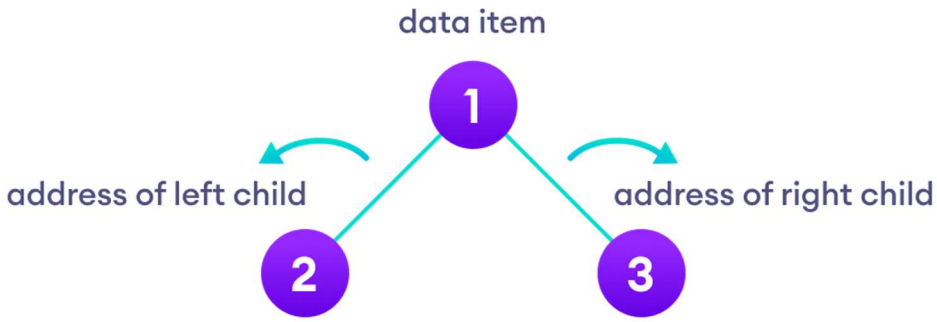
Binary Tree

A binary tree is a tree data structure in which each parent node can have at most two children. Each node of a binary tree consists of three items:

- data item
- address of left child
- address of right child

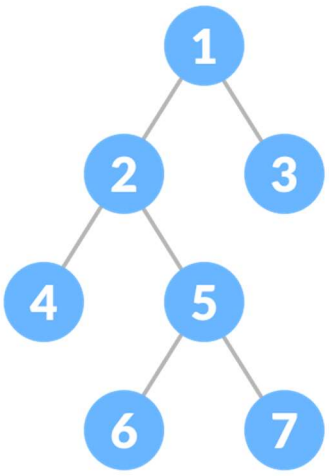


Binary Tree

Types of Binary Tree

1. Full Binary Tree

A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children.

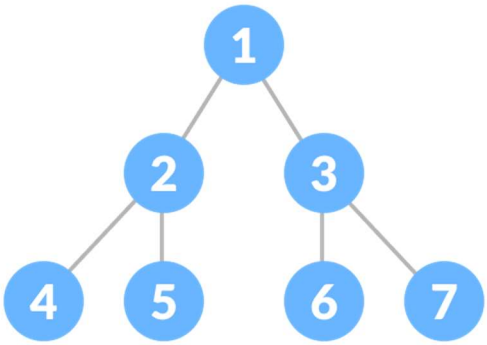


Full Binary Tree

To learn more, please visit full binary tree.

2. Perfect Binary Tree

A perfect binary tree is a type of binary tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.



Perfect Binary Tree

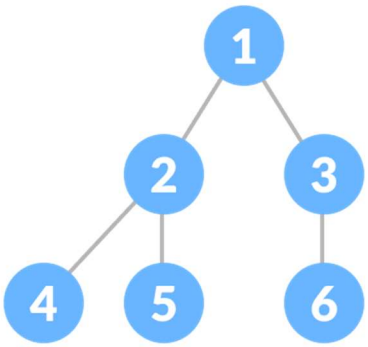
To learn more, please visit perfect binary tree.

3. Complete Binary Tree

A complete binary tree is just like a full binary tree, but with two major differences

1. Every level must be completely filled

2. All the leaf elements must lean towards the left.
3. The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree.

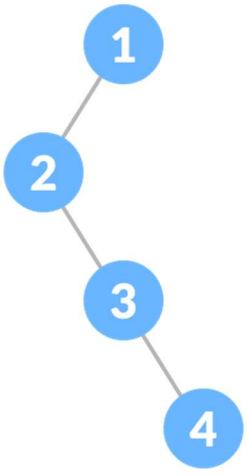


Complete Binary Tree

To learn more, please visit complete binary tree.

