Height of tree: 4

* Combition of nodes through edges(used to represent

Hierarchical data)

**TREE**

0 – Depth of node

DEPTH

Edge



Edge

**Root**

Degree of tree: 3

Siblings

1

Degree: 2

2

3

4

Leaf

Degree = 0

**Root**: Node which do not have any parent node.

**Parent**: Any node which have a node below is known as parent.

**Child**: Any node which have a node above is known as child.

**Siblings**: Nodes with same parent.

**Leaf/ External Node**: Any node which do not have any child is known as LEAF or External Node

**Internal Node**: Any node which have a child is known as Internal node

**Depth**: No. of edges above a node is known as its depth.

**Height**: No. of edges below a node is known as its height.

**\*\*\* Height of tree: No. of edges in longest branch of tree. \*\*\***

**Ancestors**: nodes before a node are called its ancestors.

**Descendants:** Nodes after a node are called its Descendants.

**Degree: No. of Childs a node have is known as its degree.**

**\*\*\* Degree of tree: The highest degree of all nodes is known as degree of tree \*\*\***

**CEO**

Root(Example)

**Binary Tree**

**N nodes = N -1 edges**

Head

Head

Head

Head

COO

CTO

*A Tree with at most 2 degree is known as Binary Tree. A binary tree always follows, hum 2 humaare 2. (1 aur 0 bhi ho skta h.)*

Heads of different departments

This is also a binary tree

with degree 1

**It’s not a Binary Tree**

**As its tree degree is 3.**

1. **Full or Strict Binary Tree:** The Binary tree whose all nodes, either have 1 or 0 degree is known as full or strict binary tree.

**Types of Binary Trees**

NOT A FULL OR STRICT



As degree of this

node is 1

1. **Perfect Binary Tree:** The binary tree who’s all the internal nodes have degree 2 and all the leafs are on same level. (Ab leaf h toh degree toh zero hi hogi, 1 ya 2 hui toh leaf hi nai reh jayega)

All the internal nodes have 2 Childs BUT all leafs are not



On same level.

This internal node only

Have one child.



All the leafs on same level



Now in this, all the internal nodes are with degree 2 and all leafs are on

Same level.

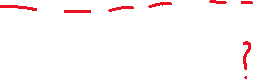
1. **Complete Binary Tree:** The binary tree who’s all levels are completely filed except possibly the last level BUT it must have its key on the left.



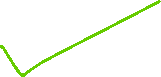
Unfilled level has its keys on left BUT it’s not the last level



* All levels filled except last but



it have its unfilled key on right.



All Levels filled except last that too all keys on as left as possible

Remember: Agar Right left dono ho, mtlb perfect binary tree is also

a complete binary tree.



1. **Degenerate Binary Tree: The binary tree who’s all the parents have 1 child.**



*It’s the correct degenerate*

*Binary tree. Last node has 0 child,*

*Hence, it’s not a parent either hence definition* It’s also a parent   
 *Is correct.*

but have 2 Childs

1. **Skewed Trees:** The tree’s who’s all the parent have only 1 child and all the keys are on either side.

**Left Skewed Binary Tree** **Right Skewed Binary Tree**

***MAXIMUM HEIGHT OF A TREE: n – 1***

***MINIMUM HEIGHT OF A TREE: log n***

(Minimum height is floor of Log n, issiliye peeche mt padna ki decimal ka kya???)

**4**

9

13

5

9

2



**Representation of BT**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 9 | 13 | 5 | Null | 2 | 9 |

1. **Array Representation of Binary Tree -**

Array size required: 2No. of levels -1

Example: size = 23 - 1 = 7



**\*\* Levels = Depth = Height \*\***

0 1 2 3 4 5 6

It 5follow’s left to right, if any position is missing, then null

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 9 | 13 | 5 | N | 2 | 9 | 6 | N | N | N | N | N | 4 | 1 |

will be placed instead of it.

**4**

9

13

5

9

2

6

1

4

Size Required: 2depth - 1

= 24 - 1

= 15



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Array Representation is not good, as to extend the tree, we need to create a new array and then copy all the elements, also to add any branch (assume after 9(index 1)) then whole array needs to be incremented.

1. **Linked Representation of binary tree: -**

Doubly linked list is used to represent a binary tree in linked representation.

