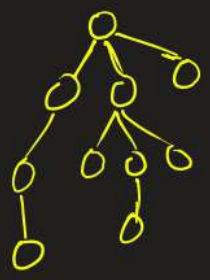
Trees & Graphs Lecture 1

Thursday, 15 August 2024 5:58 AM

Trees



Tree
n-ary tree

Every node has
at most 2 children
Binary tree

n Vertices
⇒ n-1 Edges
There is a path from every node to every other node.

```
struct TreeNode {
  int val;
  TreeNode * left, right;   <- binary tree
  vector < TreeNode * > children;   <- General tree / n ary tree
};
```

Adjacency matrix / Adjacency list representation

Graph is dense
$O(n^2)$ edges

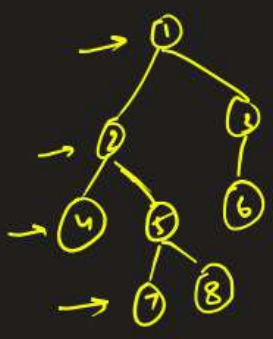
Graph/Tree is sparse
$\sim O(n)$ edges

=x=

https://www.geeksforgeeks.org/problems/binary-tree-representation/1

```
class Solution{
public:

   void create_tree(node* root0, vector<int> &vec){
      root0->left = newNode(vec[1]);
      root0->right = newNode(vec[2]);

      root0->left->left = newNode(vec[3]);
      root0->left->right = newNode(vec[4]);

      root0->right->left = newNode(vec[5]);
      root0->right->right = newNode(vec[6]);
   }

};
```

```
class Solution:
   def createTree(self, root, l):
      root.left = Node(l[1])
      root.right = Node(l[2])

      root.left.left = Node(l[3])
      root.left.right = Node(l[4])

      root.right.left = Node(l[5])
      root.right.right = Node(l[6])
```

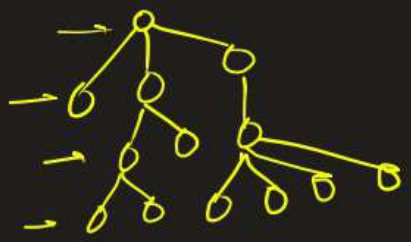


Pre :- Root LC RC — 1 2 4 5 7 8 3 6

Post :- LC RC Root — 4 7 8 5 2 6 3 1

In :- LC Root RC — 4 2 7 5 8 1 6 3

Level :- (BFS) Level wise :- Left to right — 1 2 3 4 5 6 7 8



https://leetcode.com/problems/n-ary-tree-level-order-traversal/

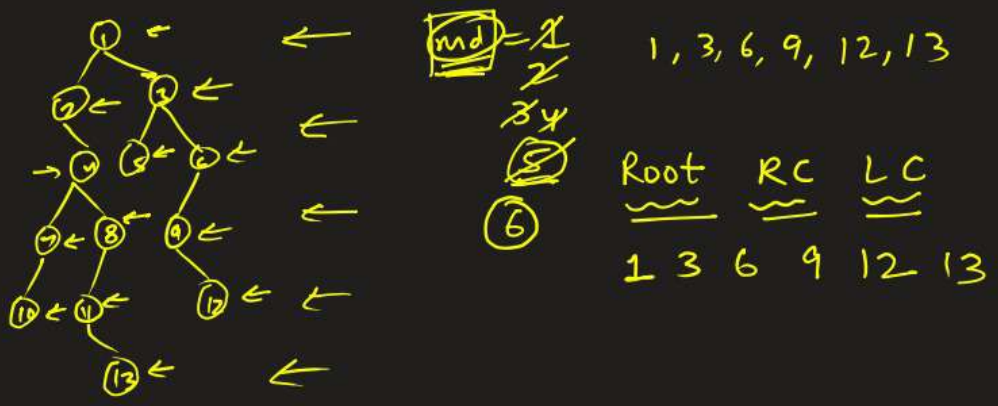
```cpp
class Solution {
public:
    vector<vector<int>> levelOrder(Node* root) {
        queue<Node*> q;
        vector<vector<int>> ans;
        if(!root) return ans;
        q.push(root);
        Node* dummyNode = new Node(-1);
        q.push(dummyNode);
        vector<int> curr;
        while(!q.empty()) {
            Node* tmp = q.front();
            q.pop();
            if(tmp->val != -1) {
                curr.push_back(tmp->val);
                for(auto n: tmp->children)
                    q.push(n);
            }
            else {
                ans.push_back(curr);
                curr.clear();
                if(!q.empty()) q.push(tmp);
            }
        }
        return ans;
    }
};
```

