Trees

Introduction

A tree is recrusively defined as a set of one or more nodes where one node is designated as the root of the tree and all the remaining nodes can be partitioned into non-empty sets each of which is a sub-tree of the root.

Basic Terminology

Root node: The root node `R` is the topmost node in the tree. If `R = NULL`, then it means the tree is empty.

Sub-trees: If the root node `R` is not `NULL`, then the trees T_1, T_2, and T_3 are called the sub-trees of `R`.

Leaf node: A node that has no children is called the leaf node or the terminal node.

Path: A sequence of consecutive edges, is called a path.

Ancestor node: An ancestor of a node is any predecessor node on the path from root to that node. The root node does not have any ancestors.

Descendant node: A descendant node is any successor node on any path from the node to a leaf node. Leaf nodes do not have any descendants.

Level number: Every node in the tree is assigned a level number in such a way that the root node is at level 0, children of the root node are at level number 1. Thus, every node is at one level higher than its parent. So, all child nodes have a level number given by parent's `level number + 1`.

Degree: Degree of a node is equal to the number of children that a node has.

In-degree: In-degree of a node is the number of edges arriving at that node.

Out-degree: Out-degree of a node is the number of edges leacving that node.

Types of Trees

6 types of trees:

- General Trees
- Forests
- Binary Trees
- Binary Search Trees
- Expression Trees
- Tournament Trees

General Trees

General Trees are data structures that store elements hierarchically. The top of the node is the root node and each node, except the root, has a parent. A node in a general tree (excet the leaf nodes) may have zero or more sub-trees. General trees which have 3 sub-trees per node are called ternary trees.

Forest

A forest is a disjoint union of trees. A set of disjoint trees (or forest) is obtained by deleting the root and the edges connecting the root node to nodes at level 1. A forest can also be defined as an ordered set of zero or more general trees. While a general tree must have a root node, a forest on the other hand may be emptybecasue by definition it is a set, and sets can be empty.

Binary Trees

A binary tree us a data structure that is defined as a collection of elements called nodes. In a binary tree, the topmost element is called the root node, and each node has 0, 1, or at the most 2 children. A node that has zero children is called a leaf node or a terminal node. Every node contains a data element, a left pointer which points to the left child, and a right pointer which points to the right child. The root element is pointed by a `'root'` pointer. If `root = NULL`, then it means the tree is empty.

Terminology

Parent: If N is any node in T that has left successor S_1 and right successor S_2, then N is called the parent of S_1 and S_2. Every node other than root node has a parent.

Level number: Every node in the binary tree is assigned a level number. The root node is defined to be at level 0. The left and right child of the root node have a level number 1. Similarly, every node is at one level higher than its parents. So all child nodes are defined to have level number as parent's`level number + 1` .

Degree of a node: It is equal to the number of children that a node has. The degree of a leaf node is zero.

Sibling: All nodes that are at the same level and share the same parent are called siblings.

Leaf node: A node that has no children is called a leaf node or a terminal node.

Similar binary trees: Two binary trees T and T' are said to be similar if both these trees have the same structure.

Copies: Two binary trees T and T' are said to be copies if they have similar structure and if they have same content at the corresponding nodes.

Edge: It is the line connecting a node N to any of its successors. A binary tree of n nodes has exactly $n - 1$ edges because every node except the root node is connected to its parent via an edge.

Path: A sequence of consecutive edges.

Depth: The depth of a node N is given as the length of the path from the root R to the node N. The depth of the root node is zero.

Height of a tree: It is the total number of nodes on the path from the root node to the deepest node in the tree. A tree with only a root node has a height of 1. A binary tree of height h has at least h node and at most $2^h - 1$ nodes. If every level has two nodes then a tree with height h will have at most $2^h - 1$ nodes from level 0, there is only element called the root. The height of a binary tree with n nodes is at least $\log_2(n+1)$ and at most n.

In-degree/out-degree: It is the number of edges arriving at a node. The root node is the only node that has an in-degree equal to zero. Out-degree of a node is the number of edges leaving that node.

