e-ISSN:2582-7219

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH IN SCIENCE, ENGINEERING AND TECHNOLOGY

Volume 7, Issue 7, July 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.521

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 7.521| Monthly, Peer Reviewed & Referred Journal|

| Volume 7, Issue 7, July 2024 |

| DOI:10.15680/IJMRSET.2024.0707238|

Comparative Analysis of Tree Traversal Techniques

Jeevanandham S, Mounika R, Sushma R

Assistant Professor, Department of Computer Science & Applications, The Oxford College of Science, Bengaluru, Karnataka, India

PG student (MSc), Department of Computer Science & Applications, The Oxford College of Science, Bengaluru, Karnataka, India

PG student (MSc), Department of Computer Science & Applications, The Oxford College of Science, Bengaluru, Karnataka, India

ABSTRACT: A data structure is a specialized format for organizing and storing data. General data structure types includes arrays, the file, the records, the tables, and the tree and so on. This paper covers the basic concepts and definition of trees as well as principles, limitations of various traversal techniques such as pre-order, in-order, post-order, Breadth first and depth first. Recursive and non-recursive methods are presented and compared, along with their tree traversal technique. It also address some implementation issues and applications of tree traversals**.**

KEYWORDS: Tree Traversal Techniques – In-Order, Pre-Order, Post-Order, Breadth First, Depth First, Recursive and Non-Recursive.

I. INTRODUCTION

Tree traversal is a critical concept in computer science and data structures that involves visiting each node of a tree. The order in which these nodes are visited helps categorize these traversals. Let us delve deeper into this fascinating topic. This may be done to display all of the elements or to perform an operation on all of the elements. Unlike linear data structures, trees are nonlinear, meaning their traversal differs significantly. With linear data structures, there is only one way to visit each node - starting from the first element and moving in a linear order. However, when it comes to trees, we have multiple traversal methods. In this paper, we will explore these methods in detail. It consists of a central node, structural nodes, and sub-nodes, which are connected via edges. We can also say that tree data structure has roots, branches, and leaves connected.

1.1 Basic Concepts and Definitions:

A tree definition in data structure is a tree a hierarchical data structure. It consists of multiple nodes that's contains the actual data. The in degree of root is, by definition , zero with the exception of the root, all of the nodes in a tree must have an in degree of exactly one; that is , they may have one predecessor all the node in the tree can have zero, one, or more branches.

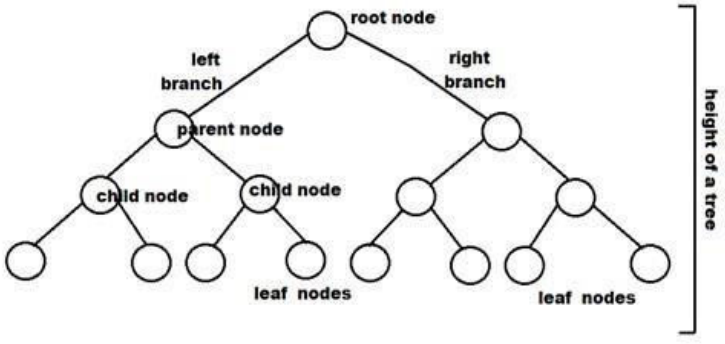

Fig.1.1: Structure of Tree

Root node: The node that is at the top of the hierarchy is called as the root node of the tree.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 7.521| Monthly, Peer Reviewed & Referred Journal|

| Volume 7, Issue 7, July 2024 |

| DOI:10.15680/IJMRSET.2024.0707238|

Child node: A child node in a tree structure is a node that has a parent node connected to it. It is like the branches extending from a main branch in a tree. Each child node is directly linked to its parent node in the tree hierarchy.
Leaf node: A leaf node is a node that does not have any children. It is like the end of a branch in a tree; there are no more nodes extending from it. When traversing a tree, when you reach a leaf node, it means you have reached the end of that particular path in the tree.
Branches: A branch is like a path that connects nodes together. It is similar to a branch in a real tree that extends from the trunk. A branch in a tree represents a sequence of nodes connected in a particular order, starting from the root node and branching out to child nodes.

