ISSN:0975-9646

Basic Tree Terminologies, their Representation and Applications

Ramesh M. Patelia[#1], Shilpan D. Vyas[#2], Parina S. Vyas[#3], Nayan Patel[#4]

[1]*Assistant Professor (HOD of BCA),* [2]*Adhyapak Sahayak,* [3]*Lecturer,* [4]*Assistant Professor*

[1,2]*Anand Mercantile College of Science, Management and Computer Technology, Anand, Gujarat, India.*
[3]*Shree P.M. Patel College of Electronics and Communications, Anand, Gujarat, India.*
[4]*B.N. Patel Institute of Paramedical and Science, Anand, Gujarat, India.*

***Abstract*— In the present study, general tree terminologies are explained with their representation and application in wide fields of computer science. Basically, in the field of computer science we can represent the tree data structure in very simple and convenient way in the form of figure.**

***Keywords*— Root node, leaf node, Indegree, Outdegree, Array, Doubly linked list.**

I. INTRODUCTION

Data Structure is a way of organizing data that not only the data items stored but also their relationship to each other. The data structure is classifieds into mainly two categories. They are

1. Primitive data structure
2. Non-primitive data structure

One of the most important non-primitive data structure is Tree.

II. TREE

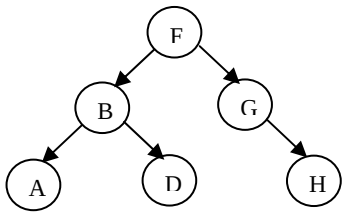
A tree is a data structure that representation hierarchical relationship between data elements.

Outdegree: Total number of leaving vertices is known as outdegree.

Indegree: Total number of entering vertices is known as indegree.

Total degree: The summation of indegree and outdegree is known as total degree.

E.g.



Indegree of a node B is 1
Outdegree of a node B is 2
Total degree of a node B is 3

Root node: A node with indegree zero is known as root node.

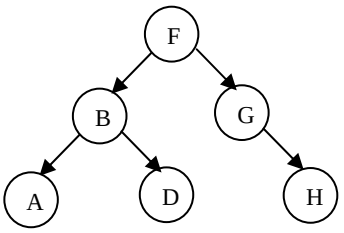
Leaf node: A node with outdegree zero is known as leaf node. A leaf node is also known as terminal node.

Branch node: All nodes except leaf node and root node are known as branch node.

Parent and Child: Parent of a node is the immediate predecessor of a node. All the immediate successor of a parent node is child.

Siblings: The nodes which have the same parents are called siblings.

E.g.

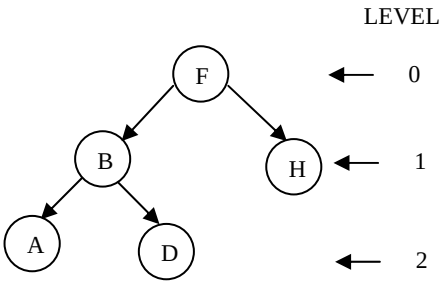


Root node is F
Leaf nodes are A, D & H
Branch nodes are B & G
A node B is parent of child node A & D.
Nodes A and D are siblings.

Level:

The level of any node is the length of its path from the root node. Root node is at level zero.

E.g.

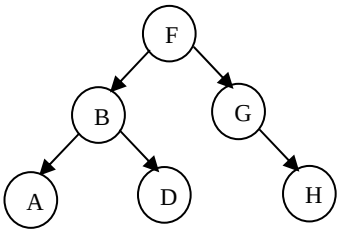


Level of node F is zero
Level of nodes B and H is 1
Level of nodes A and D is 2

A. M-ary tree

A tree with outdegree of every node is less than or equal to m then the tree is called m-ary tree.

E.g. 2-ary tree. (here m=2)

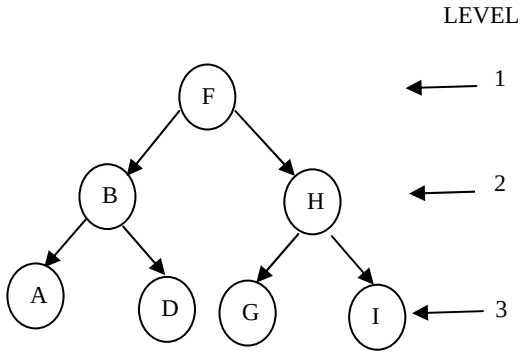


Outdegree of nodes F and B are 2
Outdegree of node G is 1
Outdegree of nodes A, D and H is 0

B. Complete M-ary tree

It is also known as Full m-ary tree.