Complete Binary Trees

A complete binary tree is a binary tree that satisfies two properties. First, in a complete binary tree, every level, except possibly the last, is completely filled. Second, all nodes appear as far left as possible. In a complete binary tree T_n , there are exactly n nodes and level r of T can have at most 2^r nodes.

The formula for finding left and right child nodes. If K is a parent node, then its left child can be calculated as $2 \times K$ and its right child can be calculated as $2 \times K + 1$ The parent of the node K can be calculated as $|\frac{K}{2}|$ The height of a tree T_n having exactly n nodes is:

$$H_n = |\log_2(n+1)|$$

Extended Binary Trees

A binary tree T is said to be an extended binary tree (or a 2-tree) if each node in the tree has either no child or exactly two children. In an extended binary, nodes having no children are called external nodes.

Representation of Binary Trees in the Memory

In computer memory , a binary tree can be maintained either by using a linked representation or by using a sequential representation.

Linked representation of binary trees

In the linked representation of a binary tree, every node will have three parts:

- the data element
- a pointer to the left node
- a pointer to the right node

In C the binary tree is built with a node type

```
struct node {
       int data;
       struct node *left;
       struct node *right;
}
```

Every binary tree has a pointer `ROOT`, which points to the element (topmost element) of the tree. If `ROOT = NULL`, then the tree is empty.

Sequential representation of binary trees

Sequential representation of trees is done using one-dimensional arrays. It is the simplest technique for memory representation, it is inefficient, it requires a lot of memory. A sequential binary tree follows the following rules:

- A one-dimensional array, called `TREE`, is used to store the elements of tree.

• The root of the tree will be stored in the first location. That is, `TREE[1]` will store the data of the root element.

• The children of a node stored in location K will be stored in locations $(2 \times K)$ and $(2 \times K + 1)$

• The maximum size of the array `TREE` is given $(2^h - 1)$, where h is the height of the tree.

• An empty tree or sub-tree is specified using `NULL`. If `TREE[1]=NULL`, then the tree is empty.

Binary Search Trees

A binary search tree, also known as an ordered binary tree, is a variant of binary tree in which the nodes are arranged in an order.

Expression Trees

Binary trees are used to store algebraic expressions.

Tournament Trees

In a tournament tree (also called a selection tree), each external node represents a player and each internal node represents the winner of the match played between the players represented by its children nodes. Tournament Trees are also called winner trees because they are used to record the winner at each level. We can also represent a loser tree that records the loser at each level.

Traversing a Binary Tree

Traversing a binary tree is the process of visiting each node in the tree exactly once in a systematic way. Unlike linear data structures in which the elements are traversed sequentially, tree is a non-linear data sructure in which the elements can be traversed in many different ways.

Pre-Order Traversal

To traverse a non-empty binary tree in pre-order, the following operations are performed recursively at each ode. The algorithm works by:

(1) Visiting the root node

(2) Traversing the left sub-tree, and finally

(3) Traversing the right sub-tree

In-Order Traversal

To traverse a non-empty binary tree in-order, the following operations are performed recursively at each node. The algorithm works by:

(1) Traversing the left sub-tree

(2) Visiting the left root node, and finally

(3) Traversing the right sub-tree

In-order traversal algorithm is usually used to display the elements of a binary search tree.

Post-Order Traversal

To traverse a non-empty binary tree in post-order, the following operations are performed recursively at each node. The algorithm works by:

(1) Traversing the left sub-tree

(2) Traversing the right sub-tree, and finally

(3) Visiting the root node

Level-Order Traversal

In level-order traversal, all the nodes at a level are accessed before going to the next level. This algorithm is also called the **breadth-first** traversal algorithm.

Constructing a Binary Tree from Traversal Results

We can construct a binary tree if we are given at least two traversal results. The first traversal must be the in-order traversal and the second can be either pre-order or post-order traversal. The in-order traversal result will be used to determine the left and right child nodes, and the pre-order/post-order can be used to determine the root node.

Step 1: Use the pre-order sequence to determine the root node of the tree. The first element would be the root node.

Step 2: Elements on the left side of the root node in the in-order traversal sequence form the left sub-tree of the root node. Similarly, elements on the right side of the root node in the in-order traversal sequence form the right sub-tree of the root node.

