ASSIGNMENT

NAME : Sanjay T

REG NO : RTC2024MCA027

SUBJECT : DATA STRUCTURE

DATE : 17/12/24

CLASS : MCA--B

Trees :-

Binary Tree :-

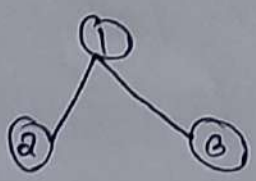
A binary tree is a tree data structure in which each node has at most two children referred to as the child and right child.

Characteristics :-

* Each node can have at most two children
* The left child is less than the Parent node.
* The right child is greater that the Parent node.

Example :

Eg : 1 2 3



Binary Search tree :-

* A binary search tree with an additional Property for any node.

* All value in the left subtree are less than the node's value

* And all values in the right subtree are greater.

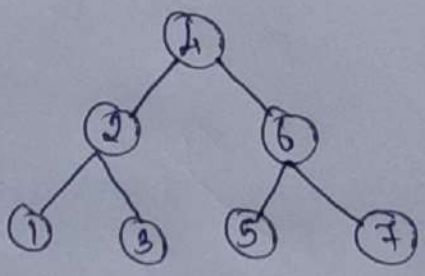
Characteristics :-

* Efficient for Searching, insertion and deletion.

* Maintain a Sorted order of elements.

Example :-

Eg: 4 2 6 1 3 5 7



Avl tree :-

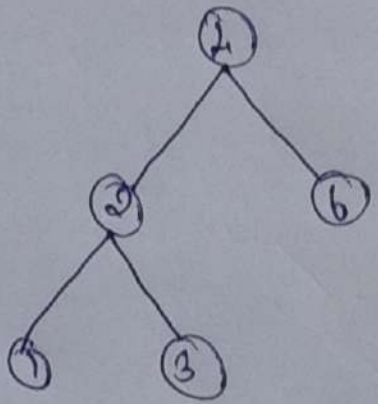
An Avl tree is self balancing binary search tree where the difference in heights of left and right subtree cannot be more than one for all nodes.

Characteristics :-

Guarantees O(logn) time complexity for search insert, and delete option.

Balances itself after every insertion and deletion

Example :



Heap tree :-

A heap tree is a special binary tree - based structure that satisfies the heap property.

Characteristics :-

Max - Heap :

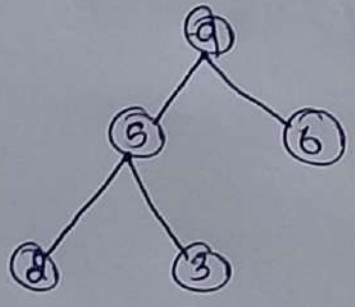
Parent Nodes are always greater than or equal to their child Nodes.

Min - Heap :

Parent Nodes are Always less than or equal to their child Nodes.

Commonly used for priority queues.

Eg :



Tree Representations:

Linked Representation:

```
#
Struct Node {
        int data;
        Struct Node * left;
        Struct Node * right;
};

Struct Node * CreateNode (int data)
{
    Struct Node * New Node = (Struct Node *) malloc (size of (Struct
                                                        Node));
    NewNode -> data = data;
    newNode -> left = NULL;
    NewNode -> right = Null;
    return newNode;
}
```

```
int main()
{
    struct Node * root = CreateNode(1);
    root -> left = CreateNode(2);
    root -> right = CreateNode(3);
    return 0;
}
```

Array representation :-

```
int tree[] = {1,2,3,4,5,6,7}
int leftChildIndex = 2*i+1;
int rightChildIndex = 2*i+2;
```

Parent Representation:

```
int Parent[] = {-1,0,0,1,1,2,2}
```

The Root node have a Parent value of -1.