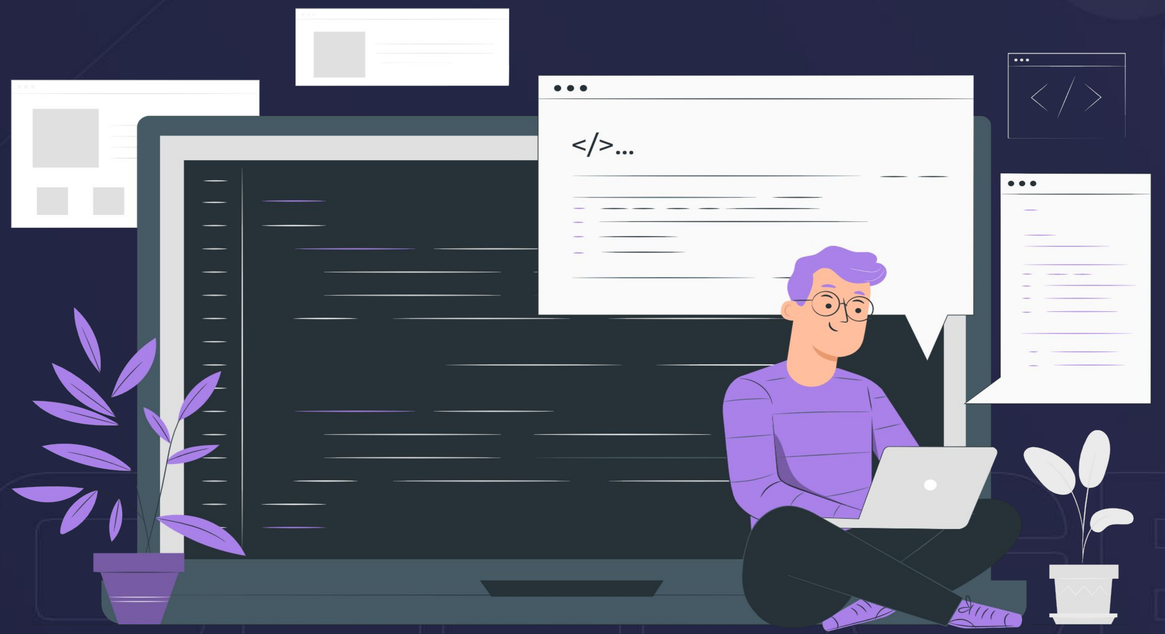
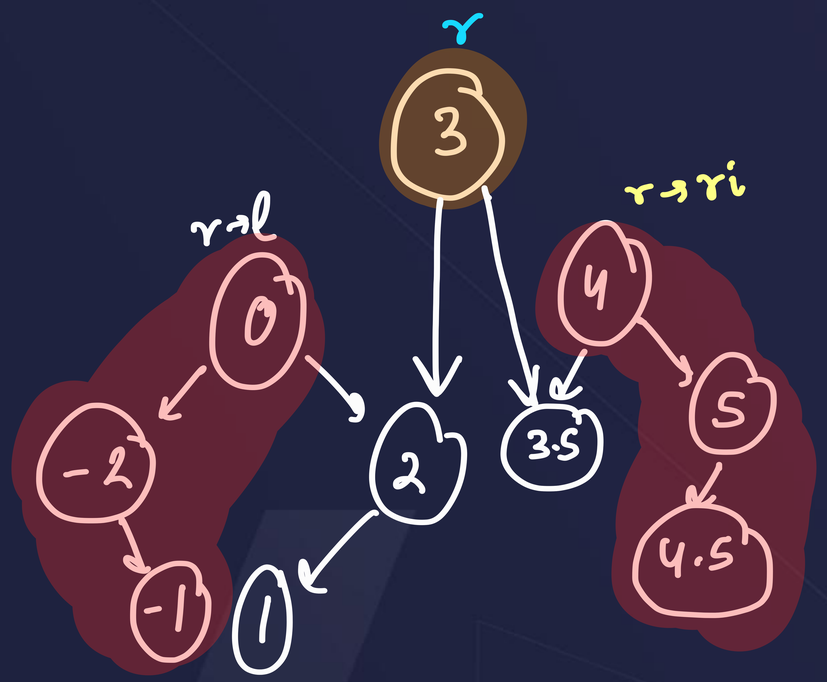
Lecture 57
Binary Search Trees – 3



