UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 14

Tree Structure Analysis

Submitted by:
Villacin, Justine R.

Instructor:
Engr. Maria Rizette H. Sayo

November 03, 2025

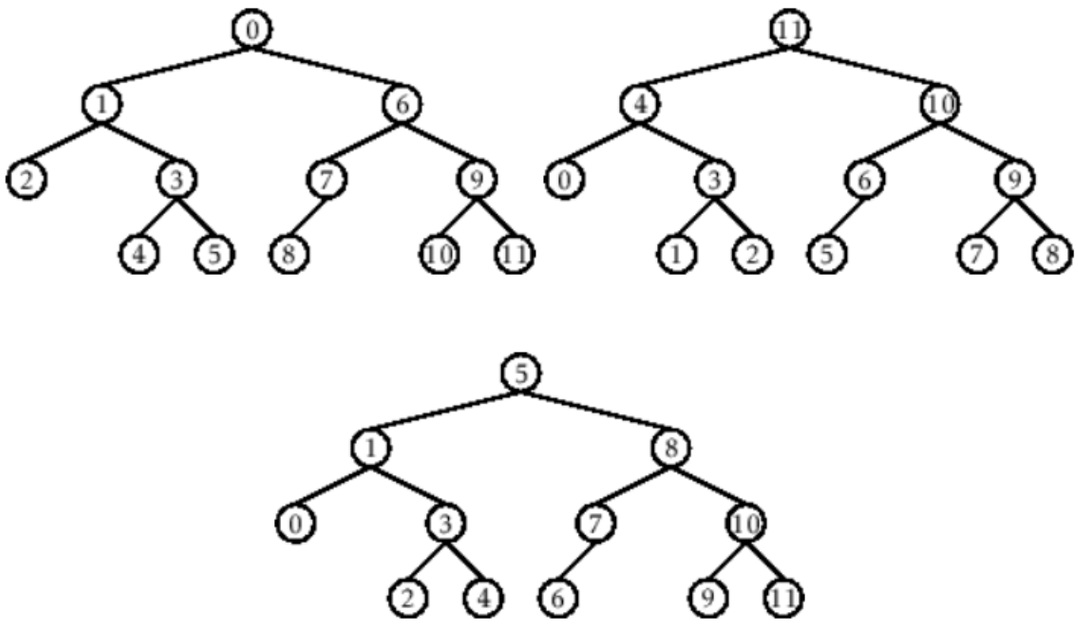
I. Objectives

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

Present the visualized procedures done. Also present the results with corresponding data visualizations such as graphs, charts, tables, or image . Please provide insights, commentaries, or explanations regarding the data. If an explanation requires the support of literature such as academic journals, books, magazines, reports, or web articles please cite and reference them using the IEEE format.

Please take note of the styles on the style ribbon as these would serve as the style format of this laboratory report. The body style is Times New Roman size 12, line spacing: 1.5. Body text should be in Justified alignment, while captions should be center-aligned. Images should be readable and include captions. Please refer to the sample below:

