Data Structures and Algorithm Report

Emmanuel Ameh

Data Structures and Algorithms Report: Evaluation of Trees

**Brief History of Trees**

The first recorded application of tree in mathematics dates back to 1857 and this was by Cayley (Kahrimanian, 1954). Great mathematical scientists such as Kirchhoff are also known to have used trees. In the nineteenth century where mathematicians were less passionate about algorithms, trees were not used in the context of data structures (Inverson, 1961). Lambda calculus in the 1930's also dwelt on abstract syntax trees where the variables were the leaf-nodes.

In Computer Science terminology, a tree is a data structure implementing the abstract data type (ADT) (Bentley & Friedman, 1979). It is also a non-linear data structure. It has the form of a hierarchical tree structure. The tree structure has root values and children as subtrees (Perils & Thornton, 1960). Structure-wise, a tree is an assemblage of nodes. It starts at the root node, with each node consisting a specific value. It also references other nodes known as children whilst ensuring non-duplication of references and none pointing to the root of the tree. In comparison to linked lists, queues, stacks, and arrays trees are non-linear data structures (Kahrimanian, 1954). In some instances, trees are empty without nodes, or may consist of a single node referred to as the root, together with one or more subtrees or zero subtrees.

**When, where trees used and similar structures**

A data structure is a method of computer data-storage such that the data can be efficiently used (Bentley & Friedman, 1979). A well-chosen data structure will allow for efficient algorithms to be applied to the data. An algorithm is an instruction-set for solving related problems or for computational purposes (Perils & Thornton, 1960). They are specific instructions for automated reasoning, data processing, and for performing calculations. There are different types of data structures to include linear data structures and non-linear data structures (Meurer et al., 2017). Linear data structures include Arrays and Lists. Examples of non-linear data structures include trees, hash-based structures, and graphs. There are also different types of tree structures to include Binary trees, Heaps, Multiway trees, Space-partitioning trees, and application-specific trees.

Mathematically, an algebraic tree or an unordered tree is also known as an algebraic structure with X as a not-empty set of nodes, whilst parent represents F(X) with node assignment parent (x) (Kahrimanian, 1954). Defined in descriptive set theory, a tree utilizes representations of (X, >=) partial orders as the prefix order between a finite sequence. There are four characterizations of trees namely trees as algebra, as partial algebra, as a prefix order, and as a partial order (Bentley & Friedman, 1979). A fifth definition is the graph-theoretic rooted tree defined as acyclic connected rooted graph. Search trees have their order established by the sibling-associated value. There are exceptions to this rule for example in XML documents JSON file lists and several other structures have independent orders not depending on the node values.

Defining ordered trees as partial algebras corresponds to binary trees, where the ordered tree becomes a structure with X a non-empty nodes-set, and rs, lc as partial maps, called right-sibling and left-child respectively, on X. The following conditions are applicable to the binary structure:

* 1. Partial maps rs and lc are totally disjoint with notation (rs) ∩ (*lc*) = ∅
  2. Inverse of (rs) ∪ (*lc*) is p, a partial map with partial algebra (X, p) is defined as an unordered tree with partial order structure  (*X*, ≤V, ≤S)  represented as (≺S) = (rs), and (≻V) = (lc) ○ (≤S). (Perils & Thornton, 1960)

**Common terminologies used with trees**:

The root is the top-node in the tree. The child is the node, which is connected directly to another node, when moving further away from the root (Kahrimanian, 1954). The child’s converse notion is the parent. Nodes having the same parent are siblings. Descendant is also known as subchild (Meurer et al., 2017). An ancestor is a reachable-node by way of repetitive proceeding from the child to the parent. The leaf is an external node and usually with no children. The branch node is an internal node usually with at least one child. The number of children for a given node is referred to as its degree. By this definition a leaf is therefore degree zero.

