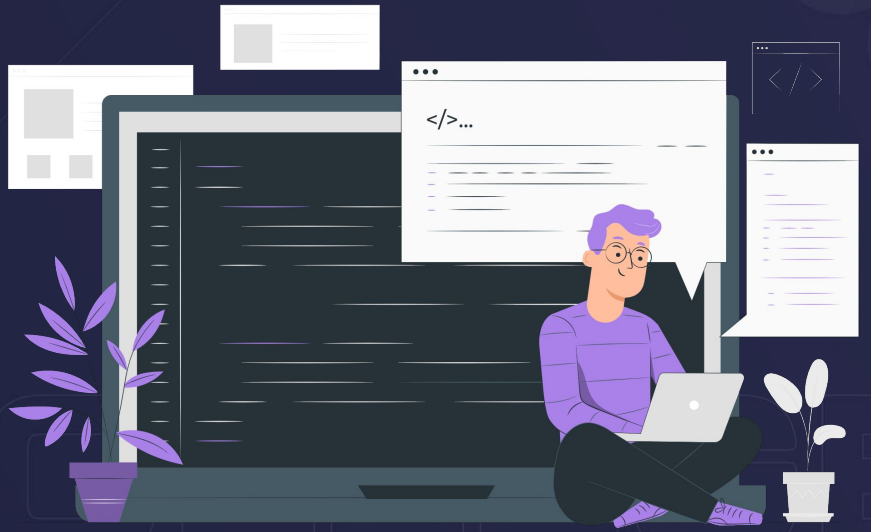
PW SKILLS

Lecture – 53

Binary Trees

[Traversals]

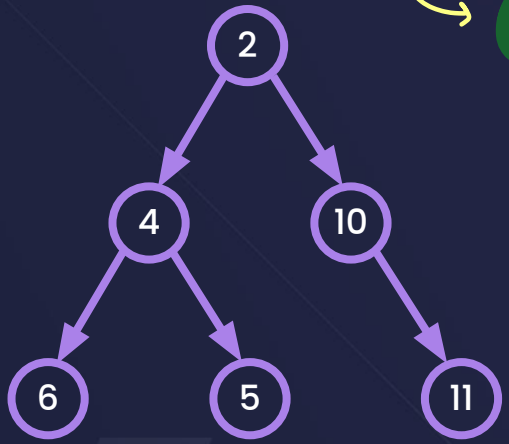


Recap

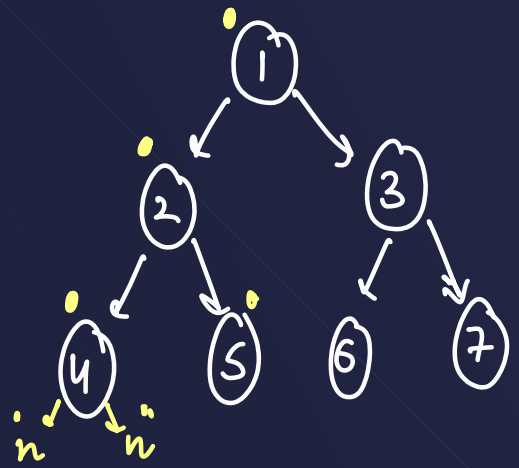
- Types of Trees, Terminology, (Size, Sum, Max, height) problems
- Some Leetcode questions

Traversals → to visit every node/member/ element

↳ DFS, BFS, Morris Traversal



1. Preorder Traversal




```
void display(Node* root){
    if(root==NULL) return; // base case
    cout<<root->val<<" "; // work
    display(root->left); // call 1
    display(root->right); // call 2
}
```

Output

1 2 4 5 3 6 7

1. Preorder Traversal

Root Left Right

⇒ 1 (2 4 5 8 9 10) (3 6 7 11)

⇒ 1 2 4 8 5 9 10 (3 6 7 11)

⇒ 1 2 4 8 5 9 10 3 6 11 7

T.C. = O(n) [where 'n' is the no. of nodes]

S.C. =

1 2 4 8 5 9 10 3 6 11 7

1. Preorder Traversal

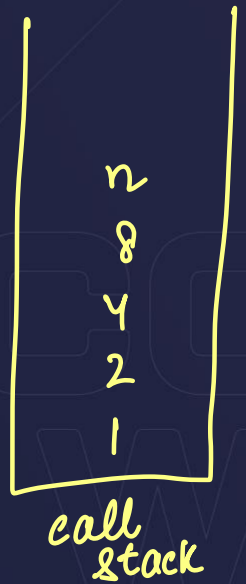
T.C. = O(n) [where 'n' is the no. of nodes]