4. Degenerate or Pathological Tree

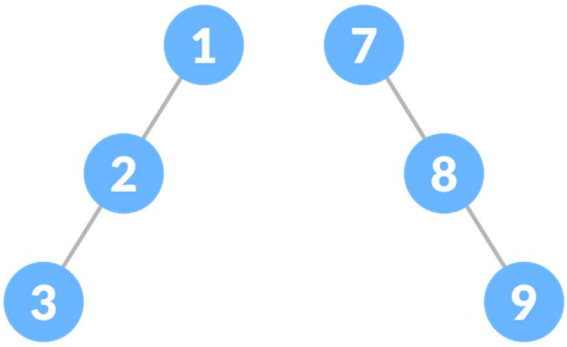
A degenerate or pathological tree is the tree having a single child either left or right.



Degenerate Binary Tree

5. Skewed Binary Tree

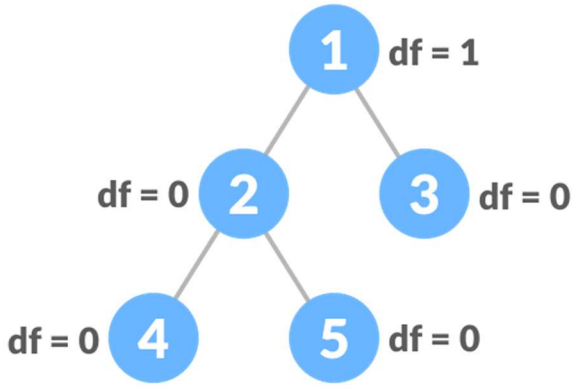
A skewed binary tree is a pathological/degenerate tree in which the tree is either dominated by the left nodes or the right nodes. Thus, there are two types of skewed binary tree: **left-skewed binary tree** and **right-skewed binary tree**.



Skewed Binary Tree

6. Balanced Binary Tree

It is a type of binary tree in which the difference between the height of the left and the right subtree for each node is either 0 or 1.



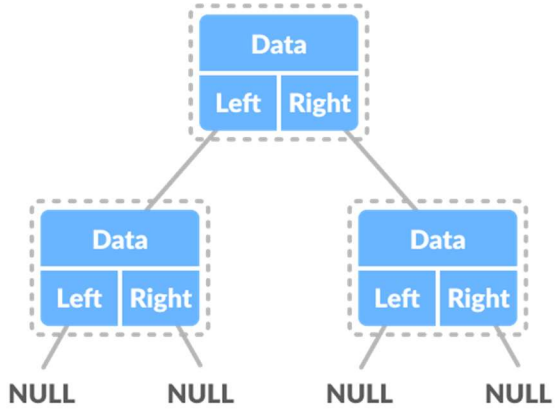
Balanced Binary Tree

To learn more, please visit balanced binary tree.

Binary Tree Representation

A node of a binary tree is represented by a structure containing a data part and two pointers to other structures of the same type.

```
struct node
{
 int data;
 struct node *left;
 struct node *right;
};
```



// Binary Tree in C++

#include <stdlib.h>

#include <iostream>

using namespace std;

struct node {

 int data;

 struct node *left;

```
  struct node *right;
};

// New node creation
struct node *newNode(int data) {
  struct node *node = (struct node *)malloc(sizeof(struct node));

  node->data = data;

  node->left = NULL;
  node->right = NULL;
  return (node);
}

// Traverse Preorder
void traversePreOrder(struct node *temp) {
  if (temp != NULL) {
    cout << " " << temp->data;
    traversePreOrder(temp->left);
    traversePreOrder(temp->right);
  }
}

// Traverse Inorder
void traverseInOrder(struct node *temp) {
  if (temp != NULL) {
    traverseInOrder(temp->left);
    cout << " " << temp->data;
    traverseInOrder(temp->right);
```

```
  }
}

// Traverse Postorder
void traversePostOrder(struct node *temp) {
  if (temp != NULL) {
    traversePostOrder(temp->left);
    traversePostOrder(temp->right);
    cout << " " << temp->data;
  }
}

int main() {
  struct node *root = newNode(1);
  root->left = newNode(2);
  root->right = newNode(3);
  root->left->left = newNode(4);

  cout << "preorder traversal: ";
  traversePreOrder(root);
  cout << "\nInorder traversal: ";
  traverseInOrder(root);
  cout << "\nPostorder traversal: ";
  traversePostOrder(root);
}
```