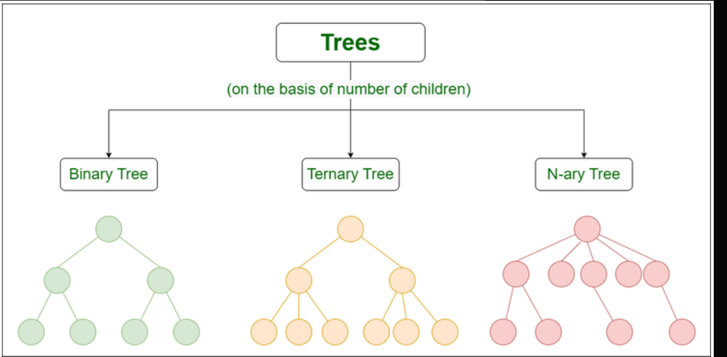
TREES

* Trees are non-linear data types because the data elements connect to each other hierarchically.
* The level is the horizontal lines that can be drawn crossing the tree.
* Preorder 🡪 node 🡨🡪left 🡨🡪right. When sequence is given then go according to the order.
* The tree's ancestors are the parent, its parent, its parent till it reaches the root.
* searching in a binary search tree has the worst case complexity of O(n). In general, the time complexity is O(h) where h is the height of BST.

**Types of Trees🡪**

1. Binary tree: In a binary tree, each node can have a maximum of two children linked to it. Some common types of binary trees include full binary trees, complete binary trees, balanced binary trees, and degenerate or pathological binary trees.
2. Ternary Tree: A Ternary Tree is a tree data structure in which each node has at most three child nodes, usually distinguished as “left”, “mid” and “right”.
3. N-ary Tree or Generic Tree: Generic trees are a collection of nodes where each node is a data structure that consists of records and a list of references to its children(duplicate references are not allowed). Unlike the linked list, each node stores the address of multiple nodes.

**Properties 🡪**

1. **Number of edges**: An edge can be defined as the connection between two nodes. If a tree has N nodes then it will have (N-1) edges. There is only one path from each node to any other node of the tree.
2. Depth of a node: The depth of a node is defined as the length of the path from the root to that node. Each edge adds 1 unit of length to the path. So, it can also be defined as the number of edges in the path from the root of the tree to the node.
3. **Height of a node**: The height of a node can be defined as the length of the longest path from the node to a leaf node of the tree. Height of the Tree: The height of a tree is the length of the longest path from the root of the tree to the leaf node of the tree.
4. **Degree of a Node**: The total count of subtrees attached to that node is called the degree of the node. The degree of a leaf node must be 0. The degree of a tree is the maximum degree of a node among all the nodes in the tree.

**Why Do We Need Trees**

1. 1. One reason to use trees might be because you want to store information that naturally forms a hierarchy. For example, the file system on a computer: File System
2. Trees (with some ordering e.g., BST) provide moderate access/search (quicker than Linked List and slower than arrays).
3. Trees provide moderate insertion/deletion (quicker than Arrays and slower than Unordered Linked Lists).
4. Like Linked Lists and unlike Arrays, Trees don’t have an upper limit on the number of nodes as nodes are linked using pointers.
5. Store hierarchical data, like folder structure, organization structure, XML/HTML data.
6. [Binary Search Tree](https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/) is a tree that allows fast search, insert, delete on a sorted data. It also allows finding closest item
7. [Heap](https://www.geeksforgeeks.org/heap-data-structure/) is a tree data structure which is implemented using arrays and used to implement priority queues.
8. [B-Tree](https://www.geeksforgeeks.org/b-tree-set-1-introduction-2/) and[B+ Tree](https://www.geeksforgeeks.org/database-file-indexing-b-tree-introduction/) : They are used to implement indexing in databases.
9. [Syntax Tree](https://www.geeksforgeeks.org/compiler-design-syntax-directed-translation/):  Scanning, parsing , generation of code and evaluation of arithmetic expressions in Compiler design.
10. [K-D Tree:](https://www.geeksforgeeks.org/k-dimensional-tree/)A space partitioning tree used to organize points in K dimensional space.
11. [Trie](https://www.geeksforgeeks.org/trie-insert-and-search/) : Used to implement dictionaries with prefix lookup.
12. [Suffix Tree](https://www.geeksforgeeks.org/pattern-searching-set-8-suffix-tree-introduction/) : For quick pattern searching in a fixed text.
13. [Spanning Trees](https://www.geeksforgeeks.org/applications-of-minimum-spanning-tree/) and shortest path trees are used in routers and bridges respectively in computer networks

**Applications of trees**

1. File System: This allows for efficient navigation and organization of files.
2. Data Compression: Huffman coding is a popular technique for data compression that involves constructing a binary tree where the leaves represent characters and their frequency of occurrence. The resulting tree is used to encode the data in a way that minimizes the amount of storage required.
3. Compiler Design: In compiler design, a syntax tree is used to represent the structure of a program.
4. Database Indexing: B-trees and other tree structures are used in database indexing to efficiently search for and retrieve data.