S.C. = O(levels + 1) ≈ O(h+2) ≈ O(h)

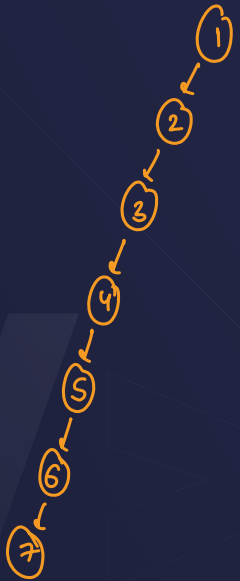
O(h)

O(n) — avg. case / worst case

O(log n)



Degenerate Trees (Worst Case)



→ Pre/In/Post : TC = O(n)

SC = O(n) or O(h)

Balanced Trees :

$$S.C. = O(h) = O(\log n)$$

$$\underbrace{1 + 2^1 + 2^2 \cdots 2^h}_{} = n$$

$$\Rightarrow \frac{1(2^{h+1} - 1)}{2-1} = n$$

$$\Rightarrow 2^h \cdot 2 = n+1 \Rightarrow h = \log_2\left(\frac{n+1}{2}\right)$$

2. Inorder Traversal

Left Root Right

⇒ (2 4 5) 1 (3 6 7)

4 2 5 1 6 3 7

4	2	5	1	6	3	7

4 2 5 1 6 3 7

3. Postorder Traversal

Left Right Root

⇒ (2 4 5) (3 6 7) 1

⇒ 4 5 2 6 7 3 1

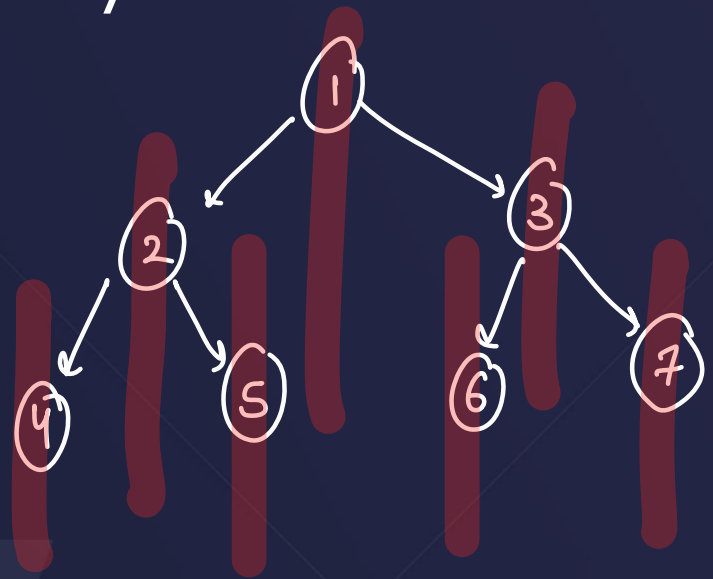
Ques: Binary Tree Preorder Traversal [LeetCode 144]

1 2 4 5 3 6 7

↓

preorder

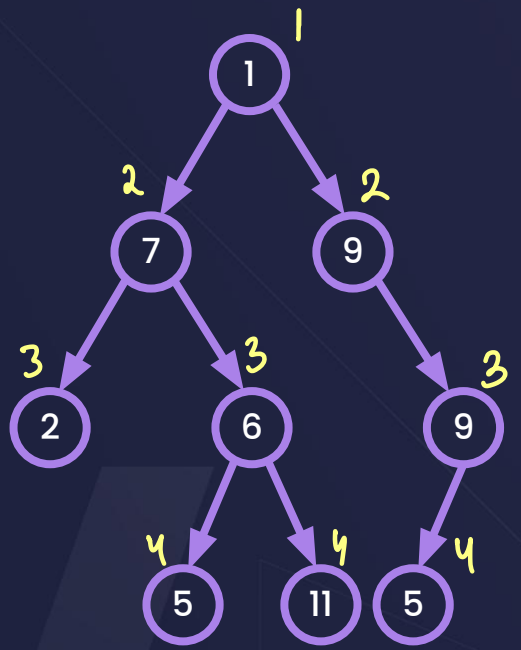
Ques: Binary Tree Inorder Traversal [LeetCode 94]



