upGrad

Full Stack Software Development

upGrad

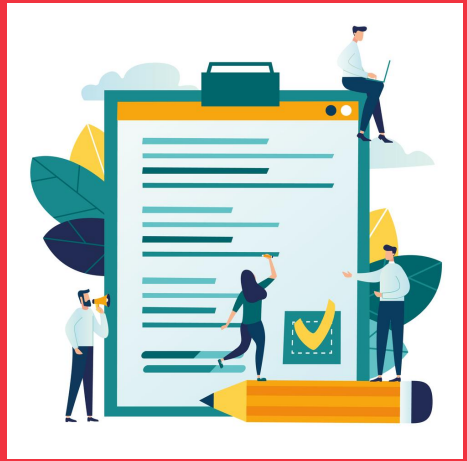
Course: Data Structures and Algorithm

Lecture On: Trees

Instructor: Arkoprovo Dey

In the previous class, we covered….

- Strings



Homework Discussion

1. Remove characters from the first String which are present in the second String.

   Sample Input:

   education

   rate

   Sample Output:

   ducion

   ('e', 'a', 't' are present in 2nd string too, so remove them from 1st string)

1. Print first non-repeated character from String.

   Sample Input:

   triggered

   Sample Output:

   t

Today's Agenda

1 Trees

Poll 1 (15 Sec.)

Which of the below is a linear data structure?

1. Array
2. Linked List
3. Stacks
4. Queues

Poll 1 (Answer)

Which of the below is a linear data structure?

1. **Array**
2. **Linked List**
3. **Stacks**
4. **Queues**

Non-Linear Data Structures

upGrad

We explored linear data structures such as arrays and linked lists. However, in the real world, data modelling is not always possible in a linear fashion.

Can you think of some real-world example where data is stored in a non-linear fashion?

How do you think Facebook suggests friends when we visit someone's profile? Or get connected with someone new?

Non-Linear Data Structures

upGrad

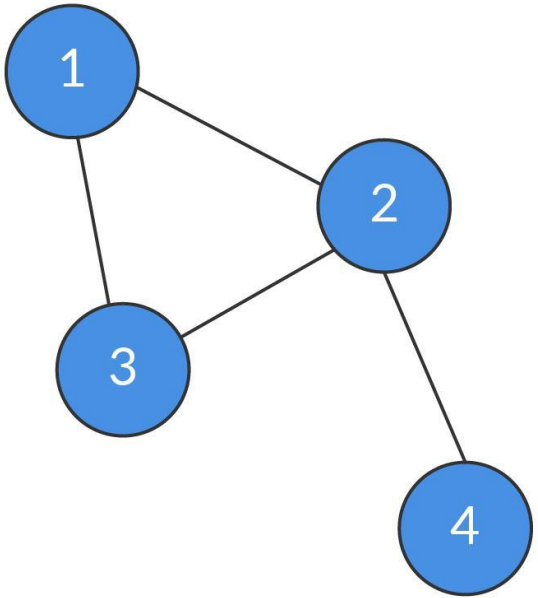
Non-linear data structures give us a powerful mechanism to store and manipulate data, making a lot of algorithms possible, which would not be easy to do if we used linear data structures.

They allow us to associate or connect one data element to several other data elements, or even model them in a hierarchical way, if we want to.

However, for the manipulation of data, be it insertion, searching or deletion, BSTs (Binary Search Trees) provide better running time.

Non-Linear Data Structures

upGrad



A graph ‘G’ can be represented by two attributes:

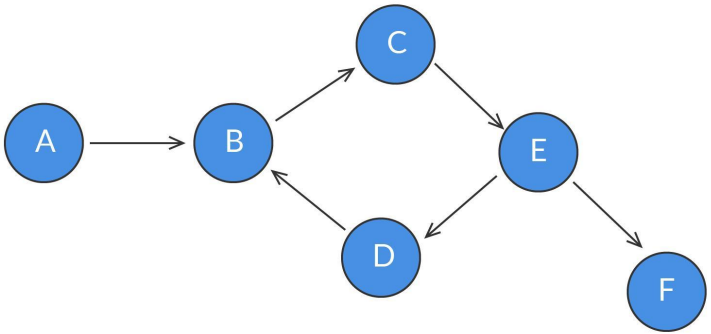
- Its set of vertices (V), and
- Its set of edges connecting these vertices (E).

Edges are represented as **pairs of vertices**. In the graph shown in this slide, the edges have no ‘direction’.

This type of graph is called an **undirected graph**.

Non-Linear Data Structures

upGrad



We can also have graphs where the **edges** have a **'direction'**.

In the adjoining graph,

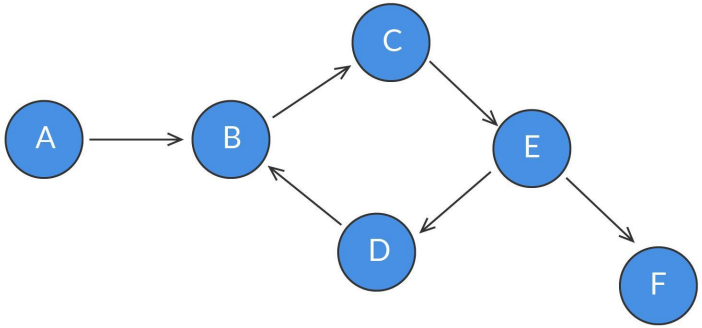
- The edge (A, B) is not the same as the edge (B, A).
- A graph might have both the edges (A,B) and (B,A), if need be.

These graphs are called **directed graphs**.

In directed graphs, the edges are represented in the format (u, v), where 'u' is the source vertex and 'v' is the destination vertex.

Non-Linear Data Structures

upGrad



Can you think of a real-world model where a directed graph would be useful?

How about one where undirected graphs would make more sense?

What kind of graph is implemented by Facebook? How about Instagram?

Poll 2 (15 Sec.)

Graph in which edges have no direction is called:

1. Directed graph
2. Undirected graph

Poll 2 (Answer)

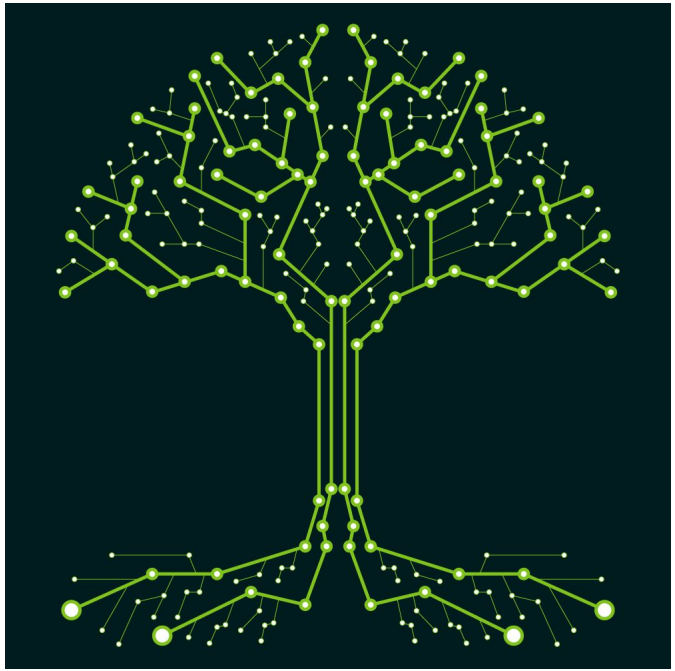
Graph in which edges have no direction is called:

