JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

The strategies for traversing trees

Krishna patel, Dr. viral parekh , Dr. Harsha Padheriya

Krishupatel114@gmail.com, viral.parekh@sal.edu.in, hpadheriya@gmail.com

SALITER, ASS. PROF SAL ENGINEERING & TECHNICAL INSTITUTE

ABSTRACT

A tree is a non-linear data structure that can store and retrieve data fast in main memory. It uses a hierarchical structure to show data. The root node is at the top of the tree, and each node has one or more nodes on the left or right side. There is a caregiver node for each node except the root node. To extract data from a tree, several traversal methods can be implemented. Going from one branch to the next is known as tree traversal.

The elements of a tree at the same time we'll look at it in this article multiple ways of tree traversal.

Keywords: Forest, Base station, Iterators, Route

INTRODUCTION

The forest is one of the most crucial pieces in computers. A structure is a tree non-linear data paradigm in which data is stored (such as graphs and trees). A node is the name given to each node in a tree. It is made up of a number of different nodes. Trees are an extremely versatile, robust, and unique data structure for portraying a sequential connection between relatives, corporate personnel, and so on. Trees are a robust, strong, and adaptable data structure, in a tree, data is organized in random order [2]. Specific element content is preserved in a tree node and connected to the next component in a tree-like structure. The highest member of a tree is its roots. Every element in a tree, with the exception of the root node, has a cluster Called head. At least one kid is received by each node. It is called a left child or a right child. A tree has different types of terminology [8]:

Node: Every type of knowledge is symbolized as a node in a tree.

Root node: The tallest point inside a tree is the root node.

Internal/Non- Terminal Node: Non-terminal nodes are those that have at least one child but are not main nodes.

External/Terminal or Leaf Node: Exterior nodes are those that have no offspring.

A degree of a node: The degree of subsets in a particular tree is indicated by the node's degree.

Siblings: Siblings are the offspring nodes of a specific parent node.

Level: A level number is 5allocated to each node in the tree. The root node is always on level zero. The root node's offspring are at level 1, their direct descendants are at level 2, and so on up the tree until the very last node.

Path: It's a series of edges from the source host to the destination network.

Depth/Height: The depth or height of a tree is the number of levels or the greatest level of a tree.

Branch: A limb is the terminal of a tree's subtree.

We visit or traverse the tree when we execute an operation on it for the purpose of retrieving information, which is known as tree traversal. The traversal of a tree employs a variety of algorithms. Preorder traversal, in-order traversal, post-order traversal, and level order traversal are all terms used to describe the process of traversing in a certain order. For tree traversal, the DFS or BFS technique is utilised. Both methods are used to explore a tree in a different way.

Application of Non-Linear Data Structure: Tree

There are many applications of non-linear data structure tree, i.e. medical data, policy proposal, a non-medical data, underwriter data sources, 3D video games, file storage, space partition, specialization of image signature [6]. Also, have the property of image filtering. Min Tree and Max Tree terms are
Used for image filtering [13].

LITERATURE REVIEW

I. **Suri Pushpa, Prasad Vinod [1]:** This paper presents a method for balancing the height of a binary tree. The author presents a technique for retaining the shape of a body. Binary tree is an effective strategy when used correctly. What's more, how do you organise your time and resources? You can manage your space requirements. This essay's author is An algorithm for balancing trees has been proposed. These formulae can be used to. It might be static (global) or dynamic (localised) in nature. There are advantages and disadvantages to each algorithm. For balancing the binary tree, Adel's son Velskii and Landis devised the AVL Tree (dynamic method). Weight-balanced trees, height-balanced trees, and B-trees are other examples that come close to this. Rebalancing is done only when it is necessary in the static algorithm. The vast majority of algorithms are static. The static technique outperforms the dynamic approach when tree sizes are small. The author has provided examples of how to keep the tree balanced after adding or removing nodes. The main objective of this project is to display a tree's height in O(log(n)), with all of the tree's core operations taking O(log(n)) time. The runtime complexity for every node in a tree is O. (N). Years ago Researchers have suggested several different balancing methods. After analysing several types of balancing algorithms, algorithms do not present a notion that balances the tree in less time. As a result, the researcher's prospects for the future look promising.

