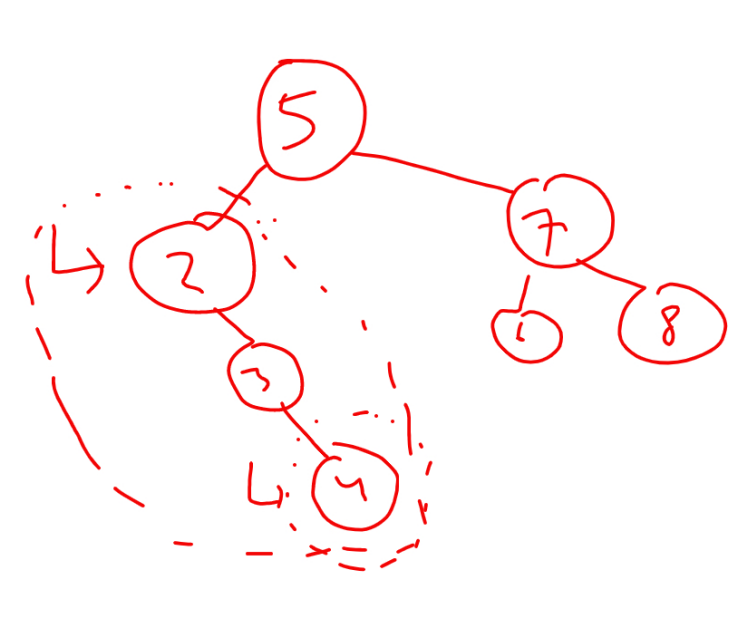
largest node in the left subtree

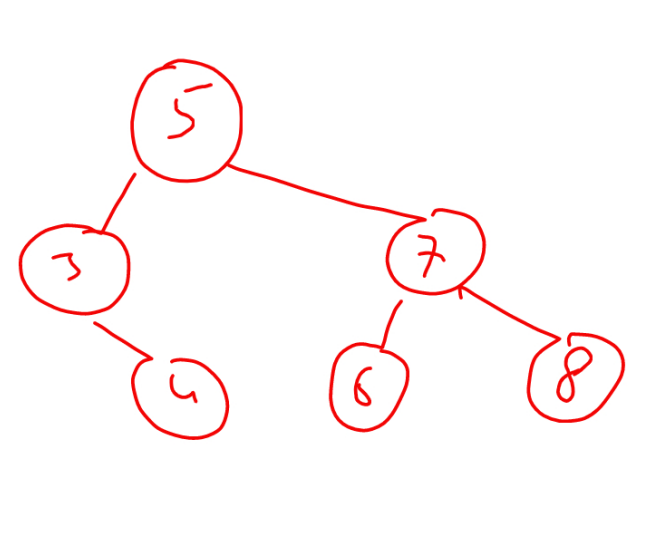

60
50
40
70
30
45
60
80
65

delete 50

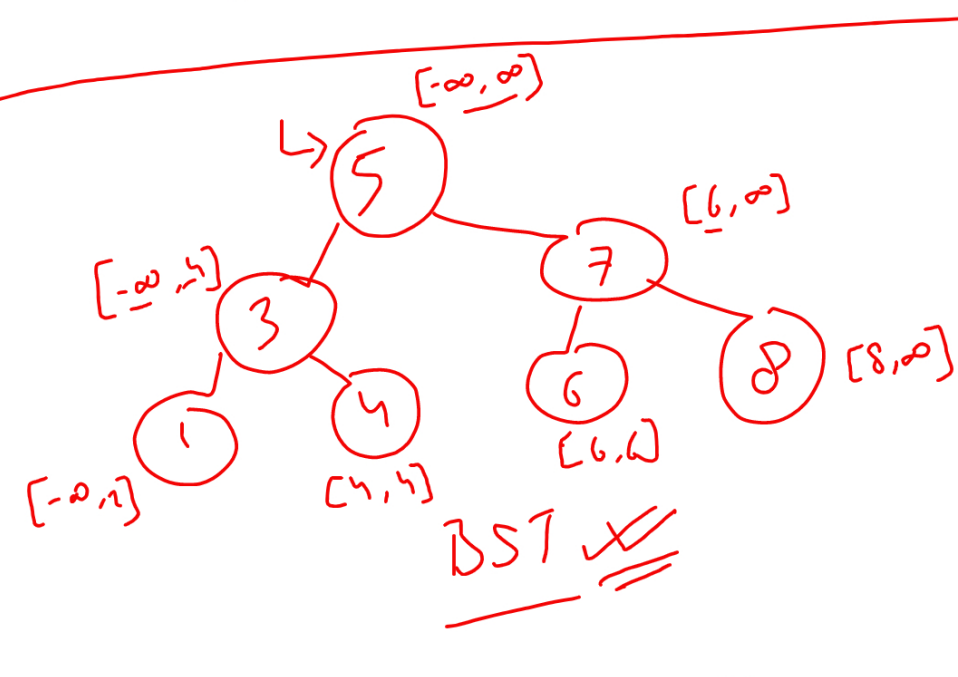

60
~~50~~
delete 60
70
40
30
45