4 2 5 1 6 3 7

Ques: Binary Tree Postorder Traversal [LeetCode 145]

Q: Print elements of nth level



n = 3

2 6 9

```
void print(root, curr, level){
    if(curr = level) cout << root->val;
    print(root->left, curr+1, level);
    print(root->right, curr+1, level);
}
```

1 3 3

n^{th} Level : Homework → T.C. & S.C

```
void nthLevel(Node* root, int curr, int level){
    if(root==NULL) return; // base case
    if(curr==level) cout<<root->val<<" "; // root
    nthLevel(root->left, curr+1, level); // left
    nthLevel(root->right, curr+1, level); // right
}
```

] → these are needless/extra if you have already reached the read level

Optimised n^{th} Level : Homework T.C. & S.C

```
void nthLevel(Node* root, int curr, int level){
    if(root==NULL) return; // base case
    if(curr==level){
        cout<<root->val<<" "; // root
        return;
    }
    nthLevel(root->left, curr+1, level); // left
    nthLevel(root->right, curr+1, level); // right
}
```

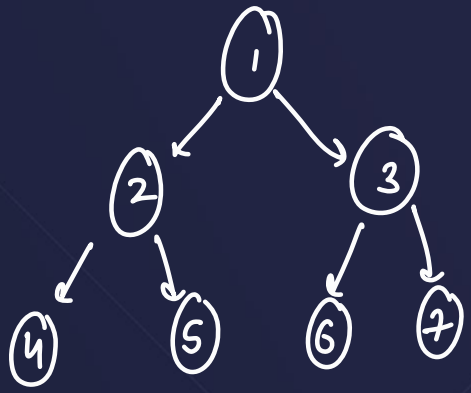
T.C. =

S.C. =

Level order traversal (~~BFS~~)

n^{th} Level (DFS)

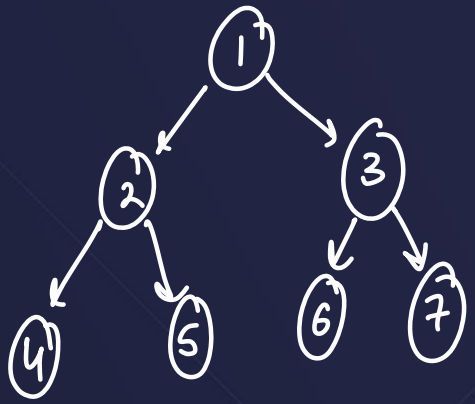
Loop



⇒ 1 2 3 4 5 6 7

⇒ 1
2 3
4 5 6 7

*Level order traversal (Right to Left)



⇒

1 3 2 7 6 5 4

or

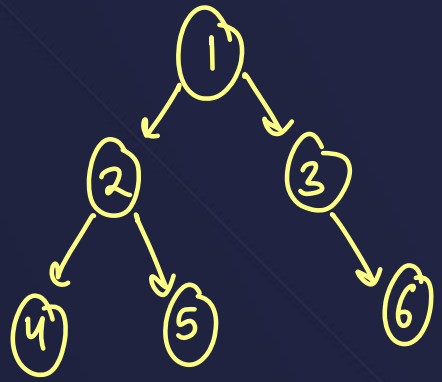
1

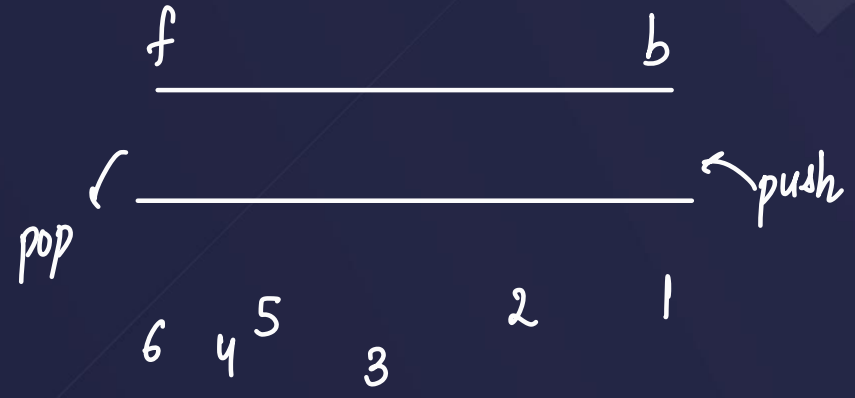
3 2

7 6 5 4

Ques: Binary Tree Level Order Traversal [LeetCode 102]

