Quick and Efficient Algorithm to Compare Trees

Sowjanya M[1], Dr. Kiran V.[2]

Dept. of Electronics and Communication[1,2]R.V College of Engineering , Bangalore, India

mail.msowjanya@gmail.com[1]

Abstract:*Tree is an important data structure in computer science and it finds its applications in various fields right from representing hierarchical data to implementing faster search algorithms or even in compiler design. This paper proposes a simple method to visualize the differences and similarities between any two trees. The result of the comparison is known as Difference Tree which provides a graphical notation to intuitively understand the differences.*

Keywords: Trees, Hierarchical Data, Difference Tree

1. INTRODUCTION

In computer science, a tree is an abstract data type that can be used to represent hierarchical data or even to depict different processes. It can be used to store sorted data in a manner that makes it faster to search, insert or delete data. This data structure mainly finds its applications in compiler design, computer networks, representing folder structure, XML, JSON, HTML and YAML data, can be used to build expression parsers and solvers and many more. It can also be used to describe any workflow or represent a particular path. A tree of a set of nodes and edges connected in such a way that no cycles are formed. In other words, it is a graph without any cycles. The edges linking the nodes describe the relationship between the nodes. It consists of a root node and contains subtrees with parent nodes each connecting their set of child nodes. Each node cannot have more than one parent node whereas the parent node can have many child nodes. The root node does not connect to any parent node. Different types of trees can be used for different applications.

A general tree can be used store any hierarchical data. A binary tree is a type of tree where each node consists of only two child nodes, namely the left child and the right child. Such trees are used to build syntax trees in compilers, to implement expression parsers and solvers, to store router tables in routers. A binary search tree is an extension of a binary tree where the value of the left node should be less than or equal to the parent node whereas the value of the right node should be greater than that of the parent node. It can be used to implement simple search algorithms, priority queues or in applications where there is data entering or leaving constantly. AVL trees are self balancing binary trees where the height of the left and the right subtree for a particular node are almost the same. AVL trees are used in situations where frequent insertions are involved and used in Memory management subsystem of the Linux kernel to search memory regions of processes during preemption.B tree is a self-balancing search tree and contains multiple nodes which keep data in sorted order. Each node has 2 or more children and consists

of multiple keys. This can be used in databases to speed up the search or to implement directories in file systems.

2. RELATED WORK

The problem of comparing tree structures is known to be a task often characterised by a particularly high computational complexity. Any attempt to reduce this complexity by considering a tree as a linear structure has generally resulted in a loss of information. Indeed, a comparison of tree structures which considers a tree as a single vector, obviously takes less execution time, but unfortunately has less credibility with respect to the classification task. The hierarchical relationships are thus ignored or suppressed and tree structures then behave like sequential data structures. [1] proposes an approach which relies on two types of traversal algorithm, namely the depth-first traversal and breadth-first traversal. The strategy aims to exploit the advantages of combining the two types of algorithms. [2] presents an analysis of the suitability of various measures of association to determine the similarity of two diagnostic trees using bootstrap simulations. It was found out that three measures of association, Goodman and Kruskal's Lambda, Cohen's Kappa, and Goodman and Kruskal's Gamma each behave differently depending on what is inconsistent between the two trees thus providing both measures for assessing alignment between two trees developed by independent experts as well as identifying the causes of the differences.Taxonomy trees are used in machine learning, information retrieval, bioinformatics, and multi-agent systems for matching as well as matchmaking in e-business, e-marketplaces, and e-learning. [3] introduces a generalized formula to combine matching and missing valueswhen the same sub-tree appears at different positions in a pair of trees. Subsequently, two generalized weighted tree similarity algorithms are proposed. The first algorithm calculates matching and missing values between two taxonomy trees separately and combines them globally. The second algorithm calculates matching and missing values at each level of the two trees and combines them at every level recursively which preserves the structural information between the two trees. The proposed algorithms efficiently use the missing value in similarity computation in order to distinguish among taxonomy trees that have the same matching value but with different miss trees at different positions. [4] introduces a novel visual analytics approach for the comparison of multiple hierarchies focusing on both global and local structures. A new tree comparison score has been elaborated for the identification of interesting patterns. A set of linked hierarchy views was developed showing the results of automatic tree comparison on various levels of details. It is common to classify data in hierarchies, they provide a comprehensible way of understanding big amounts of data. From budgets to organizational charts or even the stock market, trees are everywhere and people find them easy to use. [5] proposes TreeVersity, a framework for comparing tree structures, both by structural changes and by differences in the node attributes.

3. ALGORITHM TO COMPARE TWO TREES

This section proposes an algorithm to compare two trees describing a similar process and identify the differences. This algorithm merges the two trees to be compared and creates difference trees. The difference trees are labelled and coloured trees where nodes or edges coloured in black are common to both the trees whereas the the nodes or edges belonging to a particular tree are in a different colour.

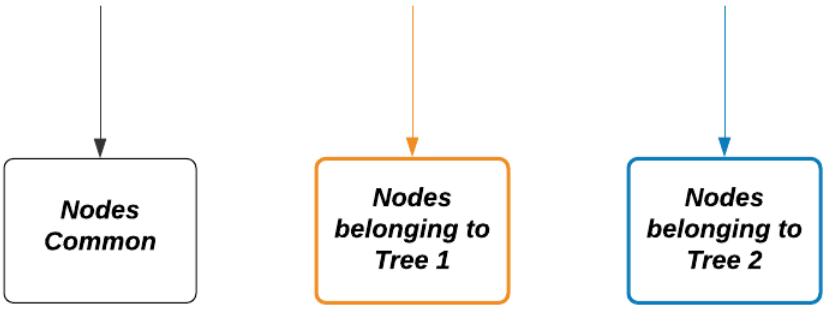

Figure 1. Colouring Convention for the Algorithm

Starting from the root node for both the trees, a level order comparison is done. Every node from each tree is compared to and added to the difference tree with the suitable colour. For example, if there are two trees T1 as shown in Figure 2 and T2 as shown in Figure 3, the difference tree would be a culmination of nodes and edges from both the trees as shown in Figure 4.