Recap

- Searching, Insertion, Deletion
- Interview Problems on BST

Ques: Trim a Binary Search Tree [LeetCode 669]

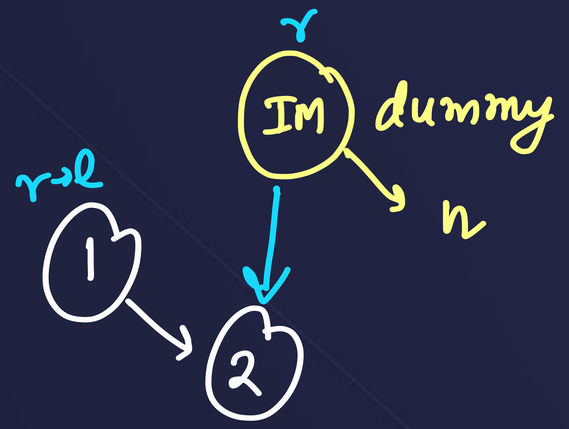


lo = 1
hi = 3

```
if(r->left->val < lo){
    r->left = r->left->right
}

if(r->right->val > hi){
    r->right = root->right->left
}
```

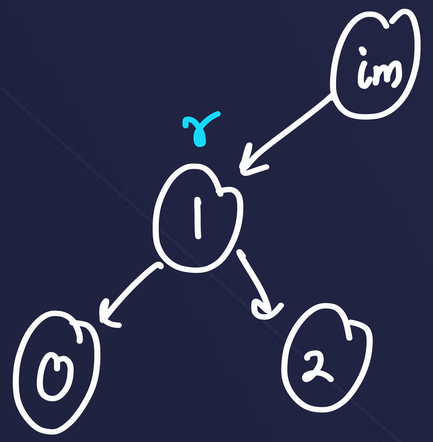
Ques: Trim a Binary Search Tree [LeetCode 669]




```
if(root->left->val < lo)
    root->left = root->left->right
```

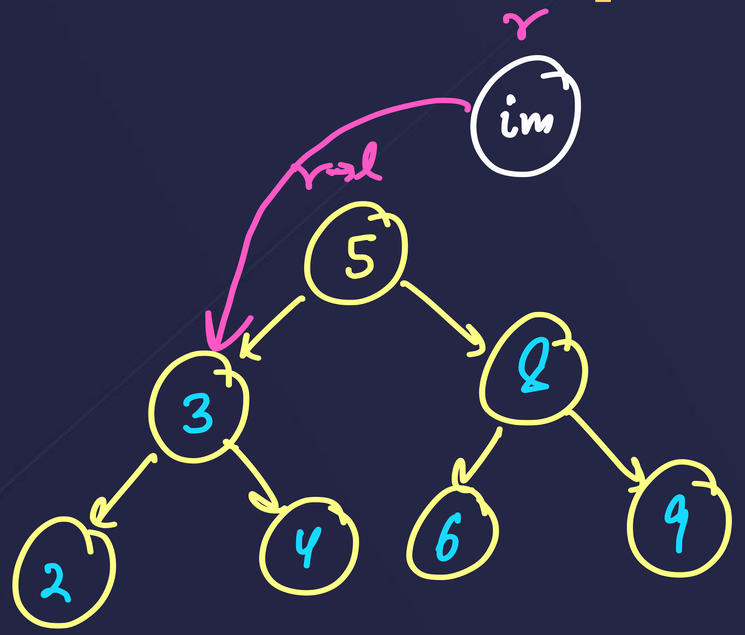
lo = 2, hi = 4

Ques: Trim a Binary Search Tree [LeetCode 669]



lo = 1

hi = 2



lo = 1
hi = 4

```
if(r->left->val > hi)
    r->left = r->left->left
```

Morris Traversal = Inorder Traversal

Pre, In, Post → Recursive / Iterative T.C. = O(n) S.C. = O(h) / O(n)

Iterative inorder traversal → S.C. = O(1)