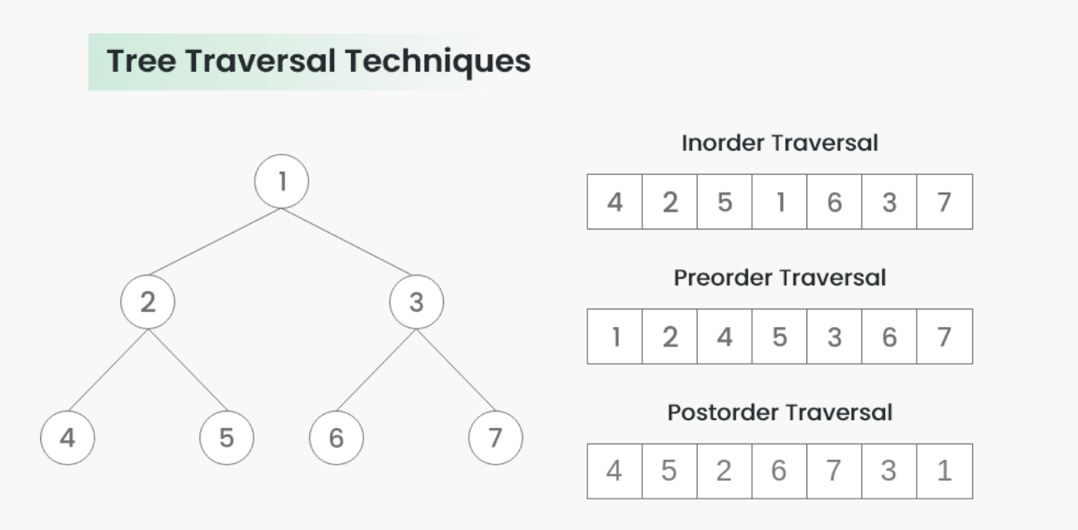
**Advantages of trees**

1. Trees provide a hierarchical representation of data, making it easy to organize and navigate large amounts of information.
2. The recursive nature of trees makes them easy to traverse and manipulate using recursive algorithms.
3. **Efficient searching:** Trees are particularly efficient for searching and retrieving data. The time complexity of searching in a tree is typically O(log n), which means that it is very fast even for very large data sets.
4. **Flexible size:** Trees can grow or shrink dynamically depending on the number of nodes that are added or removed. This makes them particularly useful for applications where the data size may change over time.
5. **Easy to traverse:**Traversing a tree is a simple operation, and it can be done in several different ways depending on the requirements of the application. This makes it easy to retrieve and process data from a tree structure.
6. **Easy to maintain:**Trees are easy to maintain because they enforce a strict hierarchy and relationship between nodes. This makes it easy to add, remove, or modify nodes without affecting the rest of the tree structure.
7. **Natural organization:**Trees have a natural hierarchical organization that can be used to represent many types of relationships. This makes them particularly useful for representing things like file systems, organizational structures, and taxonomies.
8. **Fast insertion and deletion**: Inserting and deleting nodes in a tree can be done in O(log n) time, which means that it is very fast even for very large trees.

**Disadvantages of Tree:**

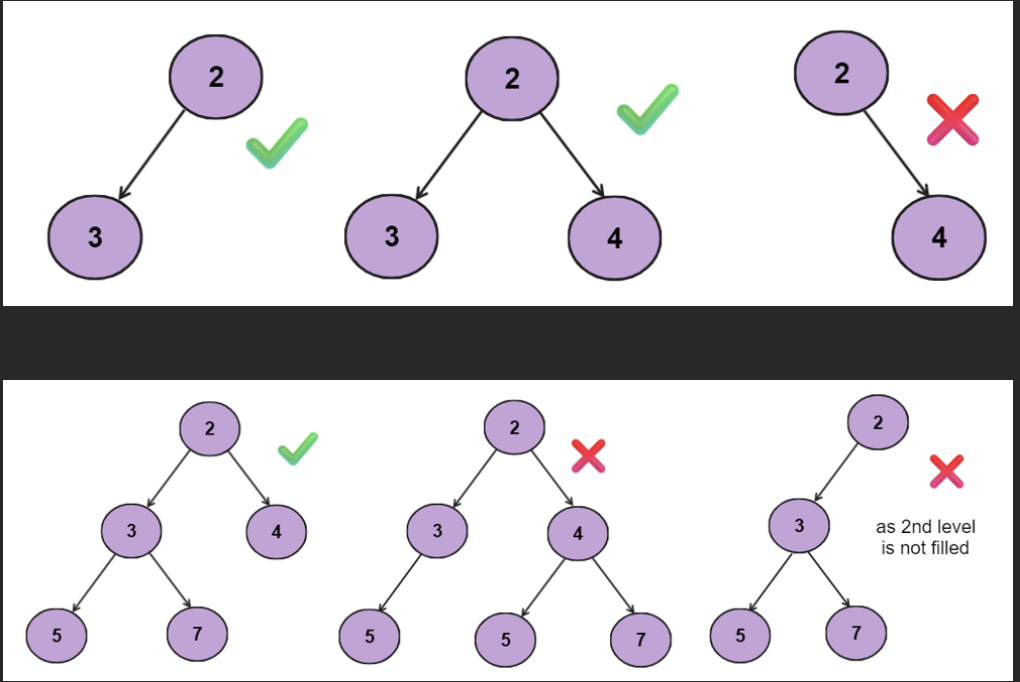
1. **Memory overhead**: Trees can require a significant amount of memory to store, especially if they are very large. This can be a problem for applications that have limited memory resources.
2. **Imbalanced trees:** If a tree is not balanced, it can result in uneven search times. This can be a problem in applications where speed is critical.
3. **Complexity**: Trees can be complex data structures, and they can be difficult to understand and implement correctly. This can be a problem for developers who are not familiar with them.
4. **Limited flexibility**: While trees are flexible in terms of size and structure, they are not as flexible as other data structures like hash tables. This can be a problem in applications where the data size may change frequently.
5. **Inefficient for certain operations:** While trees are very efficient for searching and retrieving data, they are not as efficient for other operations like sorting or grouping. For these types of operations, other data structures may be more appropriate.

**Traversal Techniques 🡪**

****

**Binary Search Tree**

* **Complete binary tree🡪**

****