empty

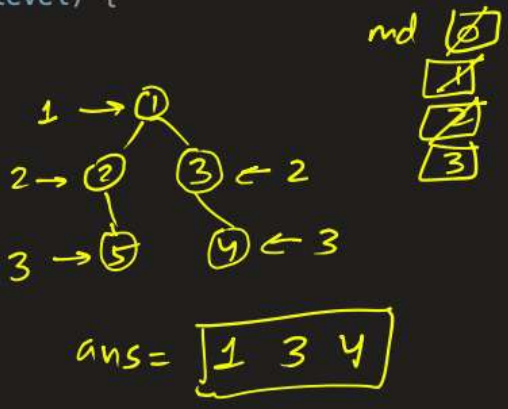
Border case

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

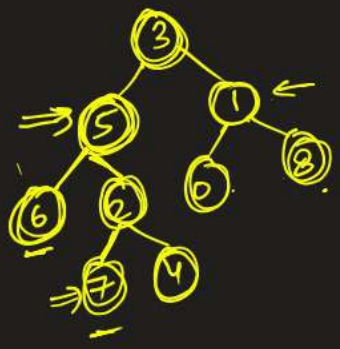
https://leetcode.com/problems/binary-tree-right-side-view/




```cpp
class Solution {
public:
    void trav(TreeNode* root, vector<int> &ans, int &md, int level) {
        if(!root) return;
        //root
        if(md < level) {
            md = level;
            ans.push_back(root->val);
        }
        // rc
        trav(root->right, ans, md, level+1);
        // lc
        trav(root->left, ans, md, level+1);
    }
    vector<int> rightSideView(TreeNode* root) {
        vector<int> ans;
        int md = 0;
        trav(root, ans, md, 1);
        return ans;
    }
};
```



https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/



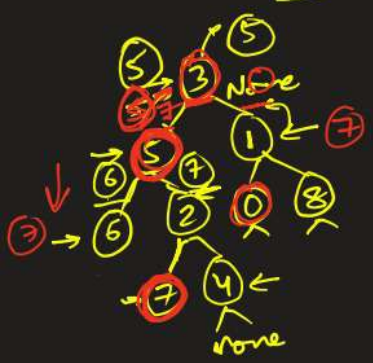
6, 7

3 → left

5 → ans ✓ LCA

- The node which has one element in left subtree & one element in the right subtree is the ~~ans~~ LCA.
- If both elements lie in left subtree then LCA lies in left subtree.
- If both elements lie in right subtree then LCA lies in right subtree

```python
class Solution:
    def lowestCommonAncestor(self, root: 'TreeNode', p: 'TreeNode', q: 'TreeNode') -> 'TreeNode':
        if not root:
            return None
        if root == p or root == q:
            return root
        llca = self.lowestCommonAncestor(root.left, p, q)
        rlca = self.lowestCommonAncestor(root.right, p, q)
    ✗   if llca and rlca:
            return root
    ✗   if llca:
            return llca
    ✓   return rlca
```

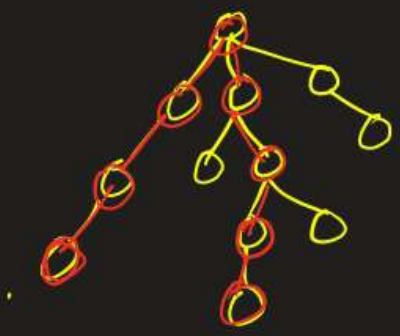


LCA(6,7)

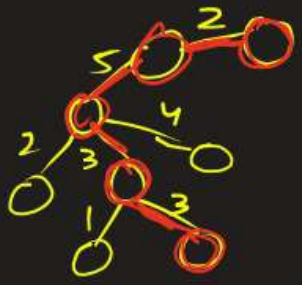
LCA(5,7)

Diameter of a Tree

↳ Length of longest path in the tree



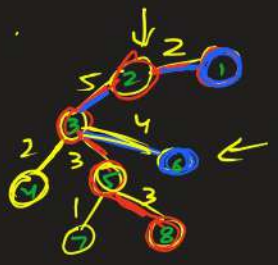
Diameter = (8)



Diameter = (13)

How to find diameter :-

- Perform dfs from any node and find the farthest node → (p)
- Perform dfs from (p) and find the farthest node → (q)
- p — q path will be the diameter of the tree



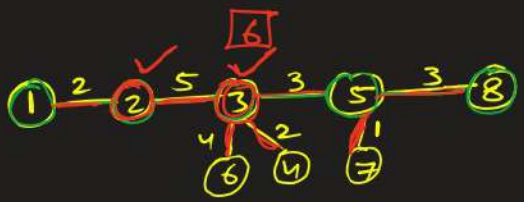
6 → (1)

(1) → (8)

1 - 2 - 3 - 5 - 8

Diameter ✓

https://codeforces.com/contest/1004/problem/E



(k) = 2

- Represent Weighted Trees
- Diameter
- DFS (weighted)