UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 14

Tree Structure Analysis

Submitted by:
Balaoro, Judge Wayne B.

Instructor:
Engr. Maria Rizette H. Sayo

November 9, 2025

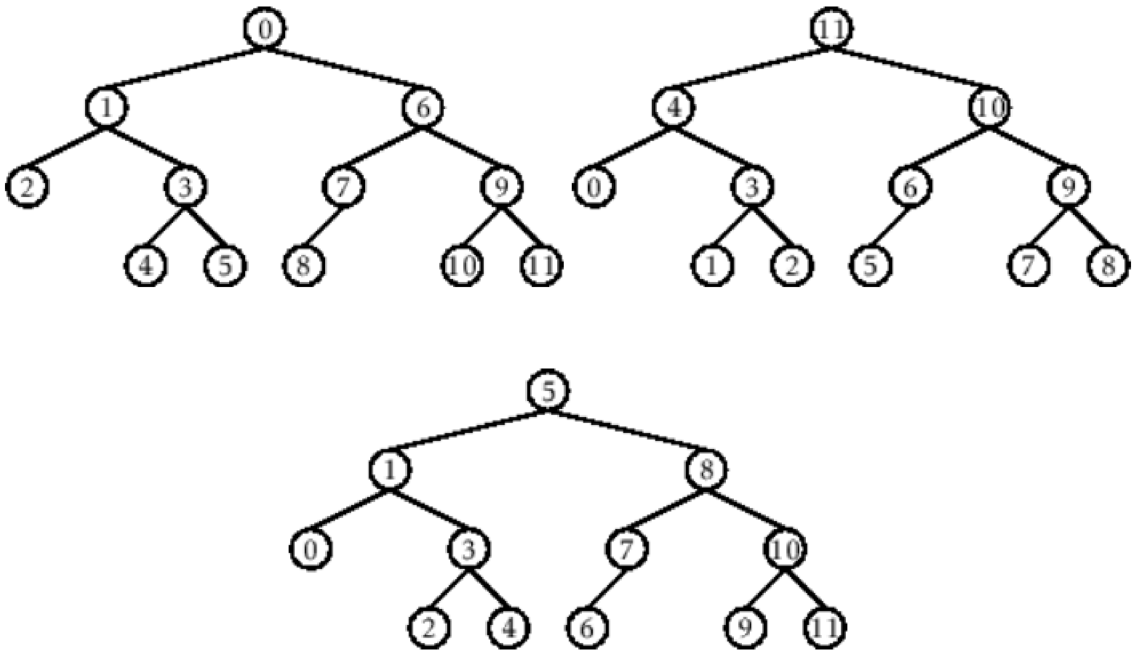
I. Objectives

Introduction

An abstract non-linear data type with a hierarchy-based structure is a tree. It is made up of links connecting nodes (where the data is kept). The root node of a tree data structure is where all other nodes and subtrees are connected to the root.

This laboratory activity aims to implement the principles and techniques in:

- To introduce Tree as Non-linear data structure
- To implement pre-order, in-order, and post-order of a binary tree



- Figure 1. Pre-order, In-order, and Post-order numberings of a binary tree

II. Methods

- Copy and run the Python source codes.
- If there is an algorithm error/s, debug the source codes.
- Save these source codes to your GitHub.
- Show the output

1. Tree Implementation

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.children = []

    def add_child(self, child_node):
        self.children.append(child_node)

    def remove_child(self, child_node):
        self.children = [child for child in self.children if child != child_node]
    def traverse(self):
        nodes = [self]
        while nodes:
            current_node = nodes.pop()
            print(current_node.value)
            nodes.extend(current_node.children)

    def __str__(self, level=0):
        ret = "  " * level + str(self.value) + "\n"
        for child in self.children:
            ret += child.__str__(level + 1)
        return ret

# Create a tree
root = TreeNode("Root")
child1 = TreeNode("Child 1")
child2 = TreeNode("Child 2")
grandchild1 = TreeNode("Grandchild 1")
grandchild2 = TreeNode("Grandchild 2")

root.add_child(child1)
root.add_child(child2)
child1.add_child(grandchild1)
child2.add_child(grandchild2)

print("Tree structure:")
print(root)

print("\nTraversal:")
root.traverse()
```

Questions:

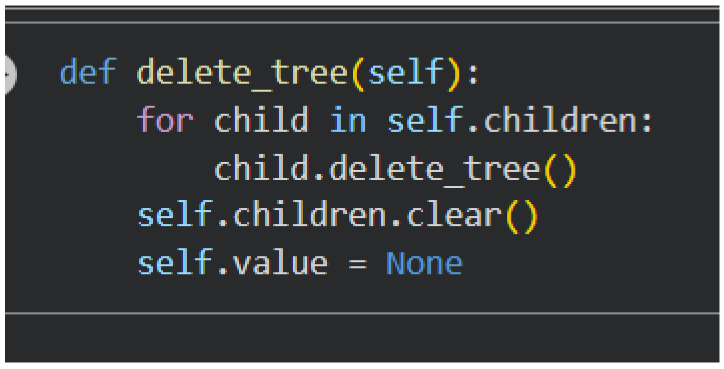
1 What is the main difference between a binary tree and a general tree?
2 In a Binary Search Tree, where would you find the minimum value? Where would you find the maximum value?
3 How does a complete binary tree differ from a full binary tree?
4 What tree traversal method would you use to delete a tree properly? Modify the source codes.

III. Results

1. The main difference between a binary tree and a general tree is that a binary tree limits each node to have at most two children, specifically designated as the left and right child, while a general tree allows a node to have any number of children without restriction.
2. In a Binary Search Tree, the minimum value is always found in the leftmost node by repeatedly moving to the left child starting from the root, while the maximum value is found in the rightmost node by following the right child all the way down.
3. A complete binary tree is a binary tree in which every level is fully filled except possibly the last, and the nodes on the last level are filled from left to right without gaps, while a

full binary tree is a tree in which every node has either exactly two children or no children at all.

4. To properly delete the tree using the given code, the correct traversal method is post-order traversal because it removes all the children first before deleting the parent node, and in this program, we can modify the TreeNode class by adding a method that recursively clears each child's subtree before clearing the current node's own value and children list.



```python
def delete_tree(self):
    for child in self.children:
        child.delete_tree()
    self.children.clear()
    self.value = None
```

Figure 1. delete_tree method

IV. Conclusion

This laboratory activity allowed us to gain hands-on experience in working with tree data structures and understanding how nodes are organized hierarchically. By implementing traversal methods and modifying the code to delete a tree using post-order traversal, we learned how different operations affect memory management and structure behavior. Overall, the activity enhanced our understanding of how trees function as a non-linear data structure and how they are applied in real-world computing problems.

References

[1] Co Arthur O.. “University of Caloocan City Computer Engineering Department Honor Code,” UCC-CpE Departmental Policies, 2020.