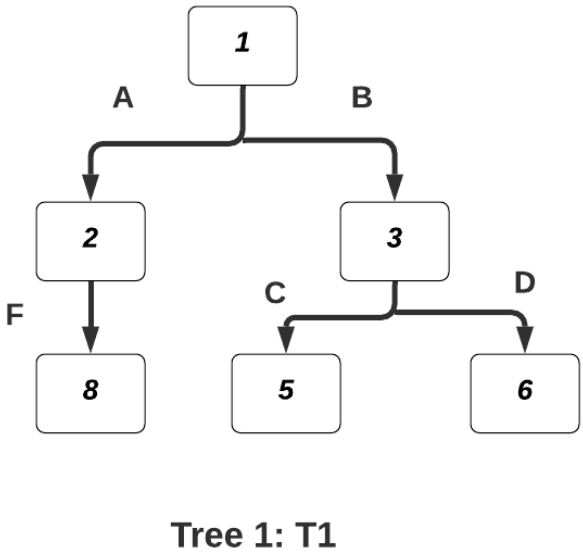

Figure 2. First Tree to be Compared

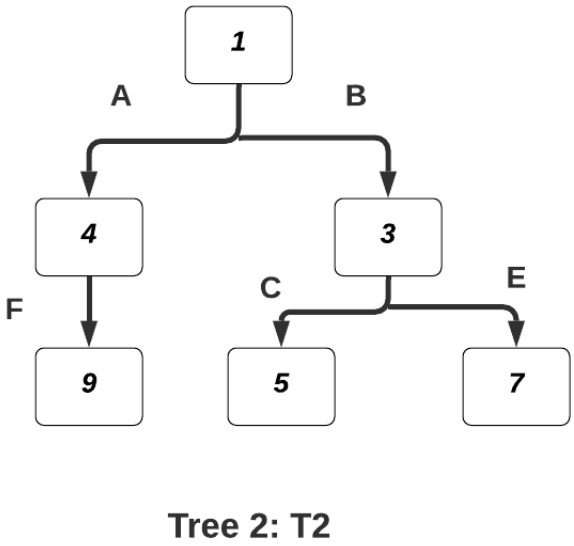

Figure 3. Second Tree to be Compared

To compare both the trees and generate the difference tree, the following algorithm was used:

1. The root node of both the trees are compared and since it is the same in this case, it is added as a black node the output tree.
2. The compared node's child node set and the corresponding edge labels are compared in both the trees.
3. Since node numbered 2 and 4 are not common to both the trees, they are added with the suitable colour scheme.
4. This process is repeated for the other nodes and edges in the tree and the difference tree is obtained.

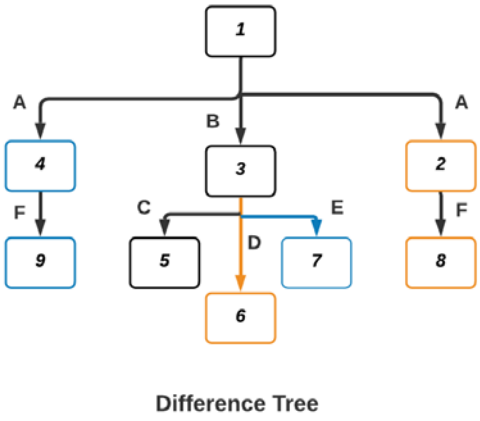

Figure 4. Output Difference Tree

Thus, the difference trees can be used to visualize all the information present in both the trees and at the same time highlight the similarities and differences in a manner which is easy to comprehend.

4. CONCLUSION

The paper has proposed a very simple graphic notation known as a Difference Tree as a way to visualise differences and similarities between two trees. The graphic notation is intuitive and unambiguous and makes it easier to define the differences.This displays the difference and similarity of two compared processes in a clear and user friendly way.
The results to date are promising. With automation tools and large case studies are on the way, the proposed method can be useful for people to study software processes as well as to design new processes. This capability offers considerable benefits for developers to compare hierarchical data or process various models.

REFERENCES

10.1 Journal Articles

[1] Souam, Fatiha & Hadj, Ali. (2016). Efficient similarity measure for comparing tree structures. International Journal of Advanced Intelligence Paradigms. 8. 77. 10.1504/IJAIP.2016.074779.

[2] Sabbaghan, S., Chua, C.E.H. & Gardner, L.A. Statistical measurement of trees' similarity. Qual Quant 54, 781–806 (2020). https://doi.org/10.1007/s11135-019-00957-8.

[3] D., P.K., Rao K., V.G. Generalized weighted tree similarity algorithms for taxonomy trees. EURASIP J. on Info. Security 2016, 12 (2016). https://doi.org/10.1186/s13635-016-0035-2.

10.2 Conference Proceedings

[4] S. Bremm, T. von Landesberger, M. Heß, T. Schreck, P. Weil and K. Hamacherk, "Interactive visual comparison of multiple trees," 2011 IEEE Conference on Visual Analytics Science and Technology (VAST), 2011, pp. 31-40, doi: 10.1109/VAST.2011.6102439.5.

[5] J. A. G. Gómez, A. Buck-Coleman, C. Plaisant and B. Shneiderman, "TreeVersity: Comparing tree structures by topology and node's attributes differences," 2011 IEEE Conference on Visual Analytics Science and Technology (VAST), 2011, pp. 275-276, doi: 10.1109/VAST.2011.610.