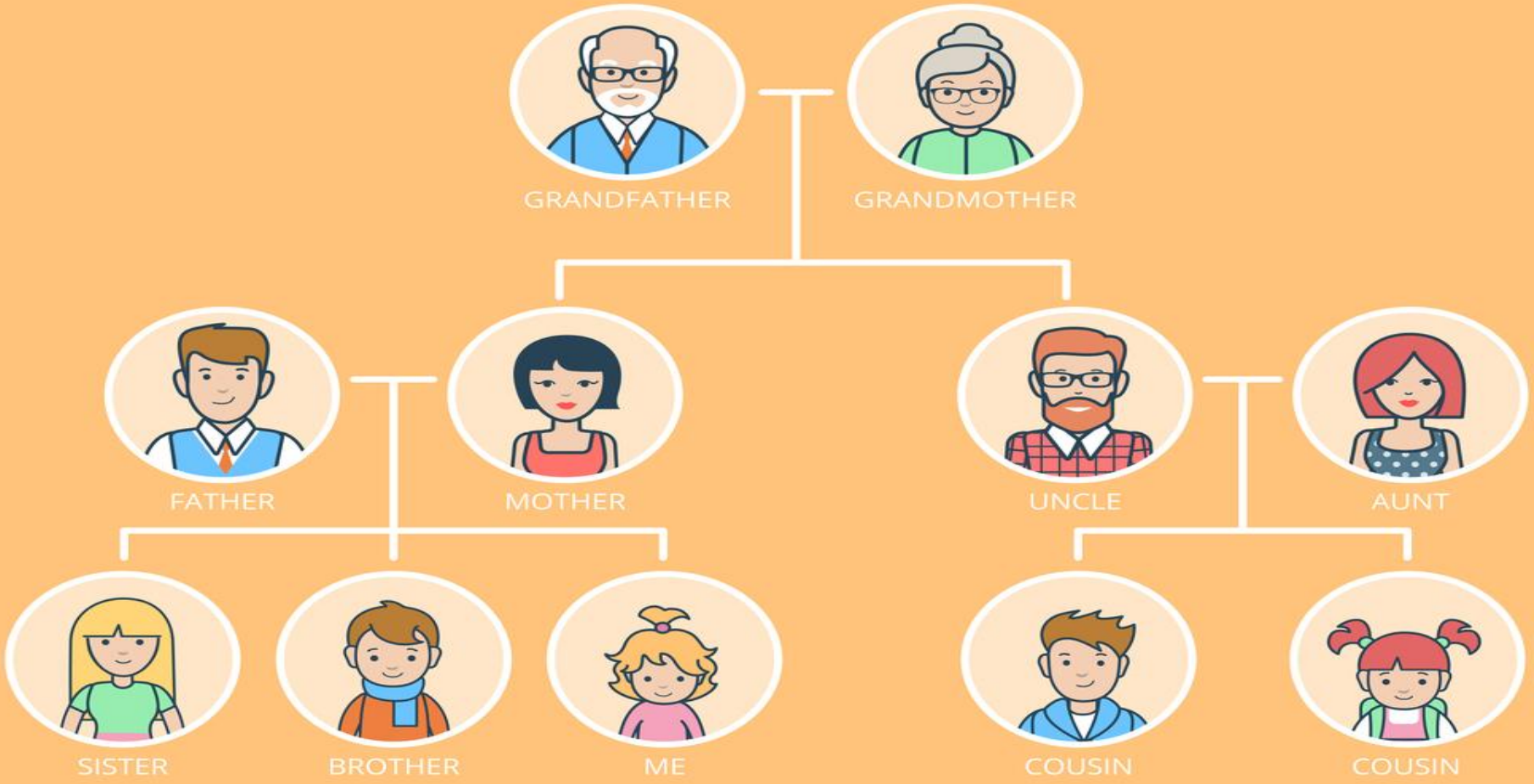
1. Directed graph
2. **Undirected graph**

Trees

upGrad

Tree data structure is a subclass of graph that deserves individual focus, simply because of the vast amount of power that it gives us in terms of efficient algorithm design. This subclass of graphs is called **Trees**.



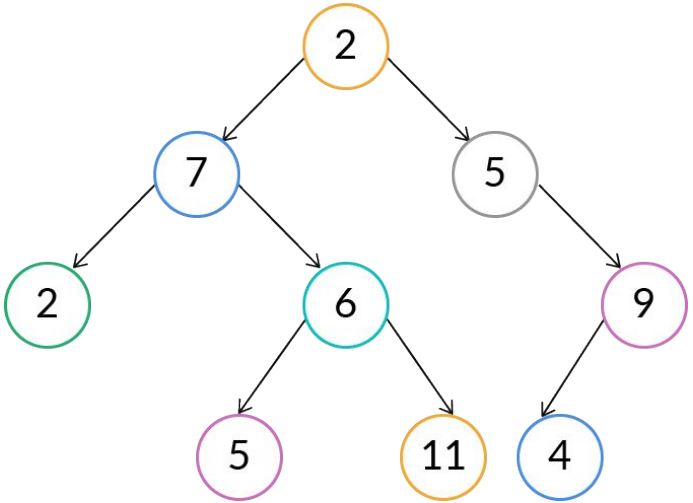

GRANDFATHER
GRANDMOTHER
FATHER
MOTHER
UNCLE
AUNT
SISTER
BROTHER
ME
COUSIN
COUSIN

Trees

upGrad

A tree is a graph that:

- is **simple**,
- **connected**,
- **undirected**,
- has **no cycles**
- and has ***n*-1** edges if the number of vertices is *n*.



A tree also has a few other features:

- The graph has no cycles, but a cycle will be formed even if one edge is added to it.
- The graph is connected, but will not remain connected if any vertex is removed from it.
- Any two nodes/vertices in the graph can be connected by **one simple unique** path.

Trees

upGrad

Now, let us take a look at a few tree terms and definitions:

- A tree consists of **one** ‘root’ node, zero or more ‘internal’ nodes and zero or more ‘leaf’ nodes.
- Each node can have zero or more children.
- A node can have **at most** one parent.
- A binary tree is a special kind of tree where a node can have zero to two nodes only.

Poll 3 (15 Sec.)

State true or false:

A tree consists of one 'root' node, zero or more 'internal' nodes and zero or more 'leaf' nodes.

