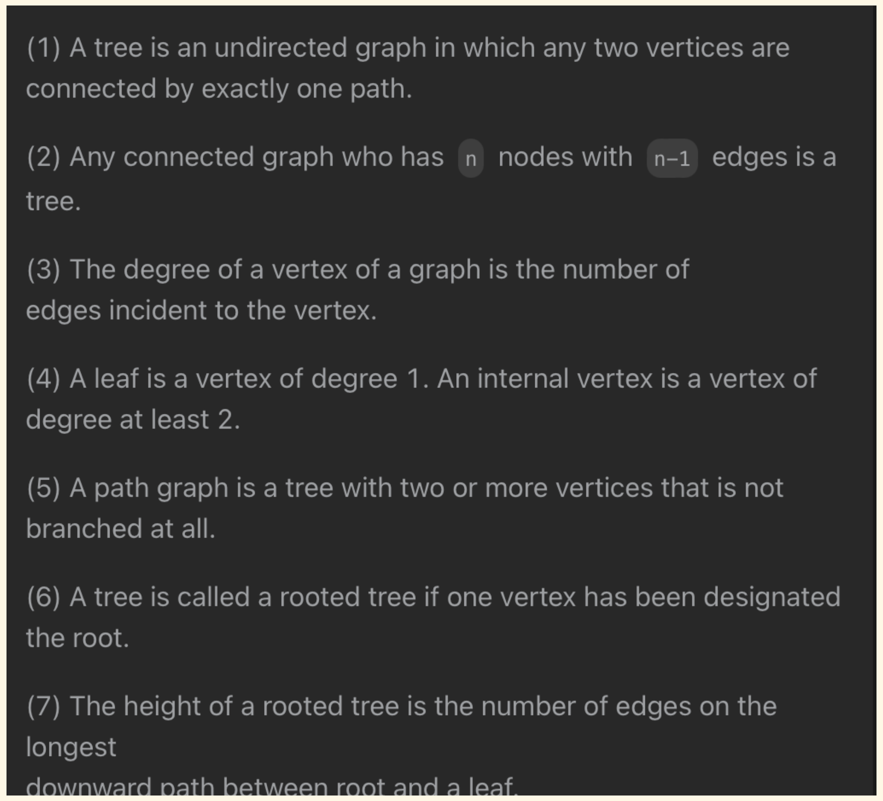
Trees

Thursday, October 26, 2023 5:52 AM



(1) A tree is an undirected graph in which any two vertices are connected by exactly one path.

(2) Any connected graph who has n nodes with n−1 edges is a tree.

(3) The degree of a vertex of a graph is the number of edges incident to the vertex.

(4) A leaf is a vertex of degree 1. An internal vertex is a vertex of degree at least 2.

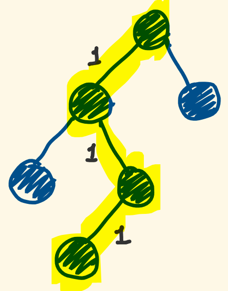
(5) A path graph is a tree with two or more vertices that is not branched at all.

(6) A tree is called a rooted tree if one vertex has been designated the root.

(7) The height of a rooted tree is the number of edges on the longest
downward path between root and a leaf.

Height of a tree

- No. of edges in the longest path from root to leaf.

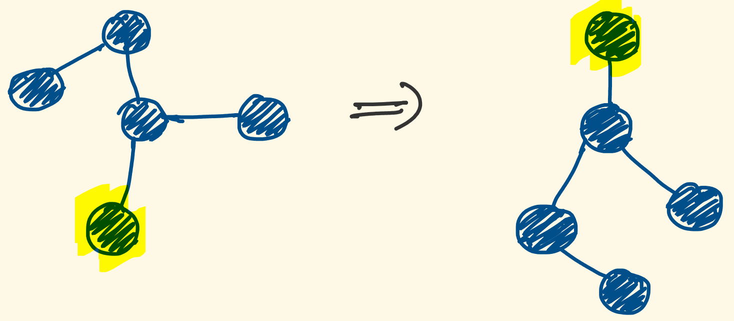


return node? max(Height(left, right))+1 : 0;

