# DSA Tutorial Sheet 3

## Binary Trees

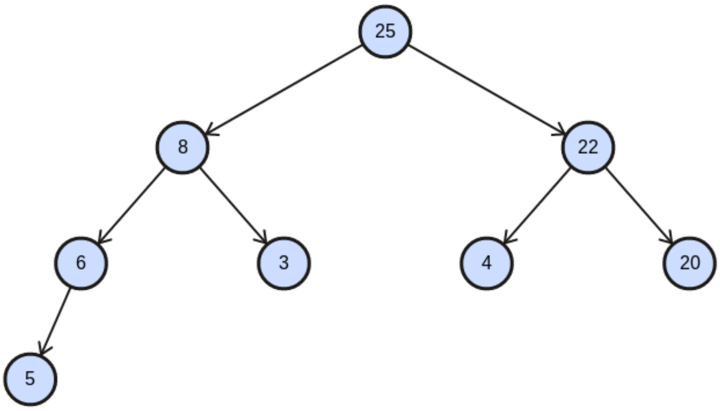
### 3.1 Binary Trees

Binary Tree is a data structure in which information is stored in the node and each node has pointers to left and right nodes. Following is the declaration of Tree Node in C++:

```
typedef struct Node {
    int data;
    Node *left;
    Node *right;

    Node(int data) {
        this->data = data;
        this->left = NULL;
        this->right = NULL;
    }

} Node;
```



### 3.1.1 Traversals

1. PreOrder Travesal : 25 8 6 5 3 22 4 20

2. InOrder Traversal : 5 6 8 3 25 4 22 20

3. PostOrder Traversal : 5 6 3 8 4 20 22 25

4. Level Order Traversal : 25 8 22 6 3 4 20 5

Try coding all the traversals recursively and iteratively (Hint: For iterative use stacks).

5. Spiral Traversal : 25 8 22 20 4 3 6 5 (Hint: Make use of queue or stack)

6. Vertical Order Traversal : 5 6 8 25 3 4 22 20

### 3.1.2 Views of Binary Tree

1. Left View : 25 8 6 5

2. Right View : 25 22 20 5

3. Bottom View : 5 6 8 3 22 20

4. Top View : 25 8 22 6 20 5

(Hint: Read about map and unordered_map Data Structures in C++ or HashMap in Java)

### 3.1.3 Querying on Trees using SubTrees

These type of questions are solved mostly using recursion on subtrees.

1. Print the height of the Tree

2. Print the number of nodes of the Tree.

3. Print the number of leaf nodes.

4. Check if two Trees are identical (The structure and node values of the Tree should be same)

5. Check if a Tree is mirror of itself.

6. Check if a Tree is balanced. (A Tree is balanced if for every node the difference of heights of left subtree and right subtree of that node is not greater than 1)

7. Print the diameter of the Tree. (The maximum number of connected nodes in the tree in the longest path.) Ex- The diameter of the tree given above is 5.

8. If you are given two traversal sequences, can you construct unique binary tree ?