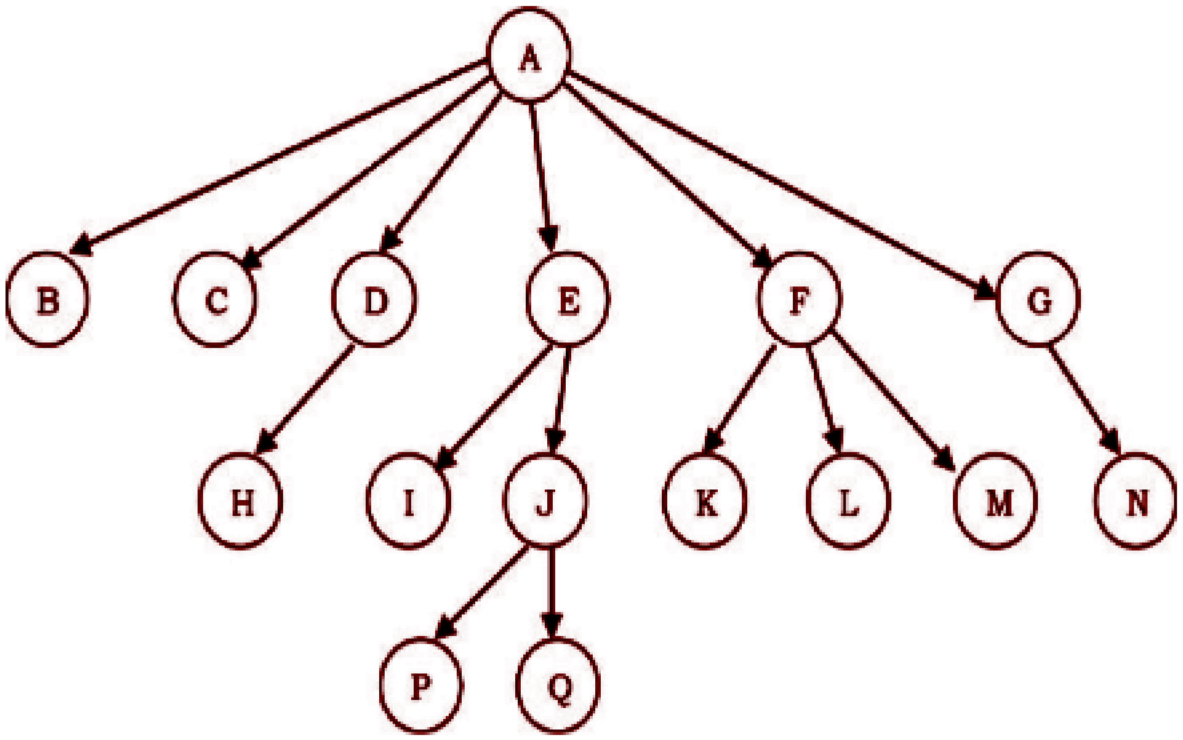
Generic Trees

Introduction

- In the previous modules, we discussed binary trees where each node can have a maximum of two children and these can be represented easily with two pointers i.e right child and left child.
- But suppose, we have a tree with many children for each node.
- If we do not know how many children a node can have, how do we represent such a tree?
- **For example**, consider the tree shown below.



Generic Tree Node

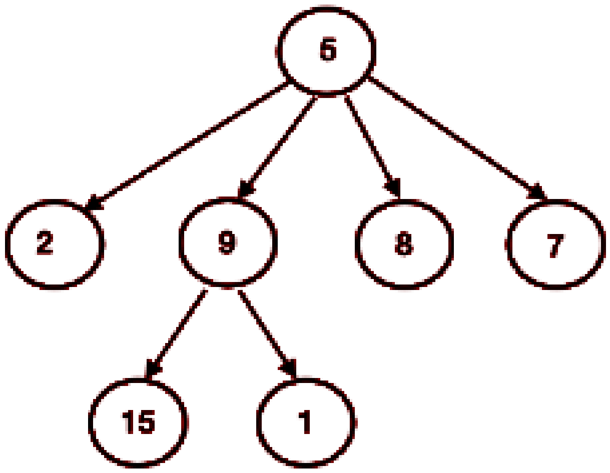
- Implementation of a generic tree node in Python is quite simple.
- The node class will contain two attributes:
  - The node **data**
  - Since there can be a variable number of children nodes, thus the second attribute will be a list of its children nodes. Each child is an instance of the same node class. This is a general **n-nary tree**.
- Consider the given implementation of the **Generic Tree Node** class:

```
class GenericTreeNode:
    def __init__ (self, data):
        self.data = data #Node data
        self.children = list() #List of children nodes
```

Adding Nodes to a Generic Tree

We can add nodes to a generic tree by simply using the **list.append()** function, to add children nodes to parent nodes. This is shown using the given example:

Suppose we have to construct the given Generic Tree:



Consider the given **Python code:**

```
#Create nodes
n1= TreeNode(5)
n2 =TreeNode(2)
n3 =TreeNode(9)
n4 =TreeNode(8)
n5 =TreeNode(7)
n6 =TreeNode(15)
n7 =TreeNode(1)

#Add children for node 5 (n1)
n1.children.append(n2)
n1.children.append(n3)
n1.children.append(n4)
n1.children.append(n5)

#Add children for node 9 (n3)
n3.children.append(n6)
n3.children.append(n7)
```

Print a Generic Tree (Recursively)

In order to print a given generic tree, we will recursively traverse the entire tree. The steps shall be as follows:

- If the root is **None**, i.e. the tree is an empty tree, then return **None**.
- For every child node at a given node, we call the function recursively.

Go through the given Python code for better understanding:

```
def printTree(root):
      #Not a base case but an edge case
      if root == None:
            return

      print(root.data) #Print current node's data
      for child in root.children:
            printTree(child) #Recursively call the function for children
```

Take Generic Tree Input (Recursively)

Go through the given Python code for better understanding:

```
def takeTreeInput():
      print("Enter root Data")
      rootData  = int(input())#TAKE USER INPUT
      if rootData == -1:#Stop taking inputs
            return None

      root = TreeNode(rootData)

      print("Enter number of children for  ",  rootData)
      childrenCount = int(input()) #Get input for no. of child nodes
      for in range(childrenCount):
            child = takeTreeInput() #Input for all childs
            root.children.append(child) #Add child
      return root
```