II. **Navneet Kaur, Deepak Garg [7],** "Depth First Search Algorithms: An Analysis," has provided a strategy to reduce a tree's superfluous research node. In this paper, the author focuses on how to improve search in a non-linear data structure (tree). For enhanced search, time and space management, and memory management, the RHS algorithm and parallel formulation are employed. The DFS algorithm has a backtracking record, which is its primary flaw. That drawback is removed by using the RHS algorithm [13]. This isn't necessary while backtracking on a node. Using the RHS approach or parallel formulation, we may cut search time and improve outcomes. The majority of DFS's poor performance can be due to backtracking.

III. **Jeffrey L. Eppinger [9],** This paper's author has demonstrated how to do a random insertion or deletion operation on a tree. This process is performed on a binary tree using symmetric or asymmetric techniques. This study's author compared insertion and asymmetric deletion algorithms for predicting the internal path of a binary tree. The study's main purpose is to illustrate how we may improve the insertion and asymmetric deletion algorithms to build a binary tree with shorter internal routes after operations. Because the length of the internal path lengthens when we perform an operation on a random binary tree (insertion or asymmetric deletion), the cause-effect on the random binary tree's balance grows. The author, on the other hand, offered a symmetric strategy for addressing the aforementioned problems. A middle ground was discovered using the asymmetric deletion approach. The binary tree is produced at random whenever an insertion or deletion operation is done.

IV. **Parth Patel and Deepak Garg [4],** The author explored a variety of advanced tree data structures. B-tree and Rtree are the two most basic index structures. In this study, the author examined and contrasted the B-tree and T-tree, as well as their applications. Complexity, query type support, data type support, and application are some of the features included in the data structure, and a comparison was made using those features in the advance tree data structure. Changes recommended B-tree and R-tree are used in real-world query and performance improvement. In terms of time and place, some of the trees are more straightforward. B-tree and R-tree, respectively, offer single and multidimensional searches. Based on existing data structures such as hashes, the author devised a novel index structure. It is used by the BR tree. Similar to the hash index structure, a new index structure is provided by merging two existing index structures that have gone through algorithmic alterations.

V. **Niloofar Aghaieabiane, Henk Koppelaar, and Peyman Nasehpour [10**], The study included the development of a binary tree reconstruction method. The majority of methods have been demonstrated using binary tree preorder and inorder traversals. In this research, an Inpos approach is developed for rebuilding a binary tree from inorder and postorder traversal in O(n2) time. In this study, an improved inpos method is described, which takes time and has a space complexity of n. The inpos algorithm also has a feature that tells you how many sub-nodes a parent node has, which can be one, two, or three. The inpos algorithm creates a structure that is built on matrices. As a result, we may obtain information from a tree in linear time.

VI. **R. Kanagavalli1, G. Maheeja [11],** This paper introduced the concept of extracting data from a tree structure. The author expressly mentions IR (Information Retrieval) or IE in this book (Information Extraction). Both IE and IR have distinctive aims. IE is useless, while IR is useful. The purpose of IR is to meet users' requests for information or data from a non-linear data structure. The primary goal of information retrieval is to provide answers to questions posed by the user.
In this study, multiple data structures are employed for information extraction, including storage data structures, process-oriented data structures, and descriptive data structures.

VII. **Kevin Andrusky, Stephen Vurial and Jose Nelson Amaral [3],** The goal of this study is to provide a profiling-based approach for determining how link-based tree data structures are traversed. According to the study, the tree's traversal path is predetermined. It refers to the manner of doing a tree operation in a tree, which might be either breadth or depth first. The profiling-based analysis has a large amount of storage space. When non-instrumented code is utilised, it consumes 7% extra memory. A float point represents the result of a tree's profiling-based analysis. As a result, we may focus on the BFS or DFS valley that is nearest to us. The programmer or researcher will utilise these values to make the data structure better. According to the author, traversing the tree requires no modifications from the coder and just a minor compiler update.

