**Binary tree:**

**Introduction:**

* A binary tree can be defined as a finite set of elements, which can either be empty or have at most two children.
* A binary tree can either be empty or be divided into three disjoint subsets.
* The first subset constitutes of a single element called the root.
* The other two subsets are namely, left child and a right child. These subsets can be empty.
* Each entity that holds a value or data is termed as a node.

**Properties:**

* The order of a binary tree is ‘2’.
* In a binary tree, each node can have at most 2 children.
* A binary tree can be empty.
* A binary tree does not permit duplicate value.
* When constructing a binary tree, if an element has a value less than that of the parent node, it is placed on the left side of the parent node.
* If an element has a value greater value than the parent node, it is placed on the right side of the parent node.
* Height of a binary tree is equal to the height of the root node.
* Height of a particular node, under consideration, is the number of edges extending from that node to the deepest leaf.
* Depth of a particular node, under consideration, is the number of edges extending from the root node till that node.

**Operations:**

* Traversal .
* Insertion
* Deletion.

**Binary trees (Types):**

1. *Full binary tree*

A binary tree is said to be a Full binary tree if all nodes except the leaf nodes have either 0 or 2 children. In other words the degree of such tree can either be 0 or 2.

1. *Complete binary tree.*

A binary tree is said to be a Complete binary tree if all the levels are completely filled except possibly the last level; and the last level has all the keys towards left.

1. *Perfect binary tree*

A binary tree is said to be a Perfect binary tree if all internal nodes have exactly 2 children.

1. *Balanced binary tree*

A binary tree is said to be a Balanced binary tree if, for each node it holds that the number of inner nodes in the left subtree and the number of inner nodes in the right subtree differ by at most 1

**Comparison between Full binary tree and Complete binary tree:**

* In a Full binary tree, each node has either zero or 2 children. Its degree can not be 1 .
* While, in a Complete binary tree, every level is completely filled except the last level and the nodes on the last level are oriented towards extreme left.

**Representation of a binary tree in memory:**

In a binary tree, each node consists of a data field and since binary tree is of recursive nature, it also contains a pointer.

Hence the representation of a binary tree can be extended to two ways :

1. Sequential representation(Arrays).
2. Dynamic node representation(Linked lists).
3. **Sequential representation:**

Let us consider a complete binary tree **BT** which is represented using an array **T** keeping in mind the following points:

* 1. The root node **N** of the tree **BT** is placed on index **T[1]** and all other nodes of the tree are placed recursively.
	2. If a node is placed on **T[k]**,
		1. The left child of the node is found on the position **T[2\*k]**.
		2. The right child of the node is found on the position **T[(2\*k) + 1]**.
		3. The parent node of the node is found on the position **T[k/2].**

**For example: [ F , D , J , B , H , A , K , E , G , C , L ]**

The tree for the given data will be as follows:

The array representation of the tree structure above is shown as follows:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| -1 | F | D | J | B | E | H | K | A | C | -1 | -1 | G | -1 | -1 | L |

1. **Dynamic node representation:**

Binary trees can be represented using linked lists. Each node contains the address of the left and the right child. The leaf nodes contain a **NULL** value in its link field since it doesn’t have a left or a right child.

Let us consider a binary tree **BT** consisting of a node **N** corresponding to a location **k** which is represented as linked list **L** keeping in mind the following points:

* 1. **Left[k]** points to the location of the left child of the node.
	2. **Right[k]** points to the location of the right child of the node.
	3. **Data[k]** consists of the data of the node.

**For example: [ F , D , J , B , E , H , K ]**

The linked list representation of the given tree structure will be as follows:

**Applications:**

* Binary trees are data structures that are used to represent hierarchies.
* They are flexible and powerful data structures.
* They provide efficient insertion and deletion operations.