60
80
65


60
40
70
30
45
65
80



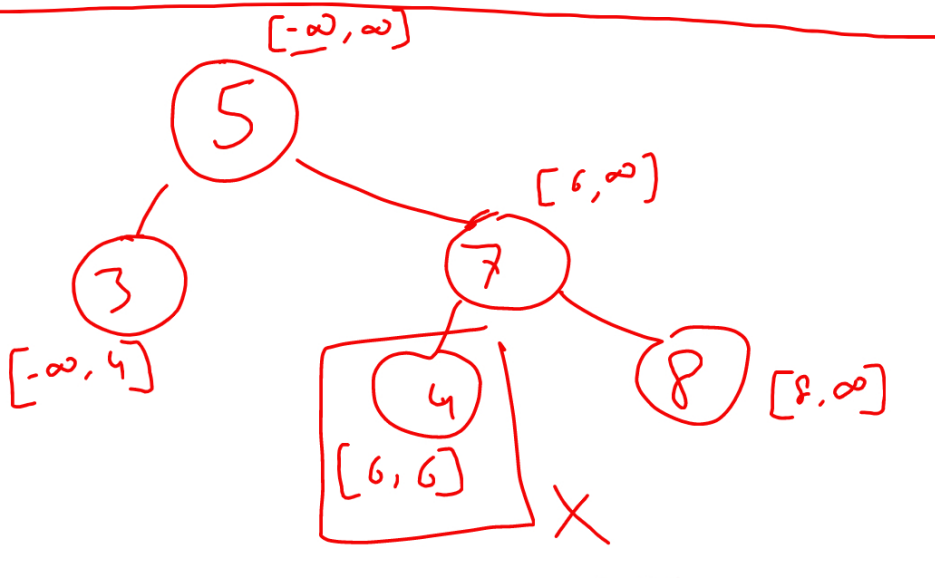


delete 2


[-∞, ∞]
5
[-∞, 5]
3
[5, ∞]
7
[-∞, 3]
1
4
[3, 5]
6
[5, 7]
8
[7, ∞]

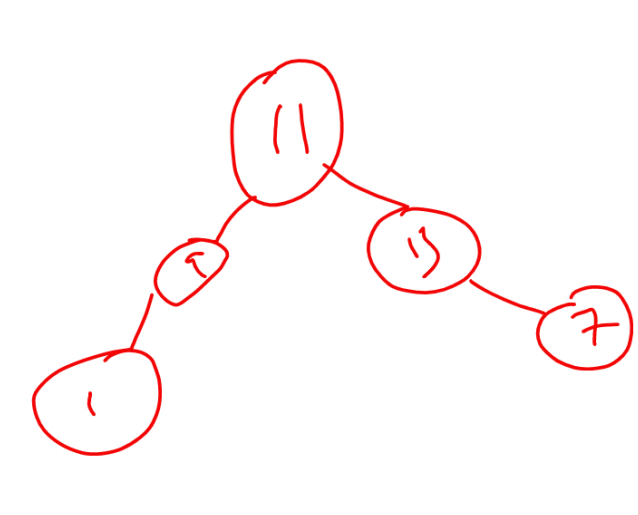
BST ✓

[1, 3, 4, 5, 6, 7, 8]