VIII. **Mr. Chandrashekahr S. Khese, Prof. Amrit Priyadarshi [6],** Arrays, vectors, and favourite lists are just a handful of the data structures available. A data structure stores data in a linear order, whereas a tree data structure stores data in a hierarchical way. It denotes a solitary individual.
A node can include one or more nodes. The information is organised in a tree format. are spread about in an unplanned manner. The form of a tree is determined by its surroundings. a kind of knowledge Data that isn't in the form of a random number is known as non-random data.On one hand, the tree's height rises while its form changes. The tree flattens out and stretches out. This study looked on worst-case scenarios in BST trees (binary search tree). The purpose of this project is to show how we can maintain the tree's height while performing operations on it in O(log(n)) time.

IX. **Dr. N. Rama, Justin Sophia. I [12],** The author proposed the DFS and BFS searching algorithms. It is impossible to overestimate the importance of data structure as a mathematical foundation for designing good software. For preordered complete binary tree traversal, the author created the DFS algorithm (FBT). The author's research aims to find out how long DFS and BFS algorithms take to traverse the height of a binary tree. The tree is thoroughly inspected during DFS traversal. The traversal process begins at the root node and progresses to the leaf node. A stack is used to implement it. Backtracking is done using the DFS algorithm. In the vast majority of situations, it becomes stuck in an endless cycle. Level by level, the traversal tree of a BFS is explored. The BFS tree is implemented using a queue. A BFS traversal, unlike a DFS traversal, does not require backtracking. It will never become stuck in a never-ending loop. The author's major goal in this research is to determine how long it would take DFS to discover a target node using the preorder traversal approach.

X. **Hua Li [5],** The binary tree and its characteristics, which is a highly important data structure in computer science, have been examined by the author. The author of this work went over both recursive and nonrecursive algorithms in great depth. Certain adjustments to the present programming would be welcomed by the author. Recursive and non-recursive algorithms are compared. A recursive algorithm is straightforward and straightforward. We were able to quickly examine the binary tree thanks to the recursive call. In the recursive technique, the pointer is necessary for entire traversal; if the pointer is lost, the traversal process is terminated. It is calling itself in a programme, reducing the program's efficiency. When the recursive program's efficiency is low, the author uses a non-recursive traversal approach. It is not the same as a non-recursive algorithm. It's quite tough to create a non-recursive binary tree traversal method. It is used to hold data from tree nodes in a stack. Once the tree operations are completed, the results appear one by one. The author of the paper aims to offer a non-recursive algorithm enhancement in order to increase programme performance and reduce the complexity of the non-recursive approach.

Table 1: An summary of this literature study in a few words

NO	TYPES OF ALGORITHM	TIME COMPLEXITY	SPACE COMPLEXITY	DATA STRUCTURE
1	BFS	O(|n|): where n is the number of nodes	O(|n|)	Queue
2	DFS	O(|n|)	Depends on implementation	Stack
3	DFS(recursive implementation)	O(|n|)	O(|h|): h is the maximal depth of tree	
4	DFS(with iterative solution)	O(|n|)	O(|n|)	Stack
5	Recursive algorithm(Fibonacci sequence)	O(2^n)	O(nm): n is the maximum depth of recursion tree	
6	Component tree computation algorithm (memory access with minimum degree b)	O(b+logbv) (as per memory access)	O(b+logbv)	Stack
7	RHS algorithm for improvement in DFS algorithm[7]	O(N)	O(N)	Stack
8	RHS (in case of complete binary tree)[7]	O(2n-1)	-	Stack
9	Iterative deepening depth first search(IDDFS) algorithm (for well - balanced tree)	O(bd): where b is the branching factor and d is the shallowest solution	O(d)	Stack
10	Martin & Ness's balancing algorithm	O(N)	The stack is used to carrying out the traversal	Stack
11	A Colin day	O(N)	little space is required	Contagious memory
12	Change & Ayengar	O(N)	Additional workspace required= size of tree	Not used stack
13	Stout & warren	O(N)	Only fixed amount of space is required	-
14	In order traversal without recursion	O(N)	-	Stack
15	In order traversal using recursion & iterative algorithm	O(n)	O(n)	Stack
	Preorder traversal (iterative and non recursive)	O(n)	O(n)	Stack
	New modified non recursive algorithm[14]	O(N)	O(NlogN)	-
	Max-tree algorithms.[13]	Shows the table 2		