1. True
2. False

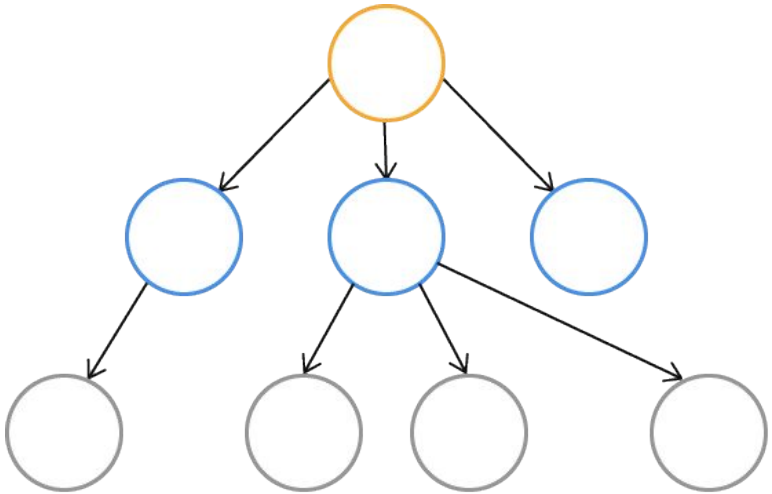
Poll 3 (Answer)

State true or false:

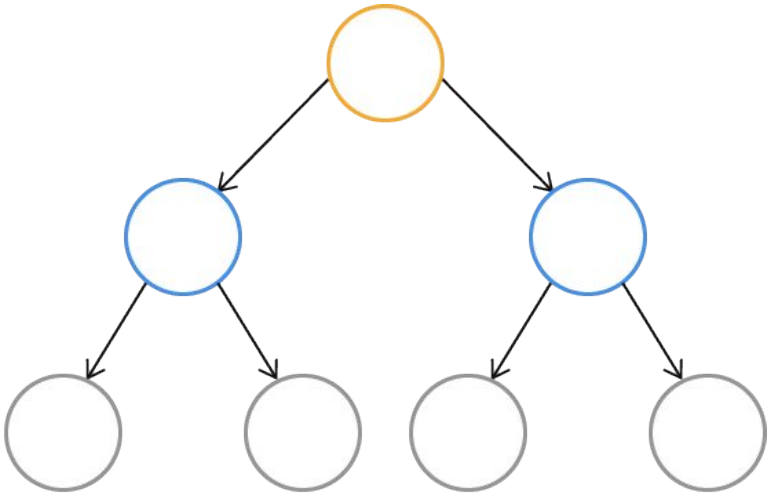
A tree consists of one 'root' node, zero or more 'internal' nodes and zero or more 'leaf' nodes.

1. **True**
2. False

Here are diagrammatic representations of a tree and a binary tree:



Tree



Binary Tree

Trees

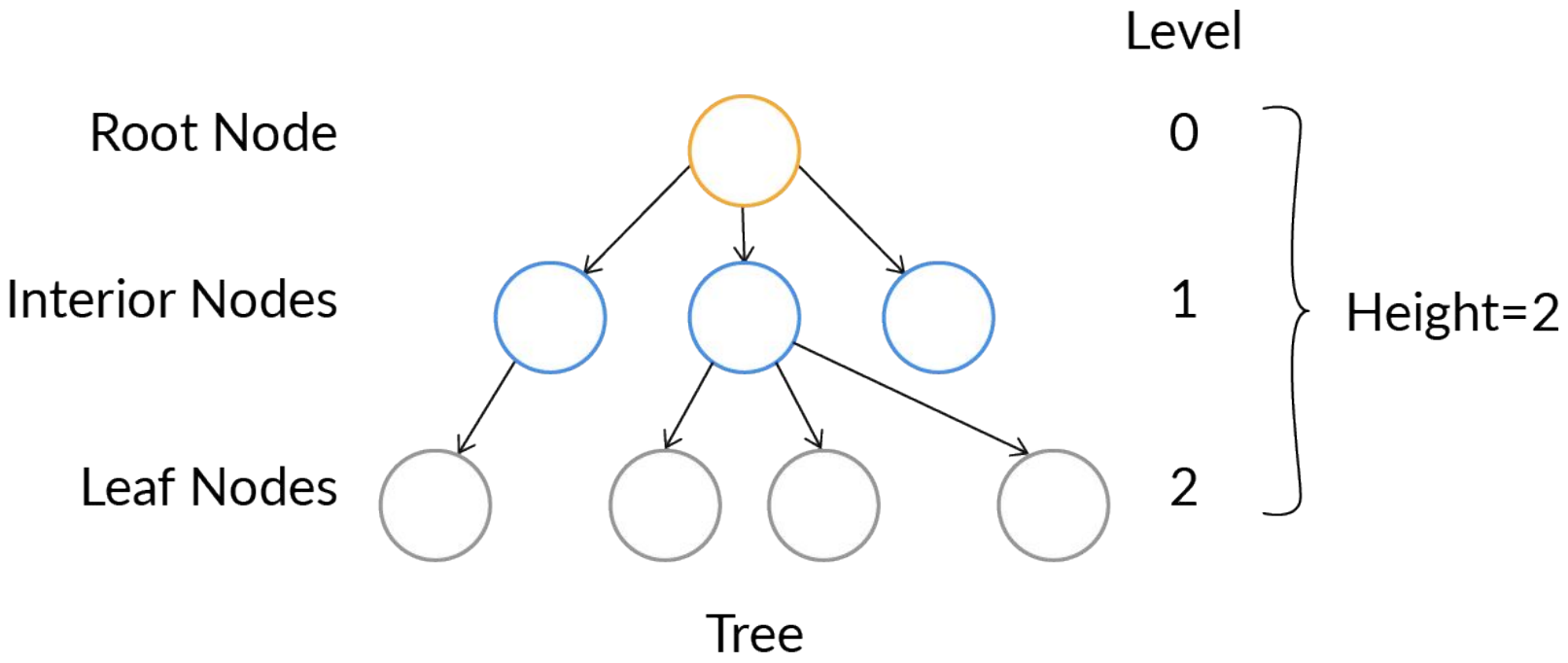
upGrad

Now, let's take a look at a few more terms and definitions related to a tree:

- The 'root' of a tree has **no parent**. (Trees are similar to traditional family trees.)
- The 'leaves' of a tree have **no children**.
- All other nodes are interior nodes.
- The height of a tree is the distance from the root of the tree to the farthest leaf node.
- The depth of any node is the distance from the root to the node.
- The 'child' nodes of all the non-leaf nodes can themselves be considered as trees in their own right, which are rooted at these nodes.

Trees

upGrad



Poll 4 (15 Sec.)

State true or false:

The height of a tree is the distance from the root of the tree to the farthest leaf node.

1. True
2. False

Poll 4 (Answer)

State true or false:

The height of a tree is the distance from the root of the tree to the farthest leaf node.

1. **True**
2. False

Now, let us learn about a few tree terms and definitions:

- The ‘degree’ of a node is the number of sub-trees or children the node has. As we can understand, the leaves are nodes with a degree of ‘0’.
- Children of the same parent node are known as ***siblings***.
- All the nodes in the path starting from the root of the tree right up to the parent of a node are known as its ***ancestors*** or ***predecessors***.

Binary Tree

upGrad

Binary trees are the most popular application of trees, and they help us in a large number of algorithms in reducing the time complexity. Binary trees, as we saw earlier, are a special type of trees, where the maximum degree of each node is 2. Binary trees, in turn, can be of a few types as shown below:

- **Degenerate** trees, where all the nodes have ***one*** child.
- **Fully** binary or **strictly** binary trees, where all the non-leaf nodes have ***two*** children.
- **Balanced** binary trees, where most of the nodes have two children and the sub-trees are mostly of equal height.
- **Complete** trees, where all the tree levels are complete, except possibly the last, which in itself must be filled starting from the left-most node.

Poll 5 (15 Sec.)

State true or false:

In fully binary trees, most of the nodes have two children and the sub-trees are mostly of equal height.

