Id:15172856 Name: Micheal Hogan 11/05/2020

**Tree structure:**

A tree structure is commonly used in computer science as a data structure. By the definition a tree structure is a finite set of one or more nodes. A node contains data and references to other connecting nodes below it in the structure. One node is specifically assigned the role of being the root. The remaining nodes are partitioned into disjoint sets. These subsets are known as subtrees of the root ( Horowitz, 2008). Tree structures organise nodes in terms of hierarchical relationships in a graphical form. The relations between nodes are shown by connecting lines also known as branches. (Rani & Suman, 2013).

**Purpose Of tree structures in computer science:**

A tree data structure is classified as a descriptive data structure. Along with storing data it also yields the description of the way the data is arranged (Data & Figure, n.d.). When it comes to organising data, tree structures make an excellent alternative to arrays. The reason is that arrays are an example of a storage structure thus they yield no description of how the data should be arranged. Storage structures are linear data structures while descriptive data structures are nonlinear (Rani & Suman, 2013). A disadvantage of using a linear data structure is that they are less efficient than their counter parts. During the search/sort process each node in a linear data structure is traversed in a sequential order until the required node has been found. While a nonlinear structure allows, traversing through the structure in a non-sequential order. Therefore, it requires a greater number of operations to search/sort a linear structure. (Lithmee, 2019) The number of operations needed to traverse a linear structure is proportional to the number of elements in the set. Linear data structures are extremely inefficient when the collection of data in the structure is quite large. The hierarchic structure of a tree is useful for simplifying and speeding up the search and sort process of data sets.

**Binary Trees:**

![A picture containing object, clock

Description automatically generated]()Binary trees are one of the most fundamental data structures in computer science and is used more than any other tree structure. “Mathematically, a binary tree is a connected, undirected, finite graph with no cycles, and no vertex of degree greater than three.” (Morin, 2013)

(Morin, 2013)

**Binary Search trees** (BST) also known as ordered or sorted trees have their data represented in an organised structured based on a searching and sorting property. That allows for optimum data retrieval. The following invariants apply for such property. (Morin, 2013)

* The left child of a parent node should be less or equal to the value of the parent.
* The right child of a parent node should be greater than the value of the parent.

![A picture containing drawing, clock

Description automatically generated]()This organisational property decreases the number of operations required to traverse a BST.

(Morin, 2013)

An **AVL** is a special kind of BTS. It is the first dynamic, self-balancing tree to be invented by Adel'son-Vel'skii and Landis in 1962. Each node in the tree has a balance factor. This factor allows the nodes to organised in a balanced structure. Height balance (v) = Height(Right(v)) - Height(Left(v)). The height of a balanced node in an AVL tree can either be +1 0, or -1. (Tsakalidis, 1985)

The order in which data is inserted into a tree structure directing affects the shape of the tree. The shape of a tree affects how efficient the structure is. If a tree is not balanced i.e. Not symmetrical in shape then an unnecessary amount of comparisons will be needed to locate the required node. An AVL tree doesn’t require the list of elements to be inserted in a set order, for the tree to be balanced. An AVL reduces the number of unnecessary comparisons needed when searching or sorting through the data structure. (Tsakalidis, 1985)

**Applications:**

* A tree structure is used to manage an operating system’s disk file system. Each folder represents a node in the tree. The root node is the root directory in the file system. This structure allows files to be created and deleted quite easily. (Rani & Suman, 2013)
* Trees are used in a large amount of networking applications in computer science. Used regularly with internet protocols (ips). (Rani & Suman, 2013)
* Used in search applications that involves data being added and removed frequently. Example set objects and maps of many language libraries. (Rani & Suman, 2013)
* Compilers use tree data structures for syntax validation. (Rani & Suman, 2013)
* Trees are used to find the shortest hop from one router to another. The tree structure containing the set of potential nodes (routers) is searched using Dijkstra's algorithm to find the node (router) that is the shortest hop from the current node (router). (Rani & Suman, 2013)

**Bibliography:**

Data, I., & Figure, T. (n.d.). *Abstract : This paper tries to throw light in the usage of data structures in the field of information retrieval . Information retrieval is an area of study which is gaining momentum as the need and urge for sharing and exploring information is growing da*. https://arxiv.org/ftp/arxiv/papers/1602/1602.07799.pdf

Horowitz, E (2008). *Fundamentals of Data Structures Algorithm Analysis*. http://dl.pnu-club.com/jozve/arshad/computer/DS E.Horowitz (En)(WWW.PNU-CLUB.COM).pdf

Lithmee, L. (2019) What is the Difference Between Linear and Non Linear Data Structures, Pediaa, available: <https://pediaa.com/what-is-the-difference-between-linear-and-non-linear-data-structures/> [06/05/2020].

Pat Morin, P.M. (2013) Open Data Structures : An Introduction, 31st ed. , Athabasca: Athabasca University Press.

Rani, N. S., & Suman, S. P. (2013). *The Role of Data Structures in Multiple Disciplines of Computer Science- A Review*. *4*(7), 2286–2291. https://www.ijser.org/researchpaper/The-Role-of-Data-Structures-in-Multiple-Disciplines-of-Computer-Science-A-Review.pdf

Tsakalidis, A. K. (1985). AVL-trees for localized search. *Information and Control*, *67*(1–3), 173–194. https://doi.org/10.1016/S0019-9958(85)80034-6