DISCRETE STRUCTURES
Lab 10
Tree

1. Introduction

In this tutorial, we will practice graph presentation on computer and matrix techniques.

2. Binary Tree representation

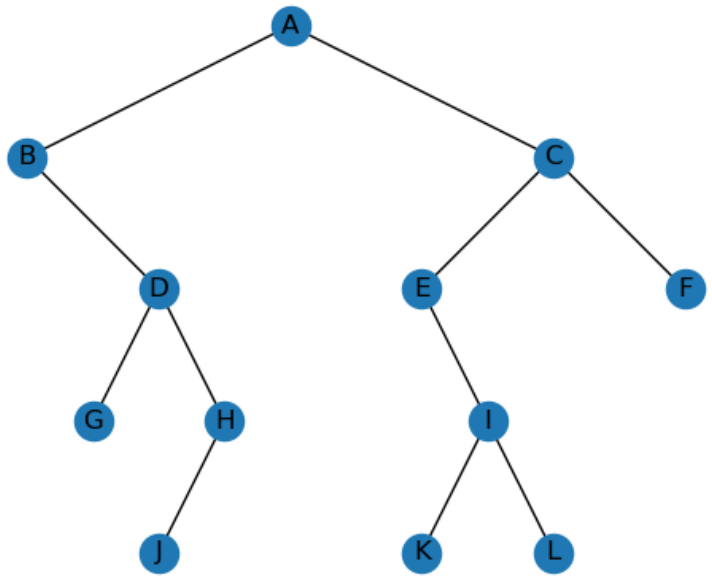
Read following lecture notes of **Discrete Structures** on site elit.tdtu.edu.vn

Week14_Graphs_and_Trees_2.pdf

2.1. Vector

A binary tree can be represented using a vector where element (count from 0) $2^i - 1 + j \; and \; 2^i + j$ is the children element $2^{i-1} - 1 + j$

Where:
i is the level of the node
j is the order of the node in i^{th} level from left to right.

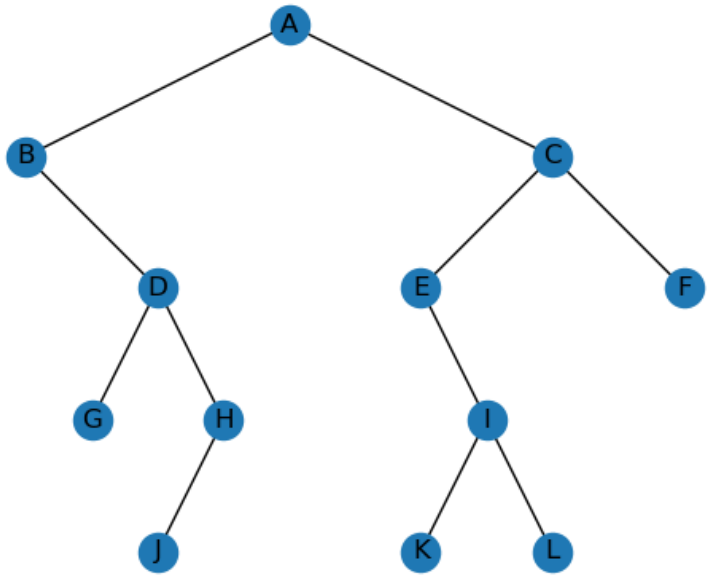


Normally, this method can only represent a complete binary tree. However, if we represent the empty children as None, the tree can be represented by

```
T=['A',
'B','C',
None,'D','E','F',
None,None,'G','H',None,'I',None,None,
None,None,None,None,None,None,'J',None,None,None,'K','L',None,No
ne,None,None]
```

2.2. Vector

Another common way to represent the tree is by using linked list. In binary tree case, we can use a linked list with maximum two children on each note.



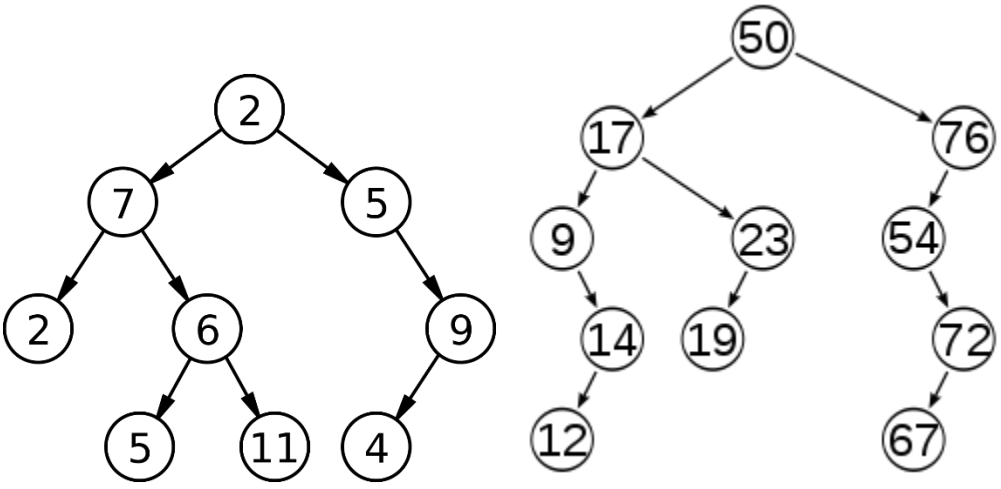
This Tree can be represented by:

```
class bNode(object):
    def __init__(self,data=None):
        self.left = None
        self.right = None
        self.data = data
A=bNode('A')
A.left=bNode('B')
A.right=bNode('C')
B=A.left
```

```
C=A.right
B.right=bNode('D')
D=B.right
C.left=bNode('E')
C.right=bNode('F')
E=C.left
F=C.right
D.left=bNode('G')
D.right=bNode('H')
E.left=bNode('I')
G=D.left
H=D.right
I=E.left
H.left=bNode('J')
J=H.right
I.left=bNode('K')
I.right=bNode('L')
K=I.left
L=I.right
```

3. Exercise

1. Represent the following trees as Vector of values and print out each level in separate line:



2. Represent the 2 previous binary trees using linked list.
3. Write functions **NLR(A), LNR(A), LRN(A)** for linked list to print out the pre-order, in-order, and post-order traversals of the previous trees.

4. Write functions **breadthFirstSearch(A,Data), depthFirstSearch(A,Data)** for linked list to find H from the previous section tree, 11 from the left tree and 19 from the right tree and print the path you have traveled to find the data.
5. Write functions **breadthFirstSearchV(A,Data)** for vector to find H from the previous section tree, 11 from the left tree and 19 from the right tree and print the path you have traveled to find the data.