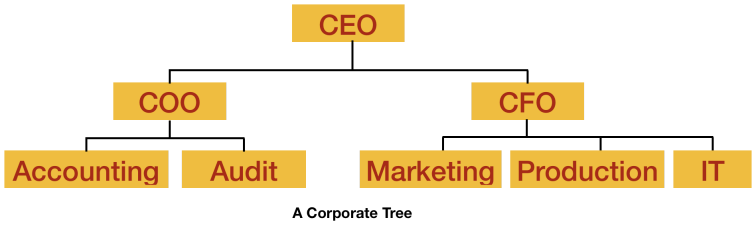
[redacted] (/)

Tree Data Structures

We have studied linked lists, stacks and queues and all of them are linear data structures. In this and the next few chapters, we will learn about trees, which is a non-linear data structure.

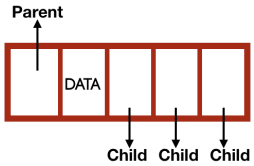
In this chapter, we are just going to look at the basic idea of a tree data structure and we will go through some specific trees in later chapters.

Trees store data in a hierarchical order. Let's look at an example of a corporate structure stored in a tree.

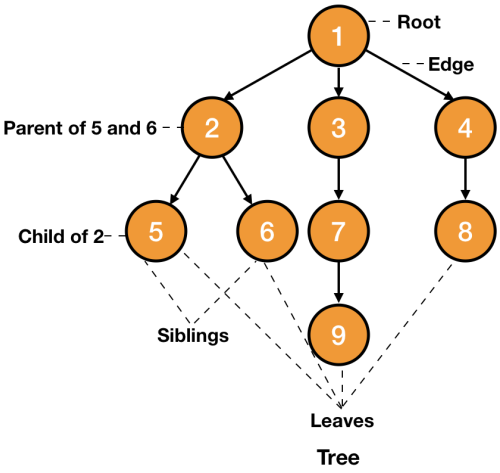


A Corporate Tree

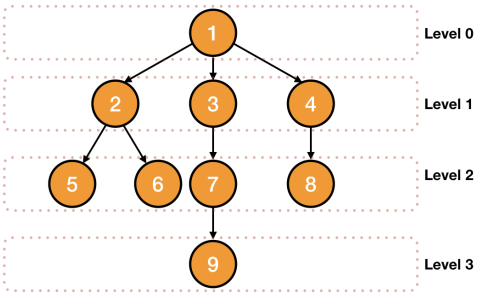
The above picture is a tree. In general, each node of a tree has children and parent as shown in the picture given below.



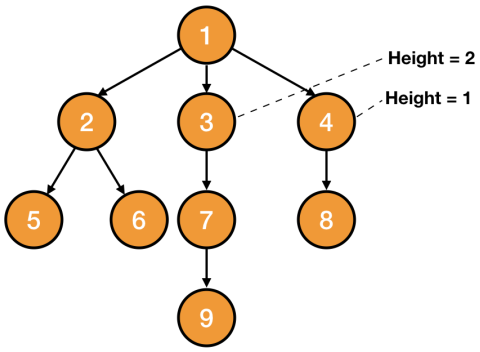
Similar to children and parent, there are many other terms which are used with a tree. So, let's have a look at some important terms used in a tree.



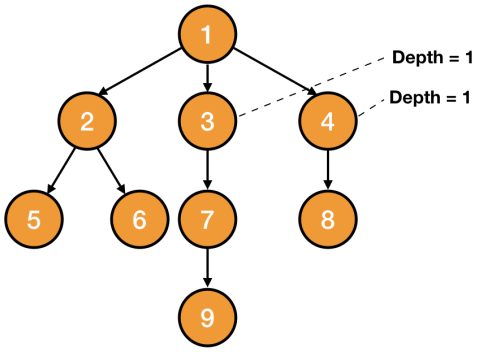
- **Root** → The topmost node of the hierarchy is called the root of the tree.
- **Child** → Nodes next in the hierarchy are the children of the previous node.
- **Parent** → The node just previous to the current node is the parent of the current node.
- **Siblings** → Nodes with the same parent are called siblings.
- **Ancestors** → Nodes which are higher in the hierarchy are ancestors of a given node.
- **Descendents** → Nodes which are lower in the hierarchy are descendants of a given node.
- **Internal Nodes** → Nodes with at least one child are internal nodes.
- **External Nodes/Leaves** → Nodes which don't have any child are called leaves of a tree.
- **Edge** → The link between two nodes is called an edge.
- **Level** → The root of a tree is at level 0 and the nodes whose parent is root are at level 1 and so on.



