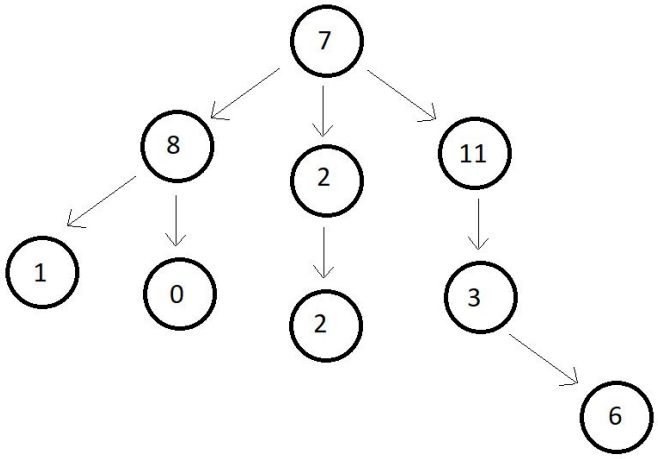
Introduction to Trees

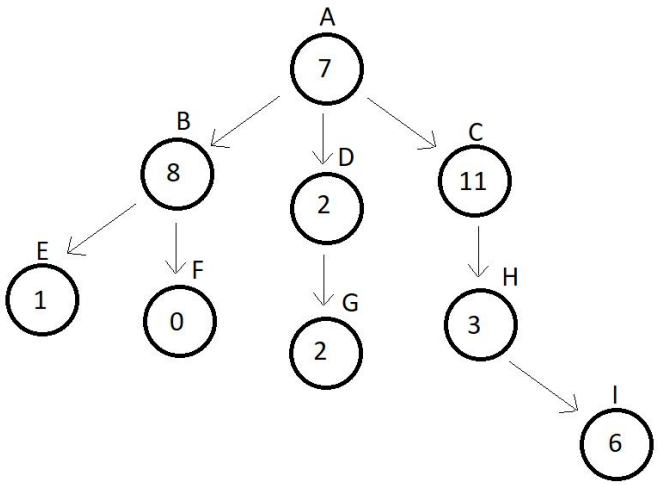
codewithharry.com/videos/data-structures-and-algorithms-in-hindi-61

In the wake of such an extensive and lengthy study of sorting algorithms, we would now like to move on to the study of some more advanced data structures. Today, we'll begin discussing trees. It is unlikely that I won't be able to explain trees to you after I finish this lecture. The flow of the concept will remain the same, first the theory and then their applications and problems.

A tree usually represents the hierarchy of elements and depicts the relationships between the elements. Trees are considered as one of the largely used facets of data structures. To give you a better idea of how a tree looks like, let me give an example:



Every circle represents a node, and every arrow represents the hierarchy. For you to be able to understand the terminology associated with trees, I will further name these nodes.



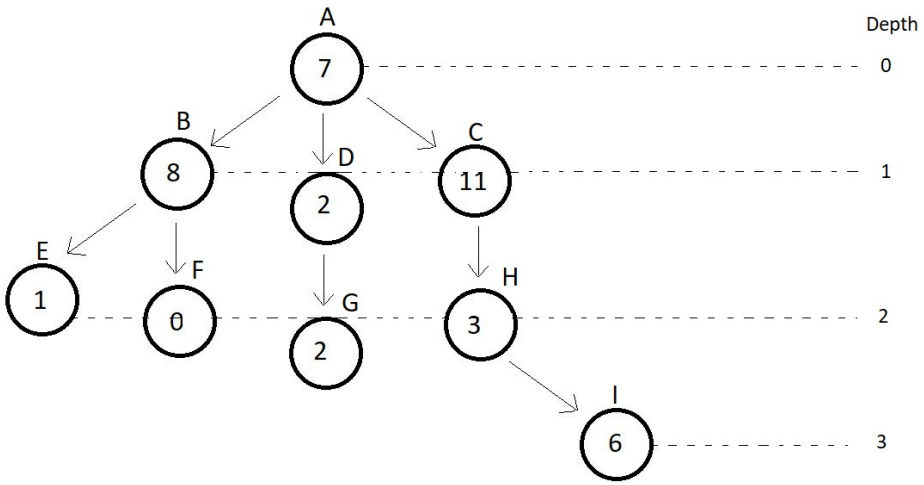
Now, you can very easily say that node C is the child of node A or node B is the parent of node E. You would be wondering what a parent or child is. I was coming to that only.

Terminologies used in trees:

1. **Root:** The topmost node of a tree is called the root. There is no edge pointing to it, but one or more than one edge originating from it. Here, A is the root node.
2. **Parent:** Any node which connects to the child. Node which has an edge pointing to some other node. Here, C is the parent of H.
3. **Child:** Any node which is connected to a parent node. Node which has an edge pointing to it from some other node. Here, H is the child of C.
4. **Siblings:** Nodes belonging to the same parent are called siblings of each other. Nodes B, C and D are siblings of each other, since they have the same parent node A.
5. **Ancestors:** Nodes accessible by following up the edges from a child node upwards are called the ancestors of that node. Ancestors are also the parents of the parents of ..... that node. Here, nodes A, C and H are the ancestors of node I.

6. **Descendants:** Nodes accessible by following up the edges from a parent node downwards are called the descendants of that node. Descendants are also the child of the child of ..... that node. Here, nodes H and I are the descendants of node C.

7. **Leaf/ External Node:** Nodes which have no edge originating from it, and have no child attached to it. These nodes cannot be a parent. Here, nodes E, F, G and I are leaf nodes.
8. **Internal node:** Nodes with at least one child. Here, nodes B, D and C are internal nodes.
9. **Depth:** Depth of a node is the number of edges from root to that node. Here, the depth of nodes A, C, H and I are 0, 1, 2 and 3 respectively.



10. **Height:** Height of a node is the number of edges from that node to the deepest leaf. Here, the height of node A is 3, since the deepest leaf from this node is node I. And similarly, height of node C is 2.

There are still a lot of concepts left. To help you get a better understanding of trees, I presented this brief introduction. These were the basics you had to learn to move to the advanced topics. To understand what lies ahead, you need to learn the terminologies. Make sure you do that at least.

I appreciate your support throughout. I hope you enjoyed the tutorial. If you genuinely appreciate my work, please let your friends know about this course too. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. See you all in the next tutorial where we'll try learning a special variety of trees called the **Binary Tree**. Till then keep coding.