Rooting a tree

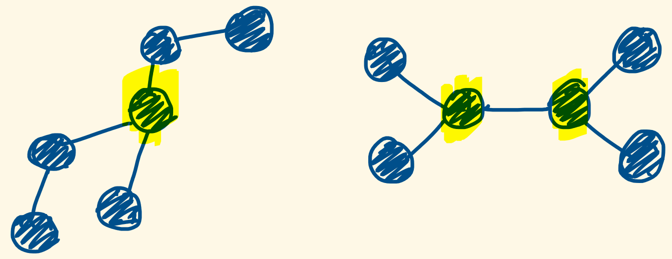
Undirected graph ⟹ Directed graph

- Choosing a root is crucial for a well balanced tree.



Center of a tree

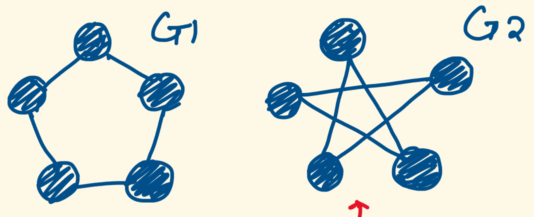
- There can be at most 2 centers.
- Middle of the longest path in a tree



- Remove the outer layers (leafs) of a graph. (like peeling an onion).
  ↳ - find degrees of each nodes.
    - leafs have degree 1.
    - Prune them & update degrees
    - You will arrive at the midpt.

Graph Isomorphism (Not sure if it is NP complete)

- Graphs that are structurally same.



u can unfold it into G1

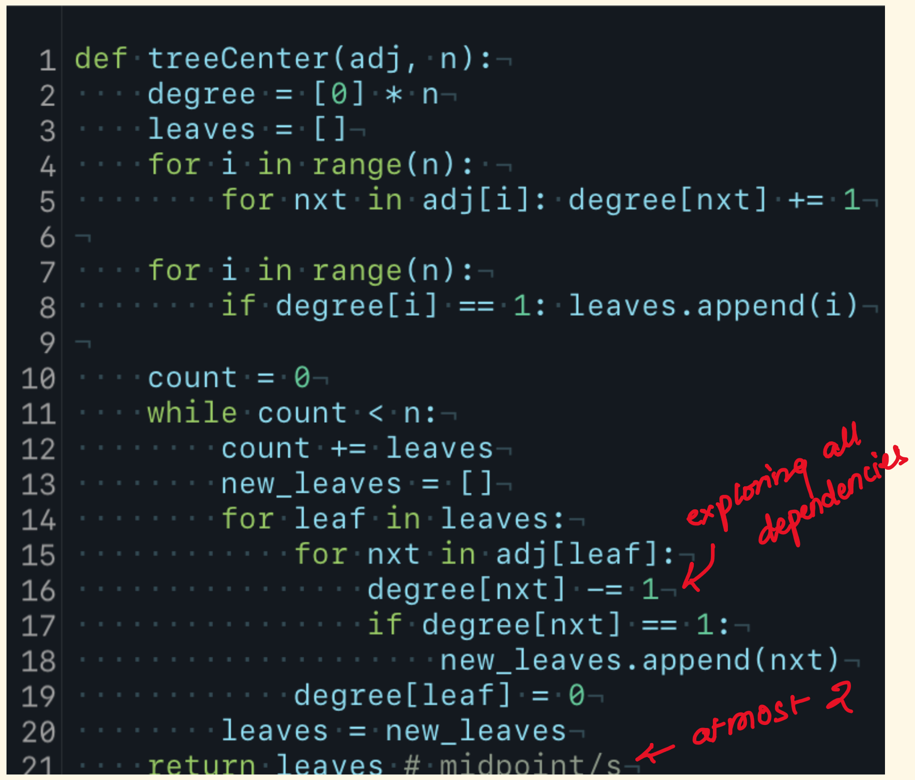
- There are several heuristic hash based algo which give acceptable solutions (but are error prone)
- Another method: we serialize the tree by encoding it into a string & comparing it.

left child: 0
right child: -

Find root to start encoding using center finding algorithm

- AHU algorithm:
  ↳ leaf nodes are assigned '()'
  ↳ Move a layer up & wrap children's encoded val in '( )'
  ↳ Move only after processing all children.