1. True
2. False

Poll 5 (Answer)

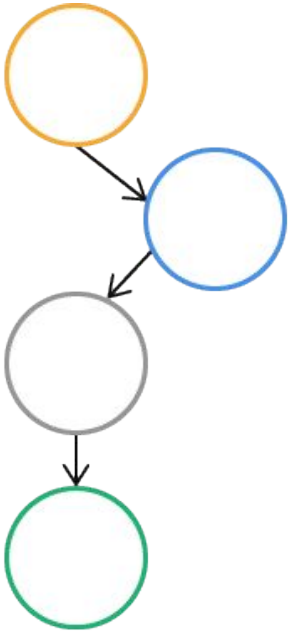
State true or false:

In fully binary trees, most of the nodes have two children and the sub-trees are mostly of equal height.

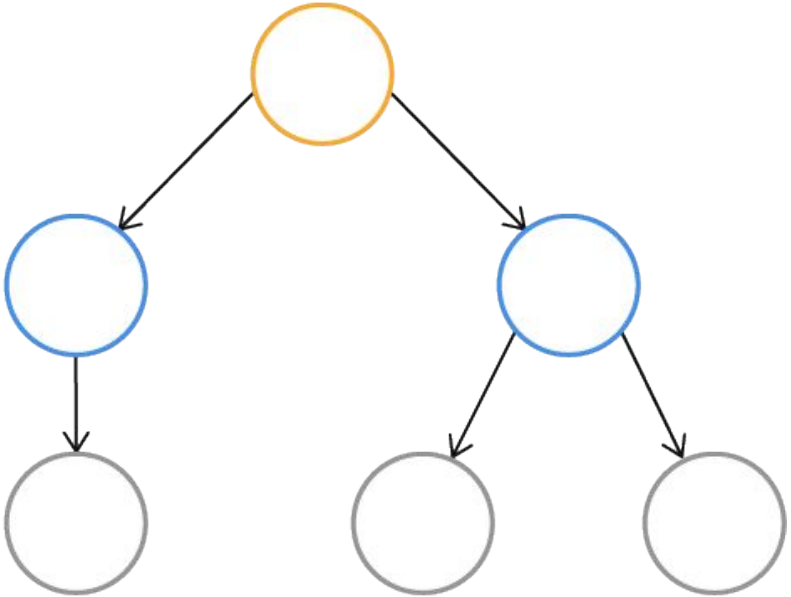
1. True
2. **False**

Binary Tree

upGrad



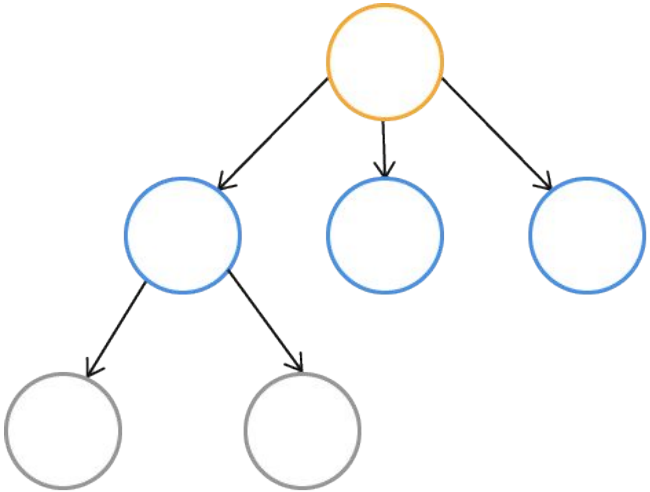
Degenerate binary tree



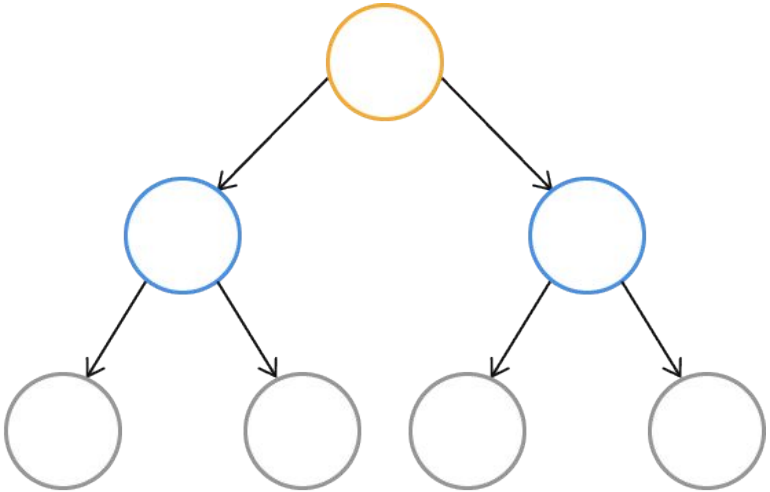
Balanced binary tree

Binary Tree

upGrad



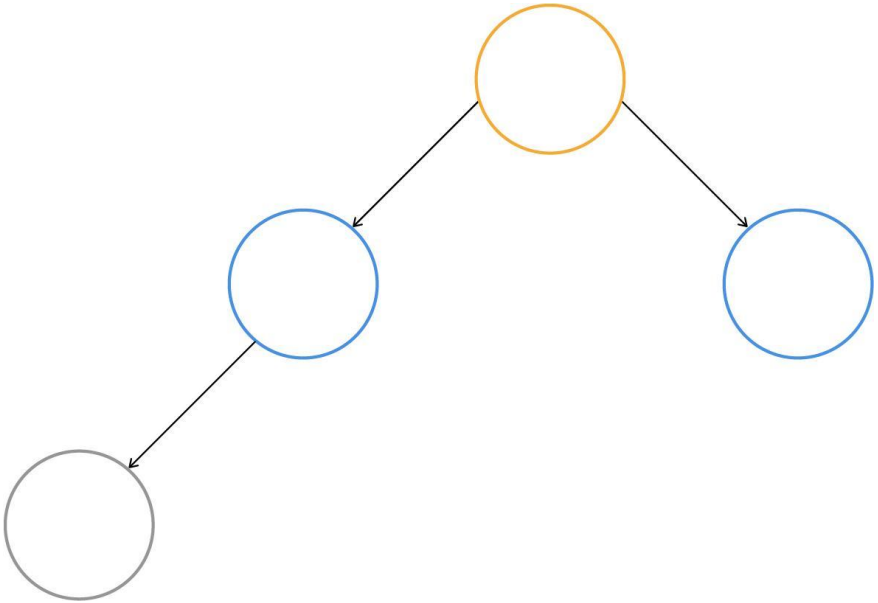
Complete binary tree



Full and Complete binary tree

Binary Tree

upGrad



Almost complete binary tree

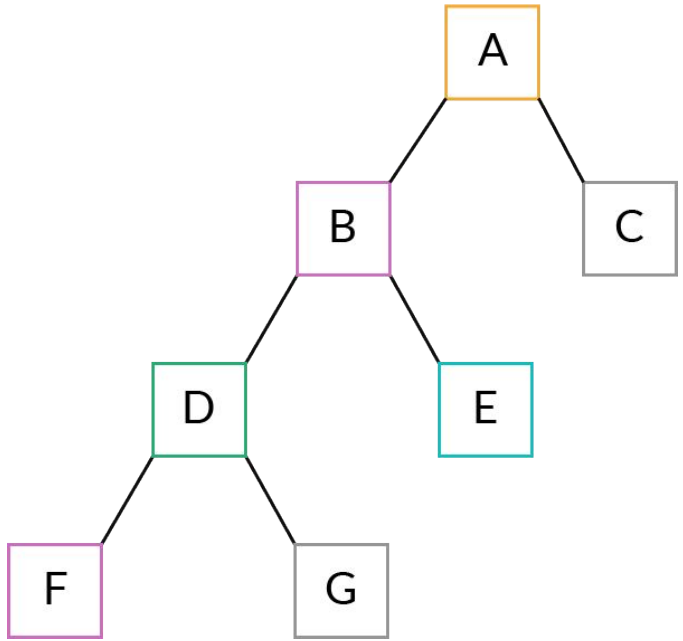
Binary Tree

upGrad

It's your turn now. Are these binary trees:

Complete?