Level order traversal (Using Queue) BFS





queue<Node*> q;

Steps:

1) Node* temp = q.front(), q.pop(), print
2) push temp->left & temp->right to q

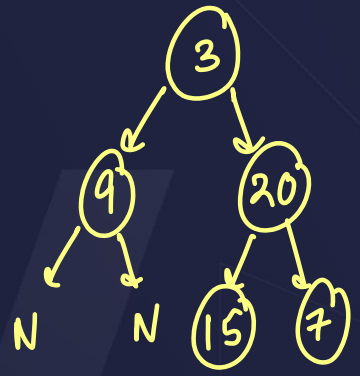
1 2 3 4 5 6

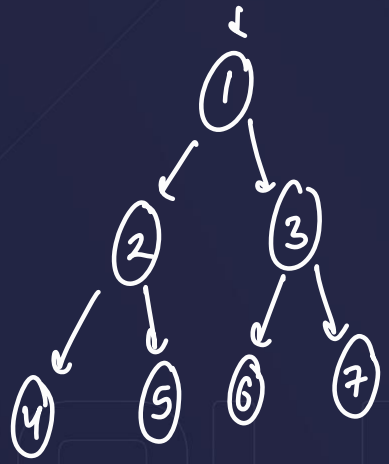
Construct Tree from Level order traversal

↓

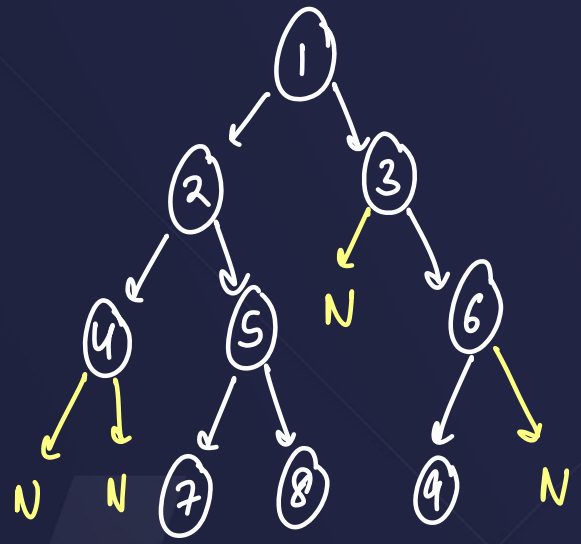
Q give an array, For Ex → 1 2 3 4 5 6 7

{3, 9, 20, N, N, 15, 7}





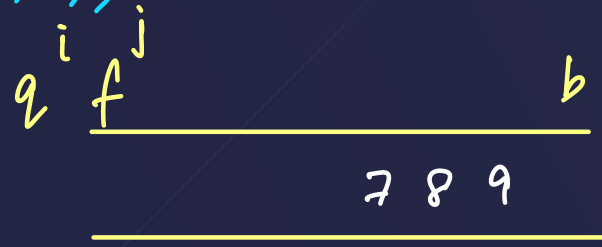
Construct Tree from Level order traversal



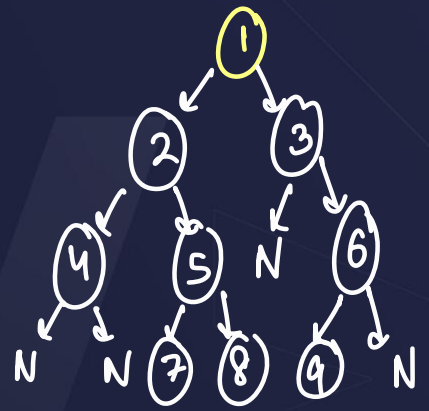
{ 1, 2, 3, 4, 5, N, 6, N, N, 7, 8, 9, N }

Construct Tree from Level order traversal (queue)

{ 1, 2, 3, 4, 5, N, 6, N, N, 7, 8, 9, }



Node* root = new Node(arr[0])



N → INT_MIN

2 3

4

6 5

Steps : Node* temp = q.front(), q.pop()

Node* l = new Node(arr[i])

Node* r = new Node(arr[j])

temp → left = l

temp → right = r → q.push(l), r

i += 2;

j += 2;

SKILLS

THANK YOU