\*\* It’s linked representation, do not call it linked list representation as

Linked list is a linear DS whereas Binary tree is a Non -Linear DS. \*\*

|  |  |  |
| --- | --- | --- |
| Pointer to Left | DATA | Pointer to Right |

Node Representation:

*In Linked Representation, a node has 3 fields, pointer to left key(node), data, pointer to right(key), Pointer holds the address of next node (whole node, not just data). If any parent has only 1 child or no child then the pointers will be NULL.*

**4**

9

13

5

9

2

6

1

4

Below is the linked rep. of this.

|  |  |  |
| --- | --- | --- |
|  | 4 |  |

|  |  |  |
| --- | --- | --- |
|  | 9 | Null |

|  |  |  |
| --- | --- | --- |
|  | 1 |  |

|  |  |  |
| --- | --- | --- |
|  | 5 | Null |

|  |  |  |
| --- | --- | --- |
| Null | 2 | Null |

|  |  |  |
| --- | --- | --- |
|  | 9 |  |

|  |  |  |
| --- | --- | --- |
| Null | 6 | Null |

|  |  |  |
| --- | --- | --- |
| Null | 4 | Null |

|  |  |  |
| --- | --- | --- |
| Null | 1 | Null |

\*\* Left hamesa left mai rahega and Right hamesa right mai rahega. Sirf root ki position badlegi vo bhi name ke according. \*\*

**Transversal of Binary Tree**

1. **Pre-Order Transversal:** First root, then left sub tree and then right sub tree.

**Root –> Left –> Right**

1. **Post-Order Transversal:** As name suggest, Root at post(LAST) and left and right as usual, on left and right

(First left sub tree, then right sub tree, and finally root)

**Left –> Right –> ROOT**

1. **In- Order Transversal:** Again, as of name, Root in between and left on left and right on right.

(First left sub tree, then root, then right sub tree)

**Left –> ROOT –> Right**

**1**

2

33

4

6

5

7

**Pre: root[left][right]**

**1 [2[4][5]] [3[][6]]** //now 2 is root and have left and right sub trees

**1 [2 [4 [][7]]] [3[][6]]**

**Pre-Order Transversal Result: 1247536**

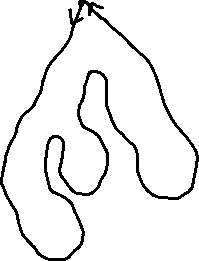
**Post-Order Transversal Result: 7452631**

**In-Order Transversal Result: 4725136**

**TIME COMPLEXITY OF ALL: O(n)**



Quick Truck to checkresults: **Above Tree is used for C code**



**1**

2

33

4

6

5

7

**1**

2

33

4

6

5

7

**1**

2

33

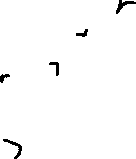
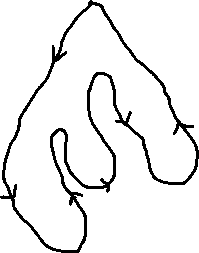
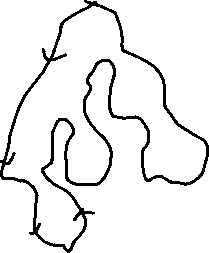
4

6

5

7

**1247536** **7452631** **4725136**



For pre, draw lines on left for post draw lines on right and for in draw lines below. Now start riding bike from left to right (as in black line) and as you cross red line, write it’s value.

**Binary search tree is a binary tree which satisfies the following properties:**

**Binary Search Tree**

1. **All the elements of left subtree are smaller than root element. (Not Equal too)**

**(The value of root element must be greater than all the values of Left sub tree)**

1. **All the elements of right subtree are greater than root element. (Not Equal too))**

**(The value of root element must be smaller than all the values of Right sub tree)**

1. **Any element is not repeated in whole Binary Tree.**
2. **Left and right sub trees are also the Binary search trees.**
3. **In-Order Transversal of a binary tree gives an ascending order Sorted Array.**



**10**

4

15

2

19

8

3

**10**

4

15

2

19

11

3



**10**

4

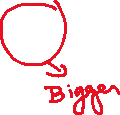
13

4

6

5

1



**In-Order Transversal: 2 3 4 8 10 15 19**

**(Perfect Sorted Array)**

1

2

3

4



**Why Binary Search Tree?**

**Searching…..**

Binary search tree makes it extremely simple and efficient for an element to search.

**Time complexity of normal tree: O(n)**

(We need to transverse through all the elements.)

Time complexity of Binary Search Tree:

**In worst case** (may need to transverse through all elements)**: O(n)**

**In General: O(h)**

**(h = height of tree)**

**\*\*Minimum Height of a Binary Tree, h = log n, Hence,**

**Best Case: O(log n)**

**log n <= h <= n**

1

2

3

4



**10**

4

15

2

19

8

3



**Worst Case of BST, h = n = 3 BEST CASE OF BST, h = log 7 = 2**

**10**

4

15

19

***MAXIMUM HEIGHT OF A TREE: n – 1***

***MINIMUM HEIGHT OF A TREE: log n***

Actually, its floor(log n)

Hence, decimal is not considered

The Height is considered log n as for now

Also, height is 2