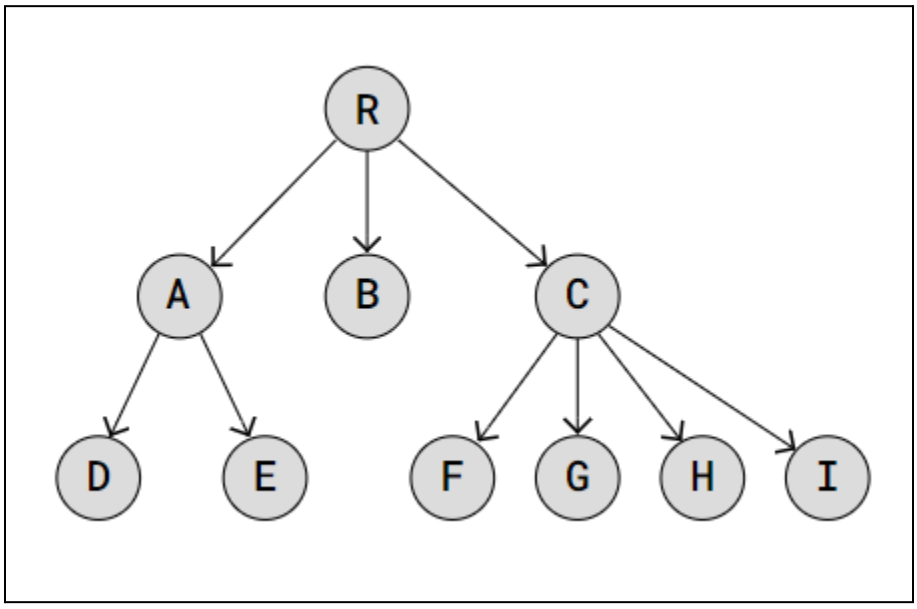
Tree

The Tree data structure is similar to Linked Lists in that each node contains data and can be linked to other nodes.

We have previously covered data structures like Arrays, Linked Lists, Stacks, and Queues. These are all linear structures, which means that each element follows directly after another in a sequence. Trees, however, are different. In a Tree, a single element can have multiple 'next' elements, allowing the data structure to branch out in various directions.

The data structure is called a "tree" because it looks like a tree, only upside down, just like in the image below.



The Tree data structure can be useful in many cases:

- Hierarchical Data: File systems, organizational models, etc.
- Databases: Used for quick data retrieval.
- Routing Tables: Used for routing data in network algorithms.
- Sorting/Searching: Used for sorting data and searching for data.
- Priority Queues: Priority queue data structures are commonly implemented using trees, such as binary heaps.

Tree Terminology and Rules

1. **Root**: The first node in a tree is called the **root node**.
2. **Edge**: A link connecting one node to another is called an **edge**.
3. **Nodes:** A **parent** node has links to its **child** nodes. Another word for a parent node is an **internal** node.
4. A node can have zero, one, or many child nodes.
5. A node can only have one parent node.

6. **Leaves or Leaf Nodes**: Nodes without links to other child nodes are called **leaves, or leaf nodes**.
7. **Height of the tree**: The **tree height** is the maximum number of edges from the root node to a leaf node.
8. The height of a node is the maximum number of edges between the node and a leaf node.
9. **Tree size:** The **tree size** is the number of nodes in the tree.

Properties of Tree -

1. The data is **not** arranged linearly.
2. The data is arranged in **different levels** from top to bottom.
3. Each element of the tree is known as a **node**. Remember, we defined a similar type of node in the linked list as well.
4. The nodes **at the upper level act as ancestors** and the nodes **at the lower are the children** of these ancestors.
5. At the **last level** of the tree, the nodes **do no**t have any children.
6. At the **first level**, there is only one node and it **does no**t have any ancestors.
7. Each node can have **any** number of children.

Types of Trees

Trees are a fundamental data structure in computer science, used to represent hierarchical relationships. This tutorial covers several key types of trees.

Binary Trees: Each node has up to two children, the left child node and the right child node. This structure is the foundation for more complex tree types like Binay Search Trees and AVL Trees.

Binary Search Trees (BSTs): A type of Binary Tree where for each node, the left child node has a lower value, and the right child node has a higher value.

AVL Trees: A type of Binary Search Tree that self-balances so that for every node, the difference in height between the left and right subtrees is at most one. This balance is maintained through rotations when nodes are inserted or deleted.