**Binary Tree**

**Properties**

1.MAX Nodes at level I = 2i

2.MAX Nodes at height h = 2h+1-1

3.MIN Nodes at height h = h+1

4.MAX height = n-1

5.MIN height = [log2(n+1)-1]

**Types**

1. Full/Proper/Strict
2. Complete
3. Perfect Binary Tree
4. **Full/Proper/Strict Binary Tree**

Each node having 0 or 2 children except leaf node.

1. **Complete Binary Tree**

If all levels are filled except the last node.

The last level has the nodes as left as possible.

1. **Perfect Binary Tree**

All internal nodes have 2 children and

all leaf nodes are at same level.

|  |  |  |
| --- | --- | --- |
|  | MAX Nodes | MIN Nodes |
| Binary | 2h+1-1 | h+1 |
| Full | 2h+1-1 | 2h+1 |
| Complete | 2h+1-1 | 2h |

|  |  |  |
| --- | --- | --- |
|  | MIN Height | MAX Height |
| Binary | [log2(n+1)-1] | n-1 |
| Full | [log2(n+1)-1] | (n+1)/2 |
| Complete | [log2(n+1)-1] | Log(n) |

**Binary Tree in an Array**

**Case1:**

If node is at index I

Left child = (2\*I) + 1

Right child = 2\*I + 2

Parent child = (I – 1)/2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Example:

I = 3

Left Child = 2\*3+1 = 7

Right Child = 2\*3+2 = 8

Parent Child = (2-1)/2 = 1

**Case 2:** If node is at index I

Left Child = 2\*I

Right Child = (2\*I) + 1

Parent Child = (I/2)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Example:

I = 3

LC = 2\*4 = 8

RH = 9

Parent = 4/2 = 2

**Convert it into Array**

A picture containing shape

Description automatically generated

ANS:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | - | - | H | I |

I = 3 (D)

LC = 2\*3+1 = 7

RC = 2\*3+2 = 8

Parent = 1 (B)

**Dynamic/Node Representation**

Inserting through recursion.

**Traversals in Binary Trees**

1. In Order = Left, Root, Right

2. Pre Order = Root, Left, Right

3. Post Order = Left, Right, Root

**Construct a binary tree from the below mentioned post and in-order traversal.**

Post Order: 9, 1, 2, 12, 7, 5, 3, 11, 4, 8

In-Order: 9, 5, 1, 7, 2, 12, 8, 4, 3, 11