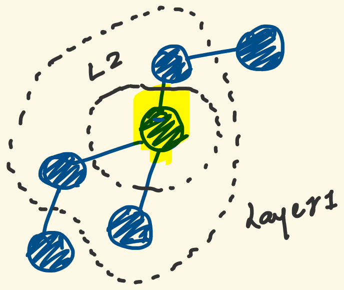
in lexicographical order

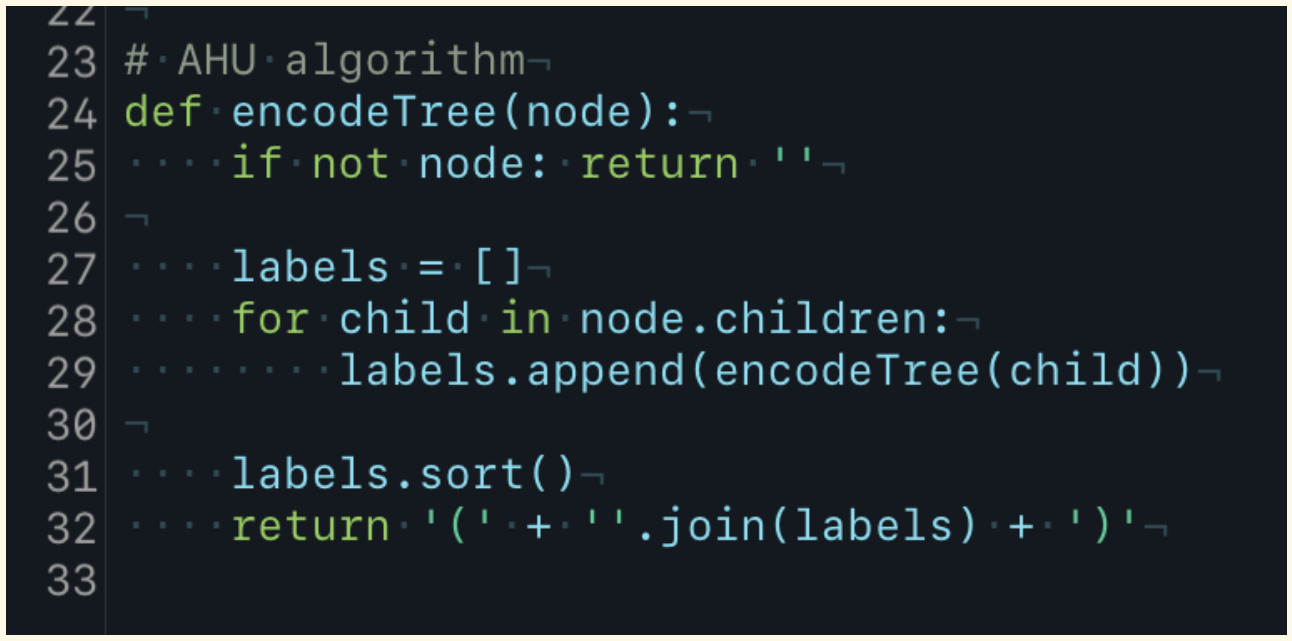


```python
def treeCenter(adj, n):
    degree = [0] * n
    leaves = []
    for i in range(n): 
        for nxt in adj[i]: degree[nxt] += 1

    for i in range(n):
        if degree[i] == 1: leaves.append(i)

    count = 0
    while count < n:
        count += leaves
        new_leaves = []
        for leaf in leaves:
            for nxt in adj[leaf]:
                degree[nxt] -= 1
                if degree[nxt] == 1:
                    new_leaves.append(nxt)
            degree[leaf] = 0
        leaves = new_leaves
    return leaves # midpoint/s
```

exploring all dependencies

atmost 2





```python

# AHU algorithm
def encodeTree(node):
    if not node: return ''

    labels = []
    for child in node.children:
        labels.append(encodeTree(child))

    labels.sort()
    return '(' + ''.join(labels) + ')'

```