II. TREE TRAVERSAL TECHNIQUES

DEFINITION: Tree traversal algorithms help us to visit and process all the nodes of the tree. Since tree is not a linear data structure, there are multiple nodes, which we can visit after visiting a certain node. There are multiple tree traversal techniques, which decide the order in which the nodes of the tree are to be visited**.**

Some of the Tree Traversal Techniques:
- pre-order
- In-order
- Post-order
- Breadth first search
- Depth first search

2.1 Pre order: Pre-order traversal will always visit the current node before visiting its children. The root is the first node visited.it follows the left path and visits each node as encounters them. After that, it follows the right paths and still visits the nodes as it encounters them before continuing with the path as shown in the figure 2.1.

Algorithm:
Pre-order (root)
- Process the root node.
- Traverse the left sub-tree, (call pre-order (root -left).
- Traverse the right subtree, (call pre-order (root- right).

Pseudocode:
```
Pre-order (Node)    if (Node != null)
      Process Node
      Pre-order (left_child)
      Pre-order (right_child)
   End if
End pre-order
```

Example:

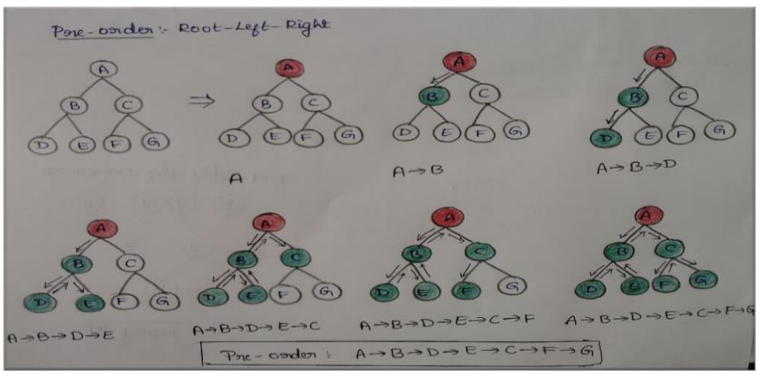

Fig 2.1: Pre-Order

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 7.521| Monthly, Peer Reviewed & Referred Journal|

| Volume 7, Issue 7, July 2024 |

| DOI:10.15680/IJMRSET.2024.0707238|

Applications of Pre-order Tree Traversal: We can create a copy of the tree using pre-order traversal. In addition, pre-order traversal can be used to get the prefix expression of an expression. Pre-order traversal is useful for creating a copy of the tree.

2.2 In-order: In-order traversal is the most common and visit the nodes in ascending order. If it were a binary search tree, this will start with the smallest value at the left most and end at the largest value at the right-most node. In-order traversal is often used for binary search trees to get nodes in sorted order as shown in the figure 2.2.

Algorithm:
In-order (root)

- Traverse the left sub-tree, (call in-order (left_child).
- Process the root node.
- Traverse the right subtree, (call in-order (right_child).

Pseudocode:

```
In-order (Node)    if
(Node != null)
In-order (left_child)        Process the Node
            In-order (right_child)
  End if
End in-order
```

Example:

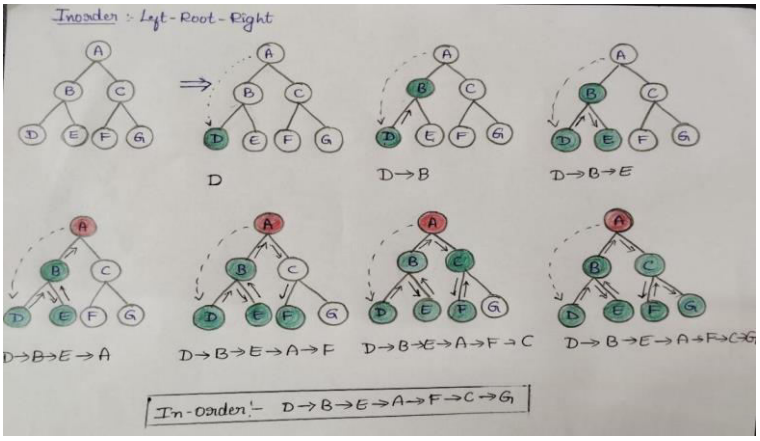

Fig 2.2: In-Order

Application of in-order tree traversal: Binary search tree or BST is a binary tree with a special property — for every node; the value stored in the left child (if it exists) is smaller than the value stored in the current node. Similarly, the value stored in the right child (if it exists) is greater than the value stored in the current node. For a BST, in order tree traversal prints the nodes in non-decreasing order. A variation of in order tree traversal can be used to obtain nodes from BST in non-increasing order.

2.3 Post order: For post-order traversal, you visit nodes children before visiting that node. This means that you will use it when you want to reach the leaves first. You visit the entire left subtree, then the entire right subtree, and then the root node of the subtree as shown in figure 2.3.
Post-order traversal is commonly used in expression trees to evaluate expressions.

Algorithm:
Post-order (root)
Traverse the left sub-tree, (call post-order (root_left).
Traverse the right subtree, (call post-order (root_right).
Visit the root node.

Pseudocode:

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 7.521| Monthly, Peer Reviewed & Referred Journal|

| Volume 7, Issue 7, July 2024 |

| DOI:10.15680/IJMRSET.2024.0707238|

Post-order (Node) if
(Node != null)
Post-order (left_child) Post-order (right_child)
Process Node
End if
End post order

Example:

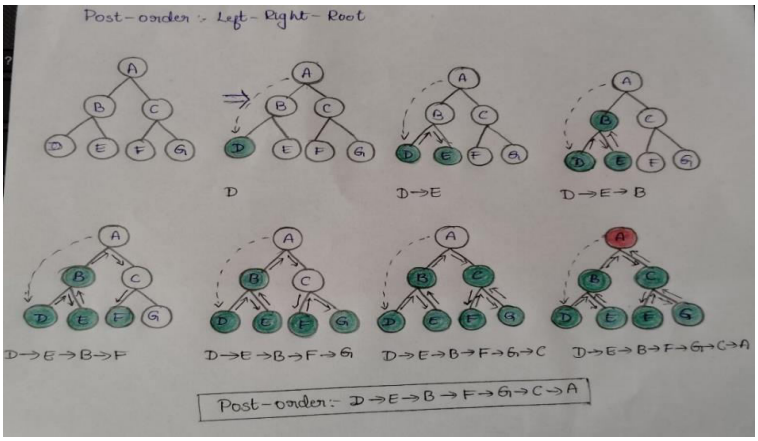

Fig 2.3: Post-Order

Applications of Post-order Tree Traversal

- Post-order traversal is used to delete the tree.
- Post-order traversal is also useful to get the postfix expression of an expression tree.

2.4 Breadth First Search: BFS is a graph/tree traversal algorithm that explores all the vertices in a graph/tree at the current depth before moving on to the vertices at the next depth level. It starts at a specified vertex and visits all its neighbours before moving on to the next level of neighbours as shown in the fig 2.4.
BFS is commonly used in algorithms for pathfinding, connected components, and shortest path problems in graphs.