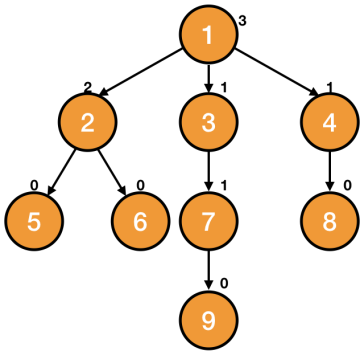
- **Height** → The height of a node is the number of nodes (excluding the node) on the longest path from the node to a leaf.



- **Height of Tree** → Height of a tree is the height of its root.
- **Depth** → The depth of a node is the number of nodes (excluding the node) on the path from the root to the node.



- **Node Degree** → It is the maximum number of children a node has.



- **Tree Degree** → Tree degree is the maximum of the node degrees. So, the tree degree in the above picture is 3.

Till now, we have an idea of what a tree is and the terminologies we use with a tree. But we don't know yet what the specific properties of a tree are or which structure should qualify as a tree. So, let's see the properties of a tree.

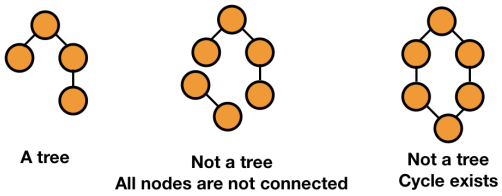
Properties of a Tree

A tree must have some properties so that we can differentiate from other data structures. So, let's look at the properties of a tree.

The numbers of nodes in a tree must be a finite and nonempty set.

There must exist a path to every node of a tree i.e., every node must be connected to some other node.

There must not be any cycles in the tree. It means that the number of edges is one less than the number of nodes.

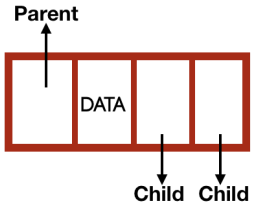


You must be pretty much clear about trees. So, let's code up a tree.

Coding Node of a Tree

Till now, it must be clear that a node of a tree contains information of its parent, children and data. So, we can accordingly make nodes in different languages. Sometimes, we also use an array to represent a tree as you will see in the next chapter of binary trees.

For now, let's just make a general node of a tree in different languages.



C

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct tree_node {
  int data;
  struct tree_node *right;
  struct tree_node *left;
  struct tree_node *parent;
}tree_node;

tree_node* new_tree_node(int data) {
  tree_node *n = malloc(sizeof(tree_node));
  n->data = data;
  n->right = NULL;
  n->left = NULL;

  return n;
}

int main() {
  return 0;
}
```

Java

```java
class TreeNode {
  public int data;
  public TreeNode right;
  public TreeNode left;
  public TreeNode parent;

  public TreeNode(int data) {
    this.data = data;
    this.right = null;
    this.left = null;
    this.parent = null;
  }

  public static void main(String[] args) {

  }
}
```

Python

```python
class TreeNode:
  def __init__(self, data):
    self.data = data
    self.right = None
    self.left = None
    self.parent = None

if __name__ == '__main__':
  pass
```

That's all about a tree for now. In the next chapters, we will look at different trees.

> Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less
>
> - Marie Curie

PREV (/course/data-structures-queue/) (/course/data-structures-binary-trees/) NEXT

Further Readings

➔ Trees in Computer Science (/blog/article/trees-in-computer-science/)