TABLE 2: MAX TREE ALGORITHM [15]

Algorithm	Time complexity			Auxiliary space requirement		
	Small int	Large int	Generic int	Small int	Large int	Generic int
Berger	O(n log n)	**O(n log n)**	O(n log n)	n+k+O(n)	2n+O(n)	n+O(n)
Berger+rank	O(n α(n))	O(n log log n)	O(n log n)	3n+k+O(n)	4n+O(n)	3n+O(n)
Naiman and couprie	O(n α(n))	O(n log log n)	O(n log n)	5n+k+O(n)	6n+O(n)	5n+O(n)
Salembier et al.	O(nk)	O(nk) ≃ (n^2)	N/A	3k+n+O(n)	2k+n+O(n)	N/A
Nister and stewenius	O(nk)	O(nk) ≃ (n^2)	N/A	2k+2n	2k+2n	N/A
Wilkinson	O(n log n)	O(n log n)	O(n log n)	N+k+O(n)	3n	3n
Salembier non-recursive	O(nk)	O(n log log n)	O(n log n)	N+k+O(n)	3n	3n

Conclusion: A variety of non-linear data structure trees and tree traversal algorithms were examined in this rese0arch. Current tree traversal algorithms have a number of flaws. There is a problem. Something happens when there is a lot of temporal and geographical complexity. as well as the height balance of a0 tree The author made changes to the original. According to a daily algorithm for improved performance in the data structure, there is a requirement for time. The algorithm can do a lot of things. Because it hasn't been built, it's impossible to balance a tree in less time. In Existing procedures of1fer the most potential for future advancement.

REFERENCES

[1] Suri Pushpa, Prasad Vinod, Binary Search Tree Balancing Methods: A critical Study *International Journal of Computer Science and Network Security,* Volume.7 No.8, August 2007

[2] Rubi Dhankhar, Sapna Kamra, Vishal Jangra, Tree Concept in the data structure, *International Journal of Computer Applications,* Volum.1, 2014

[3] Kevin Andrusky, Stephen Curial, and Jose Nelson Amaral. Tree Traversal Orientation Analysis

[4] Parth Patel and Deepak Garg, Comparison of Advance Tree Data Structures, *International Journal of Computer Applications,* Volume No. 2, March 2012

[5] Hua Li, Binary Tree's Recursion Traversal Algorithm and Its Improvements, May *2016*.

[6] Mr. Chandrashekhar S. Khese, Prof. Amirit, Binary Search Tree and Its Applications: Survey, *International Journal on Recent innovation Trends in Computing and Communication,* Volume 3, November 2015

[7] Navneet Kaur, Deepak Garg, Analysis of The Depth First Search Algorithms

[8] Ramesh M. Patelia, Shilpan D. Vyas, *Basic Tree Terminology,* Their Representation and Application 2015.

[9] Jeffrey L. Eppinger, An Empirical Study Of Insertion And Selection In Binary Search Trees, Volume No. 26, September 1983

[10] Niloofar Aghaieabiane, Henk Koppelaar, and Peyman Nasehpour**,** An improved algorithm to reconstruct a binary tree from it's in order and postorder traversals, April 4, 2018

[11] V. R. Kanagavalli1, G. Maheeja, A Study on the Usage of Data Structures In Information Retrieval

[12] Dr. N. Rama, Justin Sophia. I, An Inquisitive Result in DFS Problem of Binary Trees, *International Journal of Scientific Research and Management,* Volume.5, July 2017.

[13] Edwin Clarinet and Thierry Geraud, A Comparative Review of Component Tree Computation Algorithms, Volume. 23, No.9, September 2014

[14] Nitin Arora, Pradeep Kumar Kaushik, satendra Kumar, Iterative Method for Recreating a Binary Tree form its Traversals, *International Journal of Computer Applications,* Volume. 57 No.-11, November 2012.

[15] Bala Raj, Dhingra Sakshi; International Journal of Advance Research, Ideas and Innovations in Technology, (Volume 4, Issue 3) 2018