Algorithm:
Step 1: SET STATUS = 1 (ready state) for each node
Step 2: Enqueue the starting node A and set its STATUS = 2 (waiting state)
Step 3: Repeat Steps 4 and 5 until QUEUE is empty
Step 4: Dequeue a node N. Process it and set its STATUS = 3 (processed state).
Step 5: Enqueue all the neighbours of N that are in the ready state (whose STATUS = 1) and set
Their STATUS = 2 (waiting state)
[END OF LOOP]
Step 6: EXIT

Example:

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 7.521| Monthly, Peer Reviewed & Referred Journal|

| Volume 7, Issue 7, July 2024 |

| DOI:10.15680/IJMRSET.2024.0707238|

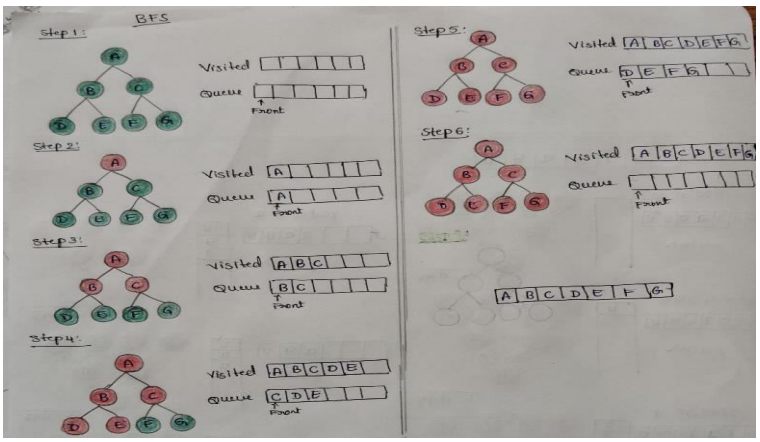

Fig 2.4: BFS

2.5 Depth First Search: DFS is a recursive traversal algorithm for searching all the vertices of a graph or tree data structure. It starts from the first node of graph G and then goes to further vertices until the goal vertex is reached as shown in the figure 2.5

- DFS uses stack as its backend data structure
- Edges that lead to an unvisited node are called discovery edges while the edges that lead to an already visited node are called block edges.

Depth First Search Algorithm

- **Step 1:** STATUS = 1 for each node in Graph G
- **Step 2:** Push the starting node A in the stack. set its STATUS = 2
- **Step 3:** Repeat Steps 4 and 5 until STACK is empty
- **Step 4:** Pop the top node N from the stack. Process it and set its STATUS = 3
- **Step 5:** Push all the neighbours of N with STATUS =1 into the stack and set their STATUS = 2

[END OF LOOP]

- **Step 6:** stop

Example:

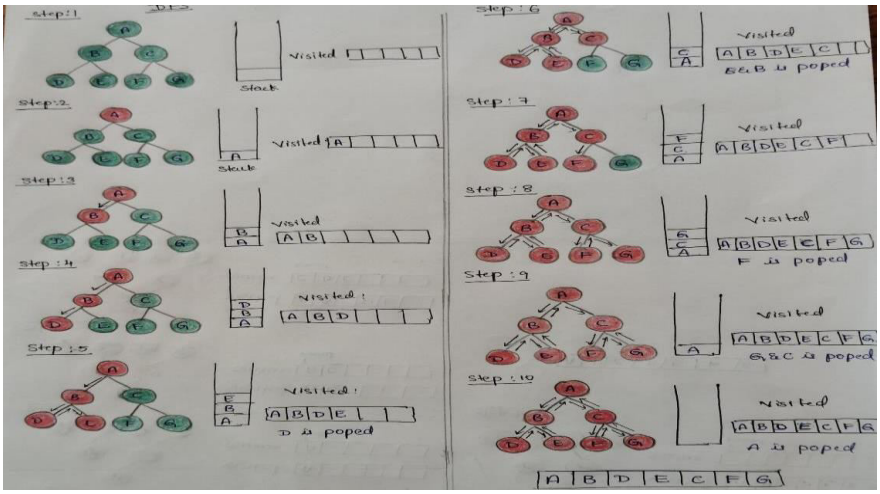

Fig 2.5: DFS

III. RECURSIVE TREE TRAVERSAL

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 7.521| Monthly, Peer Reviewed & Referred Journal|

| Volume 7, Issue 7, July 2024 |

| DOI:10.15680/IJMRSET.2024.0707238|

In recursive tree traversal, you use a recursive function to visit each node in the tree. The function typically follows a pattern where it visits the current node, and then recursively traverses the left and right subtrees. This process continues until all nodes in the tree have been visited. Recursive tree traversal methods are essential for navigating tree structures efficiently. Here are some common recursive tree traversal methods:

1. Pre order Traversal:
- Visit the current node.
- Traverse the left subtree recursively.
- Traverse the right subtree recursively.

2. In order Traversal:
- Traverse the left subtree recursively.
- Visit the current node.
- Traverse the right subtree recursively.

3. Post order Traversal:
- Traverse the left subtree recursively.
- Traverse the right subtree recursively.
- Visit the current node.

These methods are foundational for exploring tree data structures and are widely used in algorithms and applications that involve trees. Each method offers a different order of visiting nodes, enabling various operations and analyses on trees.

IV. NON-RECURSIVE TREE TRAVERSAL

Non-recursive tree traversal refers to traversing a tree data structure without using recursive functions. Instead, it uses iterative methods with data structures like stacks, queues, or Morris traversal (which uses threading). In non-recursive tree traversal, you can use a stack or a queue data structure to keep track of nodes to visit. By iteratively processing nodes without using recursion, you can traverse the tree efficiently. Here are some non-recursive tree traversal methods:

1. Stack-based traversal: Uses a stack to store nodes to visit. Useful for pre-order, in-order, and post-order traversals.
2. Queue-based traversal: Uses a queue to store nodes to visit. Useful for level order traversal (BFS).
3. Morris traversal: Uses threading to traverse the tree without extra space. Useful for in order traversal.
4. Iteration using pointers: Uses pointers to traverse the tree without extra space. Useful for pre order, in order, and post order traversals.

Non-recursive tree traversal is useful when:
- Recursion is not allowed or is too deep.
- You need more control over the traversal process.
- You want to avoid the overhead of function calls.
- You need to traverse large trees without stack overflow errors.

Some benefits of non-recursive tree traversal include:
- Efficient use of memory
- Better control over traversal order
- Avoidance of stack overflow errors
- Improved performance for large trees

However, non-recursive tree traversal can be more complex and harder to understand than recursive traversal.

V. TIME AND SPACE COMPLEXITY

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 7.521| Monthly, Peer Reviewed & Referred Journal|

| Volume 7, Issue 7, July 2024 |

| DOI:10.15680/IJMRSET.2024.0707238|

Traversal technique	Time complexity	Space complexity
Pre-order	O(n)	O(h)
In-order	O(n)	O(h)
Post-order	O(n)	O(h)
Breadth-first	O(n)	O(w)
Depth-first	O(n)	O(h)

n is the number of nodes in the tree h is the height of the tree w is the width of the tree (i.e., the maximum number of nodes at any level).

The time complexity of all the traversal techniques is O (n), as each node is visited exactly once. However, the order in which the nodes are visited varies between the techniques, resulting in different running times for specific use cases. The space complexity of the traversal techniques varies based on whether they use recursion or iteration and how the data is stored during the traversal. The pre-order, in-order, and post-order traversal techniques all use recursion, which means they have a space complexity of O(h), where h is the height of the tree. The breadth-first traversal technique uses a queue data structure, which means it has a space complexity of O (w), where w is the width of the tree. Finally, the depth-first traversal technique can be implemented using either recursion or iteration, and its space complexity can range from O(log n) to O(n), depending on the implementation. The choice of traversal technique depends on the specific use case and the characteristics of the tree being traversed. If memory usage is a concern, depth-first traversal may be preferred, while breadth-first traversal may be preferred for finding the shortest path in an unweighted graph. In-order traversal is useful for traversing binary search trees in sorted order, and pre-order and post-order traversal can be useful for creating or deleting trees, respectively.