↳ start curr with root

↳ 1) curr → left exists → pred
link
unlink & visit

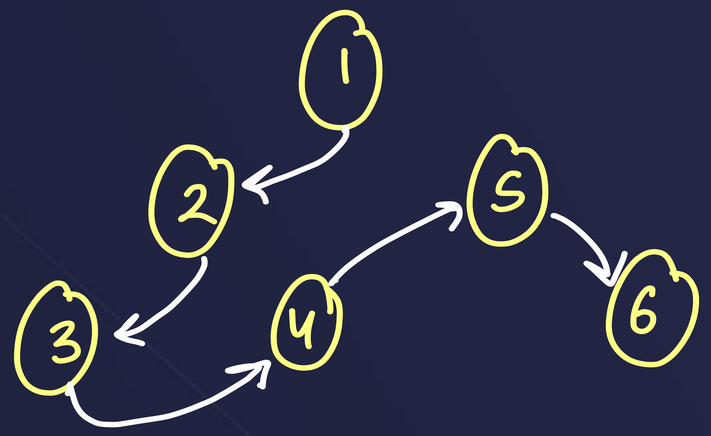
2) curr → left == NULL → visit(curr)
curr = curr → right

Morris Traversal → Predecessor (Inorder)


```
while(curr != NULL){
    if(curr → left != NULL){ //find pred
        pred = curr → left
        while(pred → right != NULL && p → right != c)
            pred = pred → right
        if(pred → right == NULL) //link
        |   pred → right = curr
        |   curr = curr → left
        }
        if(pred → right == c){// unlink
            pred → right = NULL
            visit(curr)
            curr = curr → right
        }
    }
    else{
        visit(curr)
        curr = curr → right
    }
}
```

Ques: Flatten Binary Tree to Linked List [LeetCode 114]

M-I :



Make a preorder vector of treenodes.

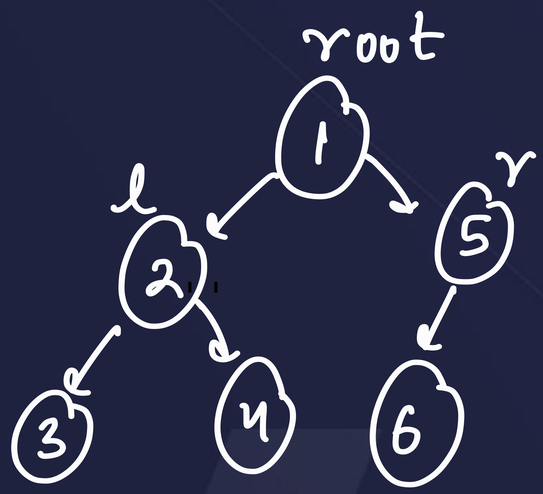
vector<TreeNode*> ans = { 1, 2, 3, 4, 5, 6 }

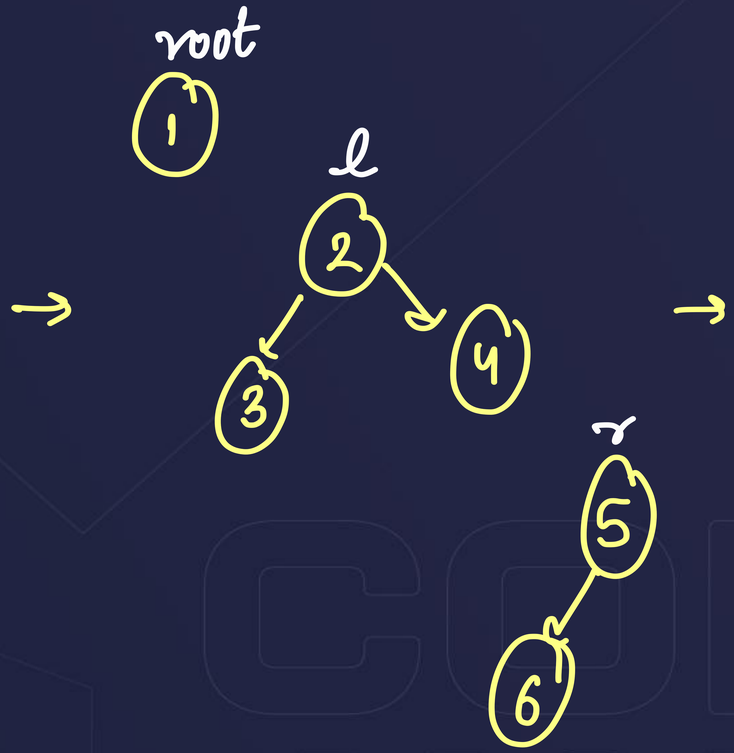
T.C. = O(n) S.C. = O(n)