Strictly binary?



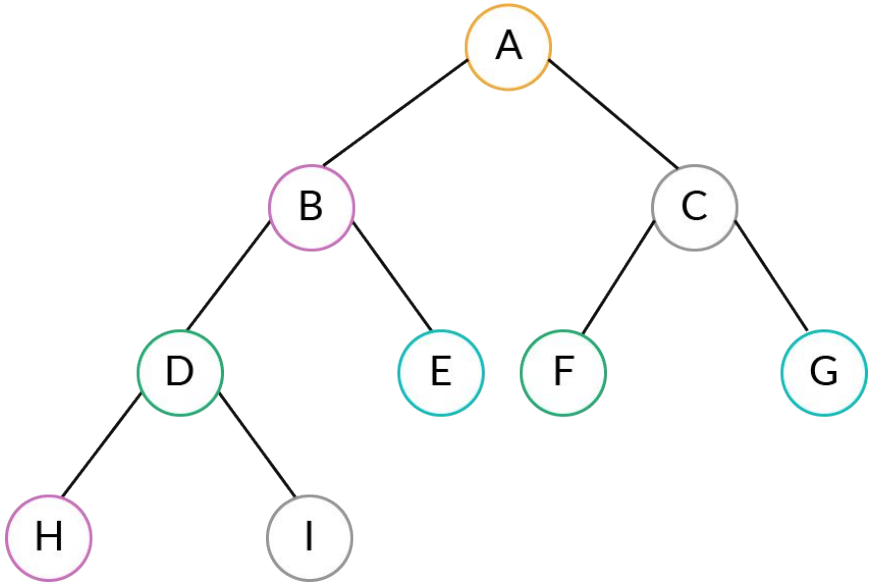
Binary Tree

upGrad

It's your turn now. Are these binary trees:

Complete?

Strictly binary?



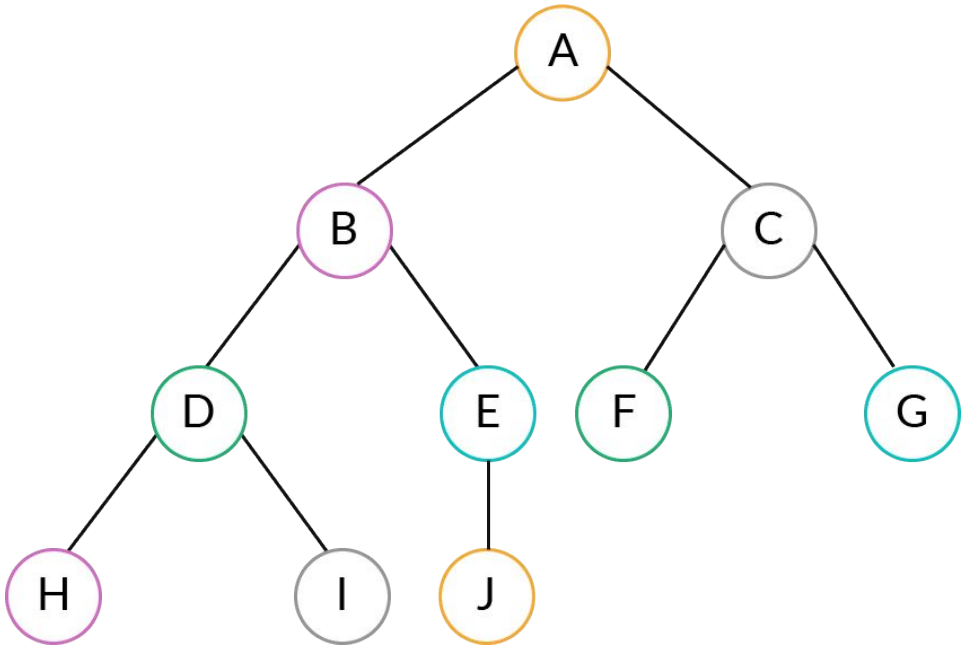
Binary Tree

upGrad

It's your turn now. Are these binary trees:

Complete?

Strictly binary?



Binary Tree

upGrad

Now, let's try to figure out a bit of mathematics:

- In a binary tree, what is the maximum number of nodes at the 'k'th level?
- What is the maximum number of nodes in a binary tree of height 'h'?
- What is the minimum number of nodes that can be present in a complete binary tree of height 'h'?

Binary Tree

upGrad

Now that we know what a binary tree is, let us see how we can represent the same in our data structures.

Binary trees can be traditionally represented in two ways:

1. With **arrays**, which is also known as the sequential representation.
2. A **linked** representation, using nodes and pointers, similar to the ones we used while discussing Linked Lists.

Array representation:

In this representation, we define an array of size 2^h-1, if 'h' is the height of the binary tree. The root is placed at the first array position. Subsequently, for any node stored at the 'x'th index:

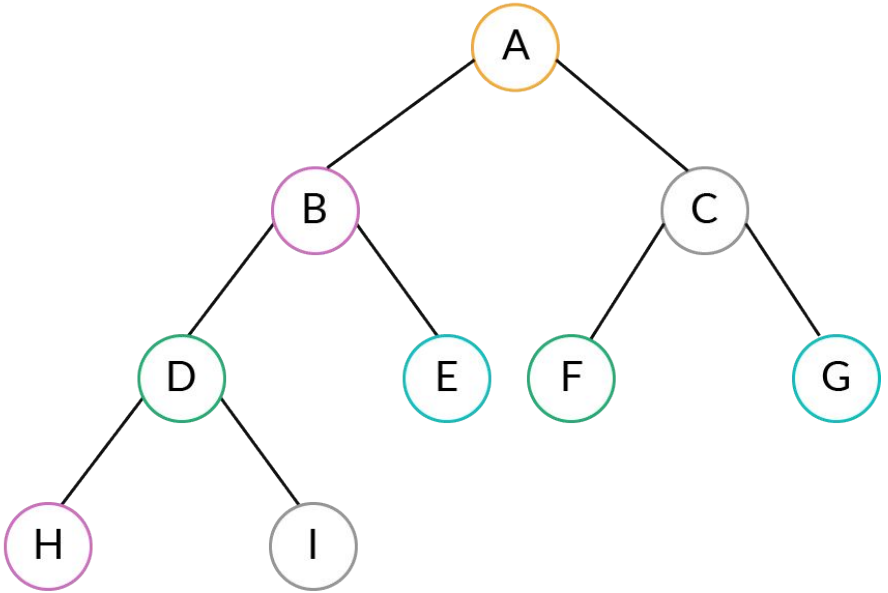
- The left child of the node will be at an array index of 2*x + 1.
- The right child of the node will be at an array index of 2*x + 2.
- The parent node will be at an array index position of (x-1)/2 if x is odd and at an array index position of (x-2)/2 if x is even.