Step 3: Recursively select each element from pre-order traversal sequence and create its left and right sub-trees from the in-order traversal sequence. Now consider the in-order traversal and post-order traversal sequences of a given binary tree. Before constructing the binary tree, remember that in post-order traversal the root node is the last node.

Huffman's Tree

Huffman coding is an entropy encoding algorithm that is widely used as a lossless data compression technique. The huffman coding algorithm uses a variable-length code table to encode a source character where the variable-length code table is derived on the basis of estimated probability of occurrence of the source character.

The key idea behind Huffman algorithm is that is encodes the most common characters using shorter strings of bits than those used for less common source characters.

The algorithm works by creating a binary tree of nodes that are stored in an array. A node can be either a leaf node or an internal node. Initially, all the nodes in the tree are at the leaf level and store the source character and its frequency of occurrence (also known as weight).

The internal node is used to store the wight and contains linkes to its child nodes, the external node contains the actual character.

The running time of the algorithm depends on the length of the paths in the tree. The external path length of a binary tree is defined as the sum of all path lengths summed over each path from the root to an external node. The internal path length of a binary tree is defined as the sum of all path lengths summed over each path from the root to an internal node.

Data Coding

When we want to code our data (character) using bits, then we use r bits to code 2^r chracters. This coding scheme has a fixed-length code because every character is being coded using the same number of bits. Variable-length coding is preferred over fixed-length coding because it requires lesser number of bits to encode the same data.

For variable-length encoding, we first build a Huffman tree. First, arrange all the characters in a priority queue in which the character with the lowest frequency of occurrences has the highest priority. Then, create a Huffman tree.

Applications of Trees

- Trees are used to store simple as well as complex data. Here simple means an integer value, character value and complex data means a structure or a record.

- Trees are often used for implementing other types of data structures like hash tables, sets, and maps.

- A self-balancing tree, Red-black tree is used in kernal scheduling, to preempt massively multi-processor computer operating system use.

- Another variation of tree, B-trees are prominently used to store tree strucutres on disc. They are used to index a large number of records.

- B-trees are also used for secondary indexes in databases, where the index facilitates a select operation to answer some range criteria.

- Trees are an important data structure used for compiler construction.

- Trees are also used in database design.

- Trees are used in file system directories.

- Trees are also widely used for information storage and retrieval in symbol tables.

Points to Remember

• A tree is a data structure which is mainly used to store hierarchical data. A tree is recursively defined as collection of one or more nodes where one node is designated as the root of the tree and the remaining nodes cane be partitioned into non-empty sets each of which is a sub-tree of the root.

• In a binary tree, every node has zero, one or at the most two successors. A node that has no successors is called a leaf node or a terminal node. Every node other than root node has a parent.

• The degree of a node is equal to the number of children that a node has. The degree of a leaf node is zero. All nodes that are at the same level and share the same parent are called siblings.

• Two binary trees having similiar structure are said to be copies if they have the content at the corresponding nodes.

• A binary tree of n nodes has exactly $n-1$ edges. The depth of a node N is given as the legnth of the path from the root R to the node N. The depth of the root node is zero.

• A binary tree of height h has at least h nodes and at most $2^h - 1$ nodes.

• The height of a binary tree with n nodes is at least $\log_2(n+1)$ and at most n. In-degree of a node is the number of edges arriving at that node. The root node is the only node that has an in-degree equal to zero. Out-degree of a node is the number of edges leaving that node.

• In a complete binary tree, every level (except possibly the last) is completely filled and nodes appear as far left as possible.

• A binary tree T is said to be an extended binary tree (or a 2-tree) if each node in the tree has either no children or exactly two children.

• Pre-Order traversal is also called depth-first traversal. It is also known as the NLR traversal algorithm (Node-Left-Right) and used to extract a prefix notation from an expression tree. In-Order algorithm is known as the LNR (Left-Node-Right). Post-Order algorithm is also known as the LRN traversal algorithm(Left-Right-Node).

• The Huffman coding algorithm uses a variable-length code table to encode a source character where the variable-length code is derived on the basis of he estimated probability of occurrence of the source character.