5.1 Implementation issues

- Choose the Traversal Type: First, decide whether you want to do in-order, pre-order, or post order traversal. Each has its own order of visiting nodes.
- Start at the Root: Begin your traversal at the root of the tree.
- Recursive Approach: For each type of traversal, you will need to implement a recursive function that visits the nodes in the correct order (left subtree, root, right subtree for in-order, for example).
- Base Case: Make sure your recursive function has a base case to stop the recursion when you reach a leaf node (a node with no children).
- Handling Left and Right Subtrees: Ensure that your recursive calls handle both the left and right subtrees correctly based on the chosen traversal type.
- Testing: Test your implementation with different tree structures to verify that it traverses the nodes in the desired order.

5.2 Applications of tree traversals

- Binary Search Trees (BST): In BSTs, in order traversal visits nodes in sorted order, making it useful for operations like finding the kth smallest element, checking if a tree is a valid BST or printing elements in sorted order. Recommendation Systems: Generate personalized recommendations based on user behaviour.
- Data Mining: Discover patterns, relationships, and insights in large datasets.
- Natural Language Processing: Parse sentences, extract meaning, and generate language models.
- Computer Vision: Analyse images, detect objects, and recognize patterns. Expression Trees: In expression trees, pre order traversal can be used to convert infix expressions to postfix or prefix notations, which are often easier to evaluate.
- File Systems: Tree traversal is used in file systems to navigate directory structures, search for files, or perform operations on files and directories.
- Binary Tree Operations: Traversals are used for various operations on binary trees, such as mirroring a tree, constructing a tree from traversal outputs, or calculating tree height.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 7.521| Monthly, Peer Reviewed & Referred Journal|

| Volume 7, Issue 7, July 2024 |

| DOI:10.15680/IJMRSET.2024.0707238|

- Graph Traversal: Trees can be used to represent graphs, and tree traversal algorithms like BFS and DFS are used to explore and analyse graph structures.
- Parsing and Syntax Analysis: In compilers and interpreters, tree traversal is used for parsing and syntax analysis of programming languages.

VI. CONCLUSION

In short, the tree traversal methods like preorder, in order, post order, BFS, and DFS each have their own strengths and are handy for different tasks:
Pre order is good for creating tree copies and evaluating prefix expressions. In order Useful for sorting binary search trees, evaluating expression trees, and in-order tree printing. Post order: Helpful for deleting tree nodes, evaluating postfix expressions, and freeing tree memory.
BFS is great for finding shortest paths in unweighted graphs, exploring nodes at specific depths, and solving maze puzzles. DFS is Ideal for tasks like topological sorting, identifying graph components, and backtracking.
Each method has its advantages and is suitable for specific scenarios based on the problem requirements. Understanding when to use each traversal method can enhance your ability to work with trees effectively.

REFERENCES

[1].https://faculty.cs.niu.edu/~mcmahon/CS241/Notes/Data_Structures/binary_tree_trav ersals.html
[2] Jain, N., & Jain, R. K. (2017). Comparative Analysis of Tree Traversal Algorithms.
International Journal of Computer Science and Information Technologies, 8(3), 119-123.
[3].Navigating the Forest: A Comprehensive Look at Tree Traversal
Techniques V. Dwaraka Srihith1, L. Rajitha2, P. Blessy2, K. Thriveni2, A. David Donald
[4].https://testbook.com/amp/gate/tree-traversal-notes?hideOpenInAppDialog=true
[5].https://www.interviewkickstart.com/blogs/learn/tree-traversals-inorder-preorderand-postorder
[6].Bharathi, B., P. Shareefa, P. Uma Maheshwari, B. Lahari, A. David Donald, and T. Aditya Sai Srinivas. "Exploring the Possibilities: Reinforcement Learning and AI Innovation."
[7]. https://data-flair.training/blogs/depth-first-search/
[8].https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com