The connection between a node and another is the edge. A sequence of edges and nodes that connect a descendant with a node is known as the path (Bentley & Friedman, 1979). The node level is the edge count between the root and the node, plus the numerical value of 1. The node depth is the edge count between the root and the node. The node height is the edge count on the longest path between the node and a descendant leaf (Perils & Thornton, 1960). The tree height is the root node height. A set of n>= O disjoint trees is known as a forest.

**Benefits and limitations of Trees**

Trees are used in the representation of hierarchical data for instance in syntax trees Kahrimanian, 1954). In tree traversal and in binary-search tree, trees are used in data storage, making it possible for efficient searches. Another beneficial application of trees is in the representation of sorted data lists (Kahrimanian, 1954). Trees could also serve as digital images workflow composition for visualization effects. For the simulation of galaxies, trees as data structure can be used beneficially. Trees cannot be used in resolving problems related to linear data structures.

**Benefits and limitations of trees in Python**

Although Python does not have an extensive range of in-built data structures as C++ or Java, due to Python’s dynamic nature a tree can easily be created (Dotson, Seyler, Linke, Gowers, & Beckstein, 2016). The Python standard library does not have tree data structures just as the .NET class library does not have for the reason of reducing locality of memory resulting in missed caches. This is one major setback or limitation for the library, nevertheless it is able to create different types of trees and searches such as in binary-search trees (Meurer et al., 2017; Dotson et al., 2016). In the following Python code, I will create a search tree structure in Python for a family tree of an anonymous individual Anthony.

In this search tree, I will use a Python Class to represent the tree nodes. All the nodes are linked, and the tree is built using a recursive function. Comments are included within the code for easy comprehension of code. After the class is constructed the tree construction follows for children (nodes) and grandchildren (sub-nodes) of “Anthony”, resulting in the subsequent Fig. 1, node tree.

**Python snippet of family tree code for Anthony to include nodes and sub-nodes**

#class for a tree with nodes

Class node (object):

#definition and initializing the class with an object  
 def \_\_init\_\_(self, object, children = [ ]) :

#objectifying the class

self.object = object

#class ready to be loaded with nodes

self.children = children

#tree creation using class defined above

# tree root = “Anthony”

Tree = node(“Anthony”, [

#child 1 = ”Robert”

#grandchildren= “Rodney”, “Rikky” “Rena”

node(“Robert”, [

node(“Rodney”),

node( “Rikky”),

node(”Rena”]),

#child 2 = ”Peter”

#grandchildren= “Paul”, “Paulo”, ”Pavlo”

node(“Peter”, [

node(“Paula”),

node( “Pablo”),

node(”Pavlo”]),

#child3 = ”Sam”

#grandchildren= “Sammy”, “Sally”, ”Serena”

node(“Sam”, [

node(“Sally”),

node( “Serena”),

node(“Sammy”])

])

**Tree generated by Python code above**

Figure 1. *Family Tree generated from Python code above*. (Ameh, 2019).

References

Bentley, J. L., & Friedman, J. H. (1979). Data structures for range searching. *ACM Computing Surveys (CSUR)*, *11*(4), 397-409.

Dotson, D. L., Seyler, S. L., Linke, M., Gowers, R. J., & Beckstein, O. (2016). Datreant: persistent, Pythonic trees for heterogeneous data. In *Proc 15th Python Sci Conf* (pp. 51-56).

Kahrimanian, H. G. (1954). Analytical Differentiation by a Digital Computer, *Symposium on Automatic Programming*, 6–14.

Inverson, K. E. (1961). A programming notation for trees. Research Report R--390, II3M Research Center (Jan. 1961). Retrieved from http://www.dl.acm.org/citation.cfm?id=366828

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., ... & Rathnayake, T. (2017). SymPy: symbolic computing in Python. *PeerJ Computer Science*, *3*, e103.

Perils, A. J. & Thornton, C. (1960). Threaded trees. *Communications for the, Association for Computing Machinery,3*, 195–204.