A m-ary tree in which the total number of nodes at level i is m^{i-1} then it is know as complete m-ary tree.
(Assume that the root node has a level number of 1)
E.g. Complete 2-ary tree. (m=2)

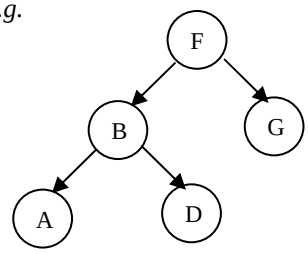


Outdegree of node F, B and H are 2
Outdegree of nodes A, D, G, I are 1
Total number of node at 1 is 1
Total number of node at 2 is 2
Total number of node at 3 is 4

C. Binary tree

A binary tree is a special form of tree. A tree is which the outdegree of each node is exactly two except leaf node is known as binary tree.

E.g.

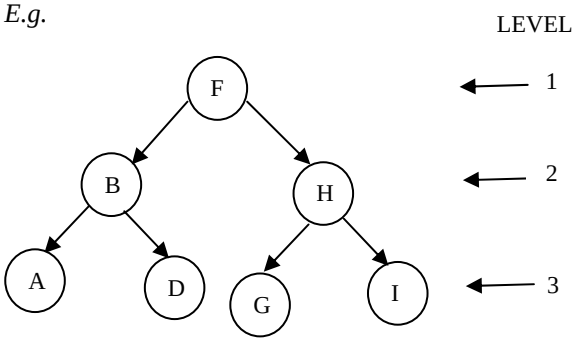


Outdegree of node F, B are 2
Leaf nodes are A, D and G

D. Complete Binary tree

A binary tree is said to be complete binary tree if all the leaf nodes are at same level. It is also known as Full binary tree.

E.g.

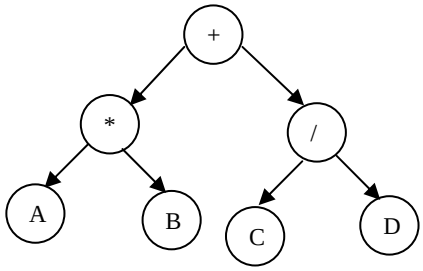


Outdegree of node F, B and H are 2.
Leaf nodes are A, D, G, I and all are at level 2.

E. Expression tree

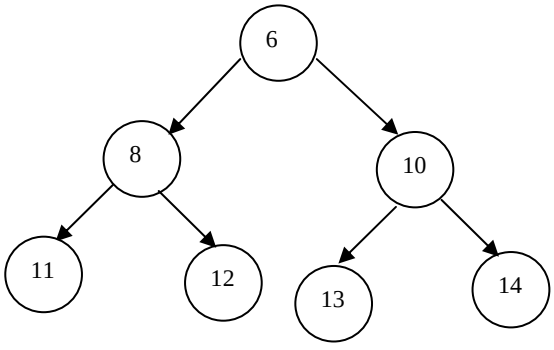
An expression tree is a binary tree which stores an arithmetic expression. The leaves of an expression tree are operands and all internal nodes are operators.

Figure shows an expression tree for the arithmetic expression (A*B) + (C/D).



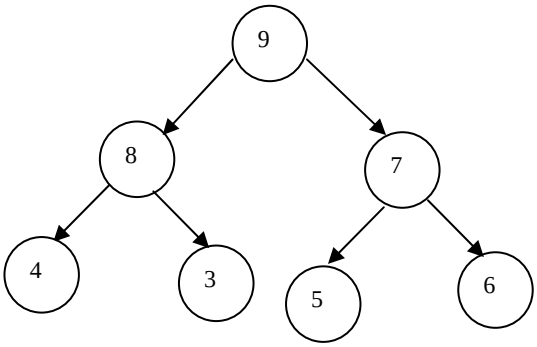
F. Heap tree

A heap is an ordered balance binary tree I which the value of the node at the root of any subtree is less than or equal to the value of either of its children. A heap satisfying the above definition is called a Min heap.
Figure: Min heap



A heap is an ordered balance binary tree I which the value of the node at the root of any subtree is greater than or equal to the value of either of its children. A heap satisfying the above definition is called a Max heap.

Figure: Max heap



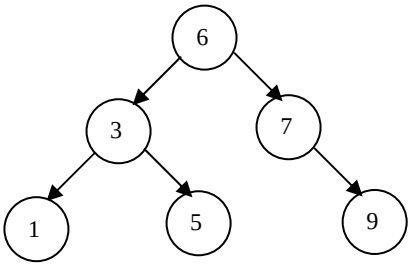
Application of heap trees

There are two main applications of heap trees known: (a) sorting and (b) priority queue implementation.