Binary Tree

upGrad

Array representation:

A
B
C
D
E
F
G
H
I



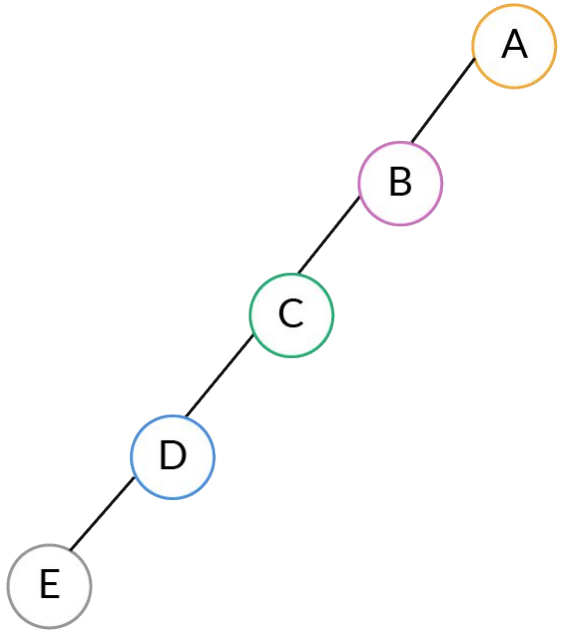
Binary Tree

upGrad

Array representation:

We can see, as a tree trends towards being degenerate, there is tremendous wastage of storage space!

A
B
-
C
-
-
-
D
-
.
E



Poll 6 (15 Sec.)

The left child of the binary tree node will be at an array index of 2*x + 1.

1. True
2. False

Poll 6 (Answer)

The left child of the binary tree node will be at an array index of 2*x + 1.

1. **True**
2. False

Binary Tree

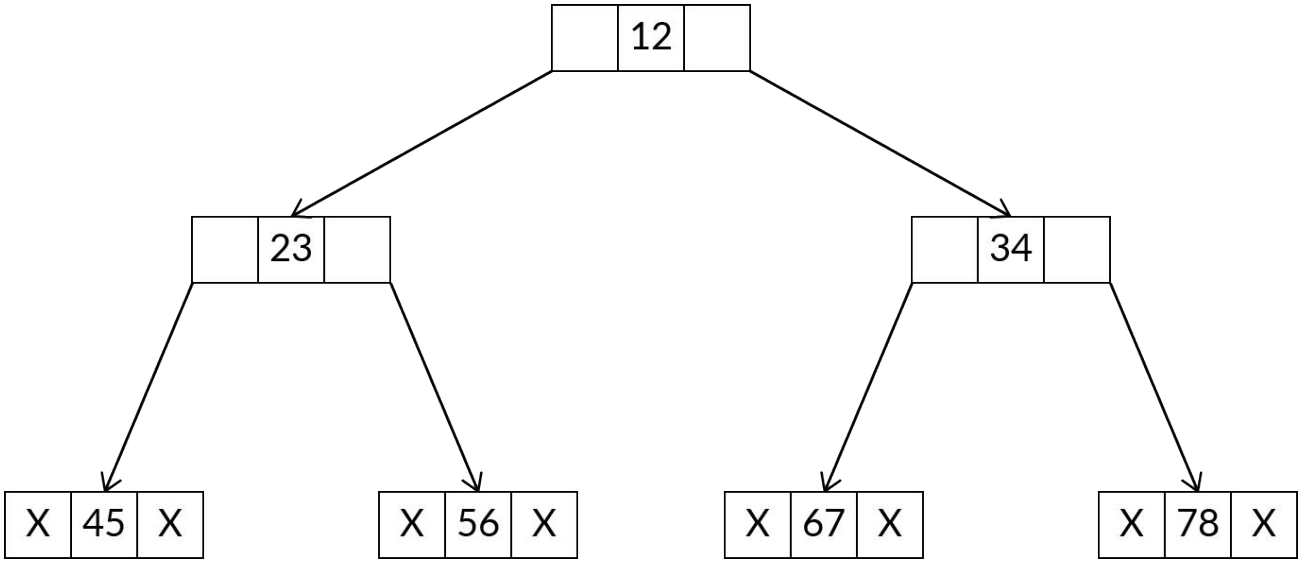
upGrad

The problem of storage wastage posed by an array representation of a binary tree is solved by making use of a linked representation. In this representation, we define an abstract data type (Nodes) like the ones we did while creating Linked Lists.

However, there is a slight difference now. Instead of having a pointer to the 'next' sequential node, we will now have pointers to the left and the right child of the binary tree.

The left and the right child pointers of these nodes will be ***null***.

As you can see, there will be no wastage of storage space as we create nodes that are equal in number to the data elements that we have.



Binary Tree

upGrad

Now that you know how to represent a binary tree, you will now learn how to go about traversing them. Conventionally, we are interested in three type of binary tree traversals:

- Inorder traversal
- Preorder traversal
- Postorder traversal

Inorder Traversal

Inorder traversal follows a very simple algorithm. Whenever we encounter a node, we:

- Perform an inorder traversal of the left sub-tree of the node.
- ‘Traverse’ or ‘process’ the node in consideration.
- Perform an inorder traversal of the right sub-tree of the node.

Now, can anyone tell which algorithmic construct we shall be revisiting here?

upGrad

Inorder Traversal

Let's take a look at a high-level algorithm for inorder traversal.

```
void inorder (Node root)
{
      if (root != null) {
            inorder (root -> left);
            process (root);
            inorder (root -> right);
      }
}
```

Preorder and Postorder Traversals

Preorder and Postorder traversals work very similarly to Inorder traversal, with the same subtle difference that prefix and postfix expressions had from infix expressions.

So, when you are required to perform a **Preorder** traversal, and you come across a node:

- 'Traverse' or 'process' the node in consideration.
- Perform a preorder traversal of the left sub-tree of the node.
- Perform an preorder traversal of the right sub-tree of the node.

Binary Tree

upGrad

Similarly, when you are required to perform **Postorder** traversal, and you come across a node:

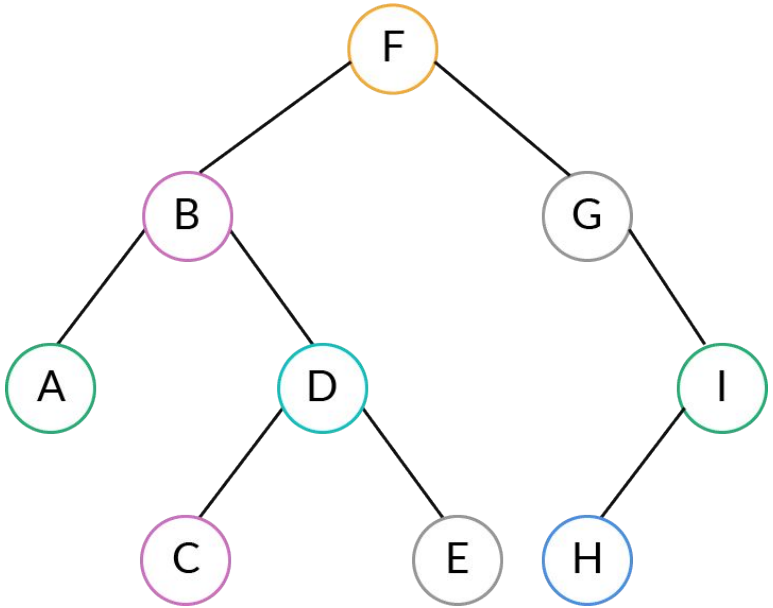
- Perform an postorder traversal of the left sub-tree of the node.
- Perform an postorder traversal of the right sub-tree of the node.
- ‘Traverse’ or ‘process’ the node in consideration.