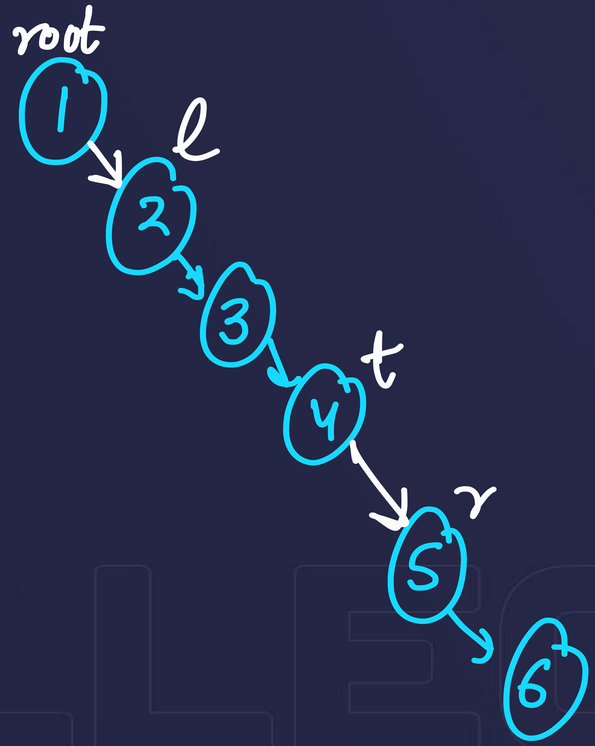
Ques: Flatten Binary Tree to Linked List [LeetCode 114]

M-2 : Recursion :

Preorder → Root Left Right







l = root → left

r = root → right

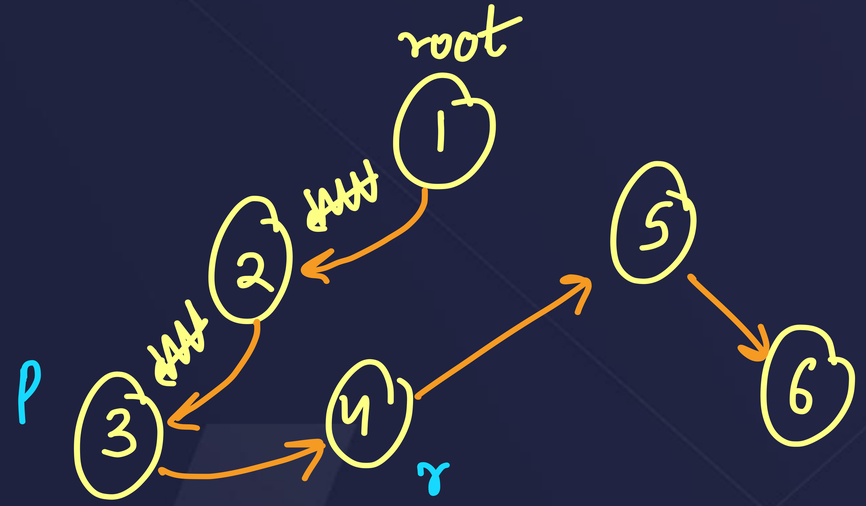
T.C = O(n)

S.C = O(n)

Ques: Flatten Binary Tree to Linked List [LeetCode 114]

M-3 : Morris Traversal

S.C. = O(1)



Curr, pred, right

```
if (C->left != NULL) {//find pred & make r
    r = curr->right
    curr->right = curr->left
    finding pred
    Link -> pred->right = r
    curr = curr->left
}
else curr = curr->right
```

C

Next Lecture

- Sets, Maps, Heaps

PW SKILLS

THANK YOU

COLLEGE WALLAH