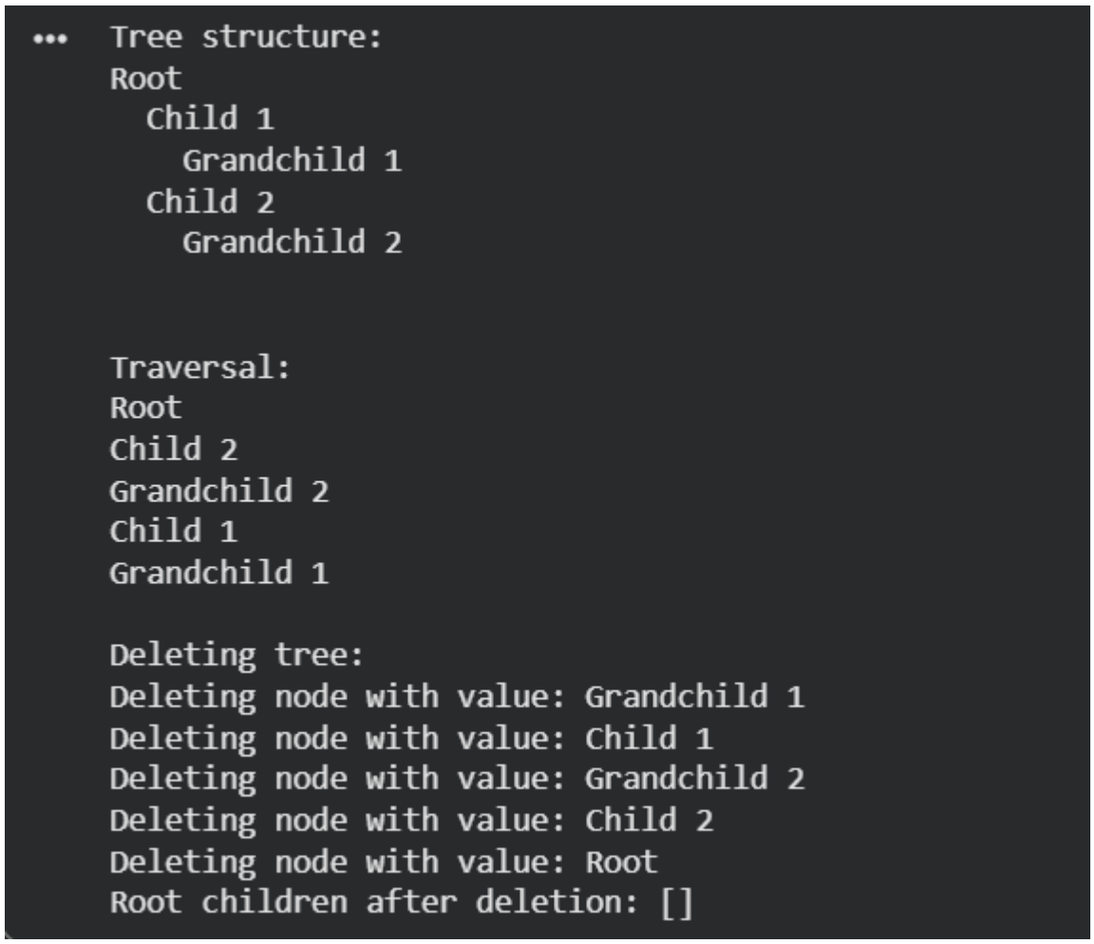


Figure 1 Output

If an image is taken from another literature or intellectual property, please cite them accordingly in the caption. Always keep in mind the Honor Code [1] of our course to prevent failure due to academic dishonesty.

ANSWERS:

1. What is the main difference between a binary tree and a general tree?

The main difference is that a node in a **binary tree** can have at most two children (typically called "left" and "right"), while a node in a **general tree** can have any number of children, from zero to many (GeeksforGeeks, 2024).

2. In a Binary Search Tree, where would you find the minimum value? Where would you find the maximum value?

In a Binary Search Tree (BST), you would find the **minimum value** in the node that is the leftmost node of the entire tree. You would find the **maximum value** in the node that is the rightmost node of the entire tree (University of Washington, 2024).

3. How does a complete binary tree differ from a full binary tree?

A **full binary tree** is a tree where every node has either 0 or 2 children. A **complete binary tree** is a tree where every level is completely filled with nodes, except possibly the last level, which must be filled from left to right (GeeksforGeeks, 2023).

4. What tree traversal method would you use to delete a tree properly? Modify the source codes.

To delete a tree properly, you should use a **post-order traversal**. This method ensures that you delete a node's children before you delete the node itself. If you delete the root node first, you would lose the references to its children, causing a memory leak.

IV. Conclusion

In summary, a general tree allows any number of children per node, unlike a binary tree. In a BST, the minimum and maximum values are found at the leftmost and rightmost nodes. A full binary tree requires nodes to have 0 or 2 children, while a complete binary tree is filled level by level from left to right. Finally, the correct way to delete an entire tree is by using a post-order traversal to remove children before their parent.

References

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

[2] GeeksforGeeks. (2023, October 17). *Full Binary Tree vs. Complete Binary Tree*. https://www.geeksforgeeks.org/difference-between-full-binary-tree-and-complete-binary-tree/

[3] GeeksforGeeks. (2024, April 10). *General Tree (Each node can have arbitrary number of children) Levels*. https://www.geeksforgeeks.org/generic-treesn-array-trees/

[4] University of Washington. (2024). *Binary Search Trees*. CSE 373. https://courses.cs.washington.edu/courses/cse373/24sp/readings/binary-search-trees/