Now, let us practice traversals a bit with a few examples:

Preorder: F B A D C E G I H

Inorder: A B C D E F G H I

Postorder: A C E D B H I G F



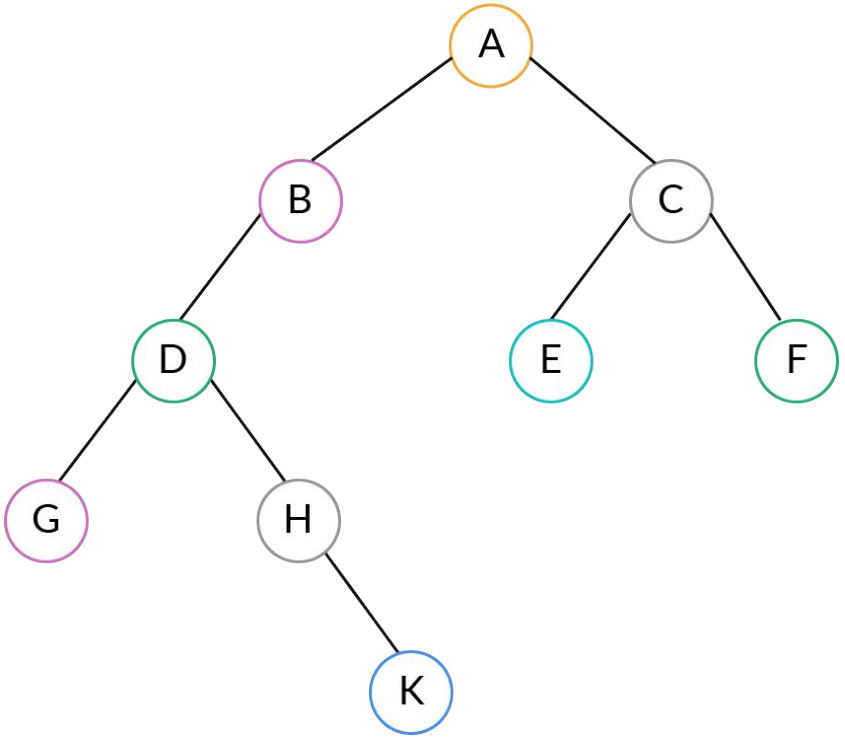
Binary Tree

upGrad

Preorder:

Inorder:

Postorder:



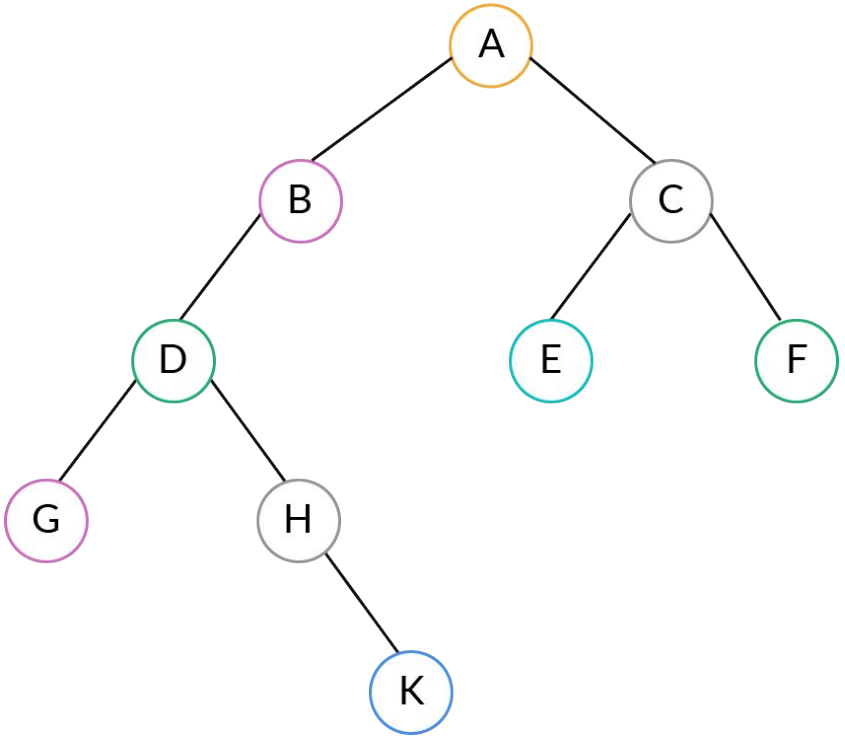
Binary Tree

upGrad

Preorder: A B D G H K C E F

Inorder: G D H K B A E C F

Postorder: G K H D B E F C A



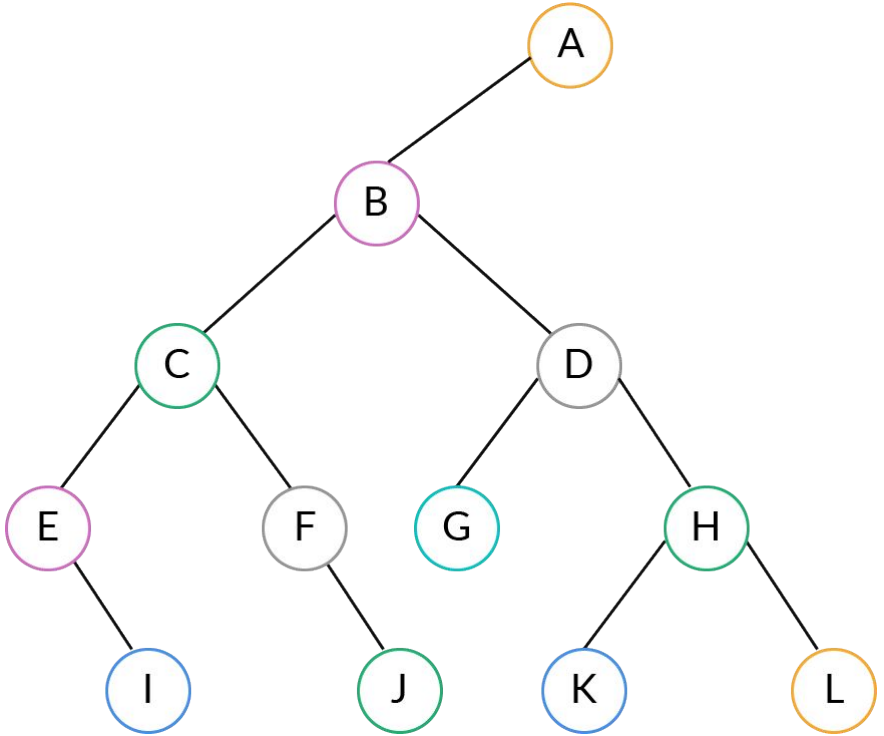
Binary Tree

upGrad

Preorder:

Inorder:

Postorder:



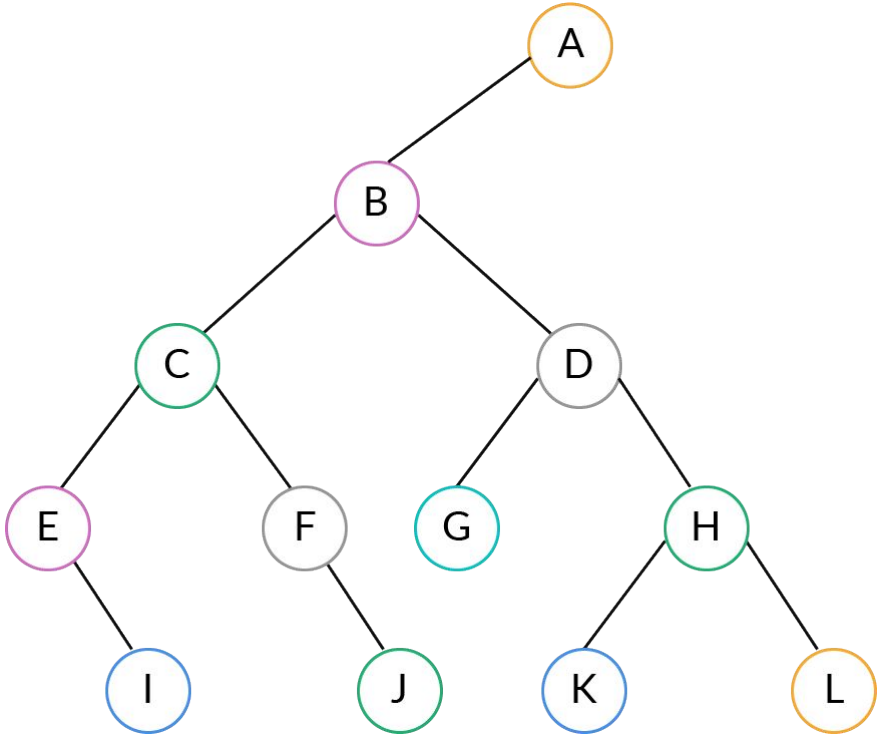
Binary Tree

upGrad

Preorder: A B C E I F J D G H K L

Inorder: E I C F J B G D K H L A

Postorder: I E J F C G K L H D B A



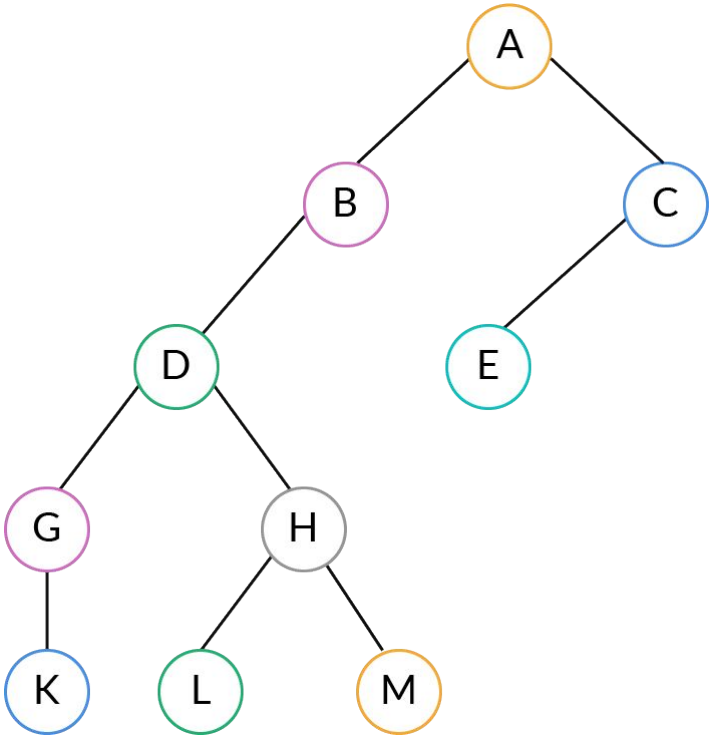
Binary Tree

upGrad

Preorder:

Inorder:

Postorder:



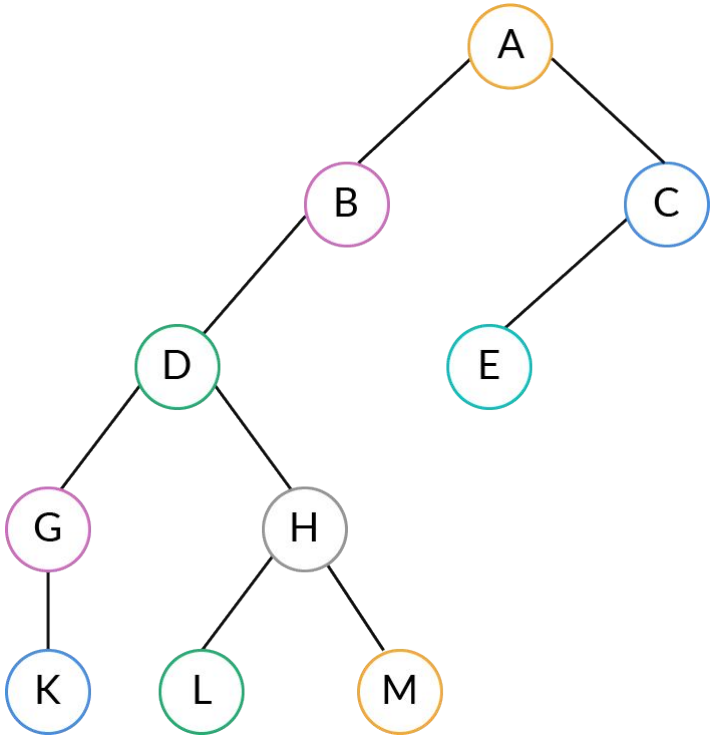
Binary Tree

upGrad

Preorder: A B D G K H L M C E

Inorder: K G D L H M B A E C

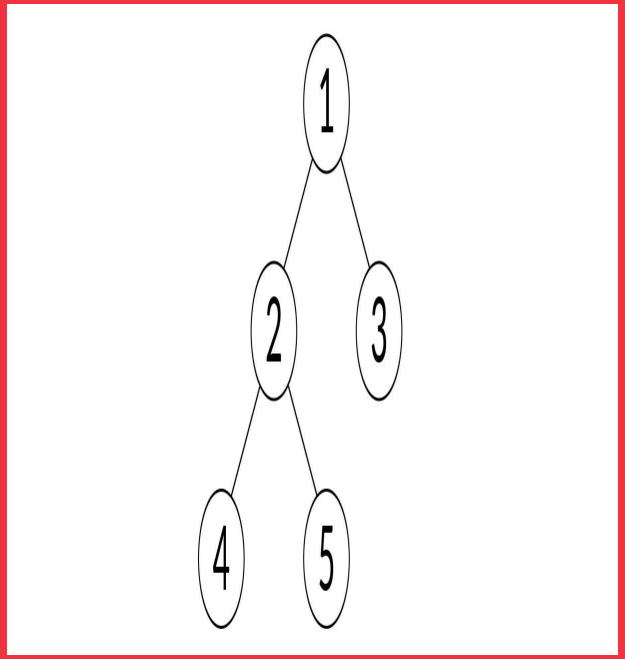
Postorder: K G L M H D B E C A



Homework

You are given a binary tree. You need to print the maximum height of the binary tree. If the tree is NULL (empty tree), print the height of the tree as 0.

Output: 3
Here, in the given tree, the maximum height of the binary tree is 3.



Tasks to complete after the session

Homework
MCQs
Coding Questions

upGrad

In the next class...

- BST

upGrad

#RahoAmbitious

Thank You!