G. Binary search tree

A binary search tree is a binary tree that is either empty or in which each node contains a key that satisfies the following conditions:

- All keys in the left subtree of the root precede the key in the root.
- The key in the root precedes all keys in its right subtree.
- The left and right subtree of the root are again binary search tree.



III. STORAGE REPRESENTAION OF BINARY TREE

There are two types of storage representation of binary tree.

1. Linear storage representation
2. Linked storage representation

Linear storage representation

A linear storage representation can be done by one dimensional array. The size of one dimensional array can be calculated by using following formula.

Formula:

Array size = $2^{d+1} - 1$

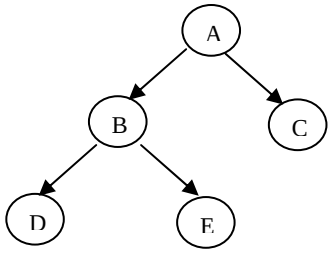
Where d is the depth of the tree. (Depth means maximum levels of the tree)

All the elements of binary tree can be stored into one-dimensional array by following rules.

1. Root node is stored at position 1.
2. If the node is stored at position N then its left child node is stored at position 2*N and its right child node is at 2*N+1.

Example:

Consider the following incomplete binary tree:



Array size = $2^{d+1} - 1$

$= 2^{2+1} - 1$

$= 2^3 - 1$

$= 8 - 1$

$= 7$

All the elements are stored in array as follow:

1	2	3	4	5	6	7
A	B	C	D	E	-	-

Advantages:

- It is simple and easy to implement.
- It possible to find parent node for any child node. (Position of a parent node = position of child node / 2)

Disadvantages:

- There is wastage of memory (some of the memory locations are left empty).
- Insertion and deletion of a node required a lots of data movement.

Linked representation:

A linked storage representation can be done by doubly linked list.

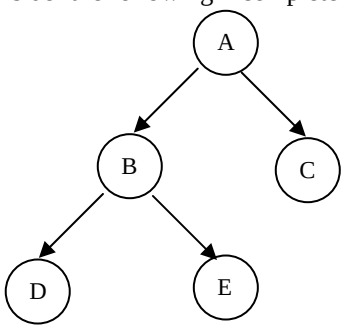
LPTR	INFO	RPTR

LPTR – pointing to the left child node
RPTR – pointing to the right child node
INFO – actual value

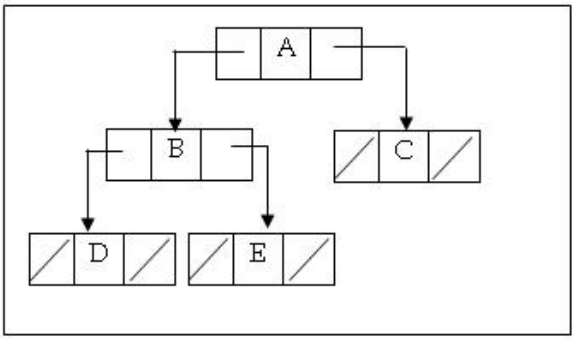
In linked representation, a node has two pointers fields and one information field. Each of pointer field contains the address of left or right child node. An information field contains the actual data about node. When node has no child the corresponding pointer fields are null.

Example:

Consider the following incomplete binary tree:



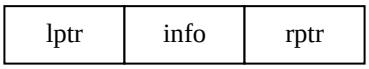
In linked representation, above tree can be represented as



In a linked list format, a tree data structure can be defined as follows:

```
struct  node
{
        int info;
        struct node  *lptr;
        struct node *rptr;
};
```

Where, info is the information part of the node, *lptr is a pointer to left child of the node and *rptr is a pointer to right child of the node. Therefore, a node can be pictorially represented as follows:



Node representation

Advantages:

- It is most efficient.

Disadvantages:

- Wastage of memory space in null pointers.
- It is difficult top find parent node from the given child node.
- It is difficult to implement in old languages (such as FORTRAN, BASIC) that do not support dynamic storage techniques.

Application of tree:

The following are the application of tree:

- The manipulation of arithmetic expression
- Symbol table construction
- Syntax analysis
- Directory of the file system

IV. CONCLUSION

In this paper, we provide the descriptions of several tree terminologies and different type of tree with their explanation and figure. The main aim of this paper is that to present the importance of tree data structure in the field of computer science.

REFERENCES

[1] Tremblay J. & Sorenson P. G. : An Introduction to Data Structures with Applications, 2nd Edition, McGraw-Hill International Edition, 1987.

[2] Singh Bhagat & Naps Thomas : Introduction to Data Structures, Tata McGraw-Hill Publishing Co. Ltd.,1985.

[3] D. Samanta : Classic Data Structure, Prentice-Hall of India