[-∞, ∞]
5
3
[-∞, 4]
[5, ∞]
7
4
[6, 6]
8
[7, ∞]

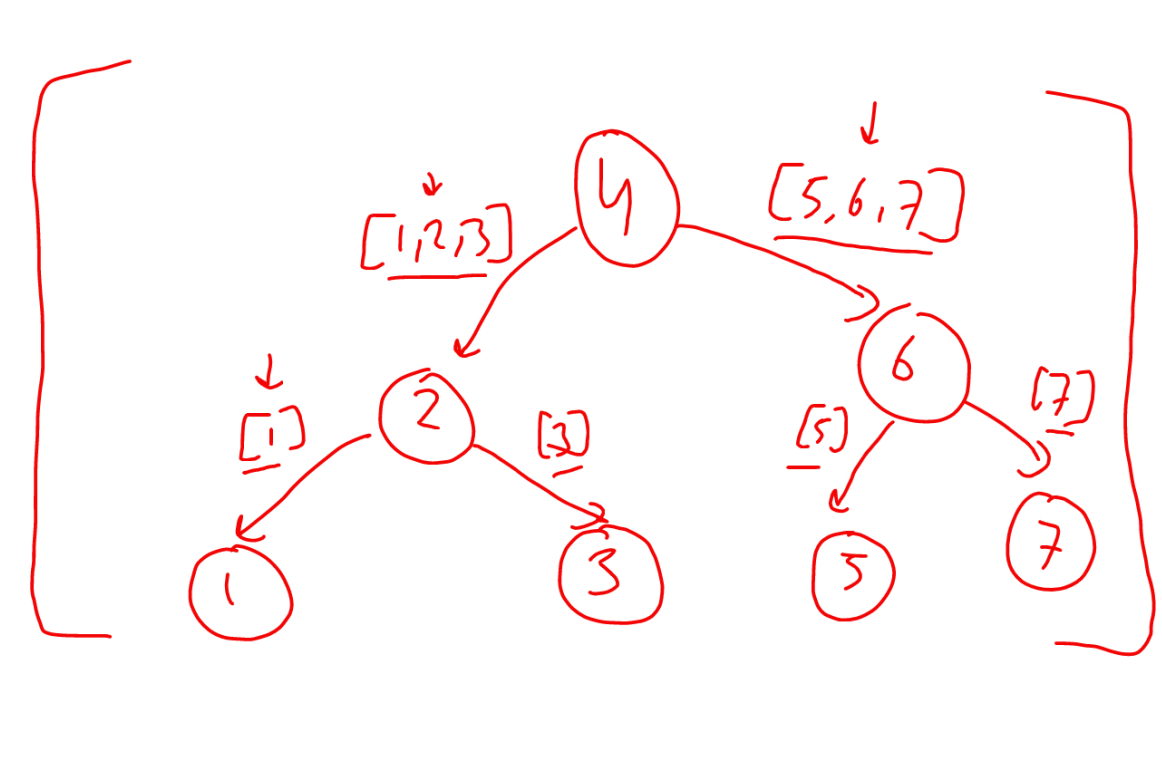
not a BST

search 4 root ~~5~~ 3 4



[1, 2, 3, 4, 5, 6, 7]

abs(lh - rh) ≤ 1 for all nodes


[1, 2, 3]
4
[5, 6, 7]
2
6
[1]
[3]
[5]
[7]
1
3
5
7