Find your way here

CSC 250: Foundations of Computer Science I

Fall 2023 - Lecture 13

Amitabha Dey

Department of Computer Science
University of North Carolina at Greensboro

Tree Data Structure

A tree data structure is a hierarchical structure that is used to represent and organize data in a way that is easy to navigate and search.

It is a collection of nodes that are connected by edges and has a hierarchical relationship between the nodes.

The topmost node of the tree is called the root, and the nodes below it are called the child nodes.

Each node can have multiple child nodes, and these child nodes can also have their own child nodes, forming a recursive structure.

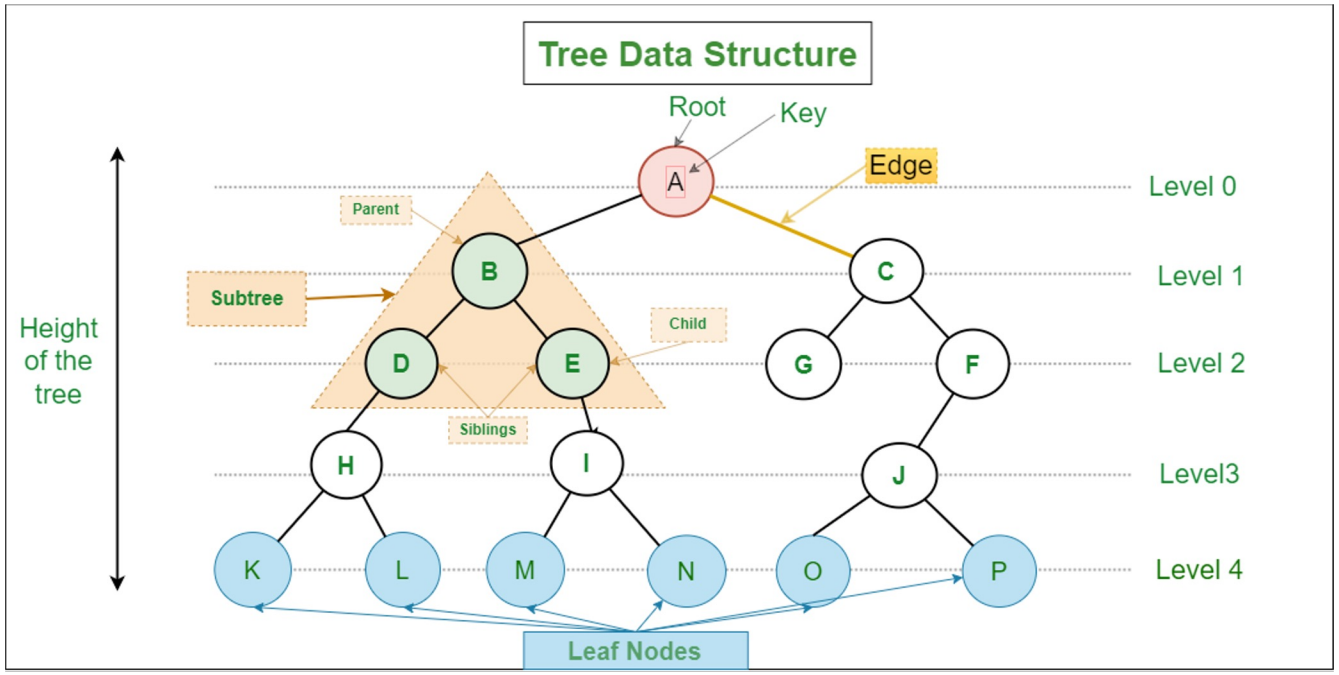


Image credit - https://www.geeksforgeeks.org/

Binary Tree

Binary Tree is defined as a tree data structure where each node has at most 2 children. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

Special types of Binary Trees

- A **full binary tree** is a special type of binary tree in which every parent node/internal node has either two or no children.
- A **complete binary tree** is a special type of binary tree where all the levels of the tree are filled completely except the lowest level nodes which are filled from as left as possible.
- A **degenerate binary tree** is a binary tree in data structure in which each parent node has only one child node associated with it. Such a tree will behave in the manner of linked list data structure.
- A **perfect binary tree** is a binary tree in which all interior nodes have two children and all leaves have the same depth or same level.
- A **balanced binary tree**, also referred to as a height-balanced binary tree, is defined as a binary tree in which the height of the left and right subtree of any node differ by not more than 1

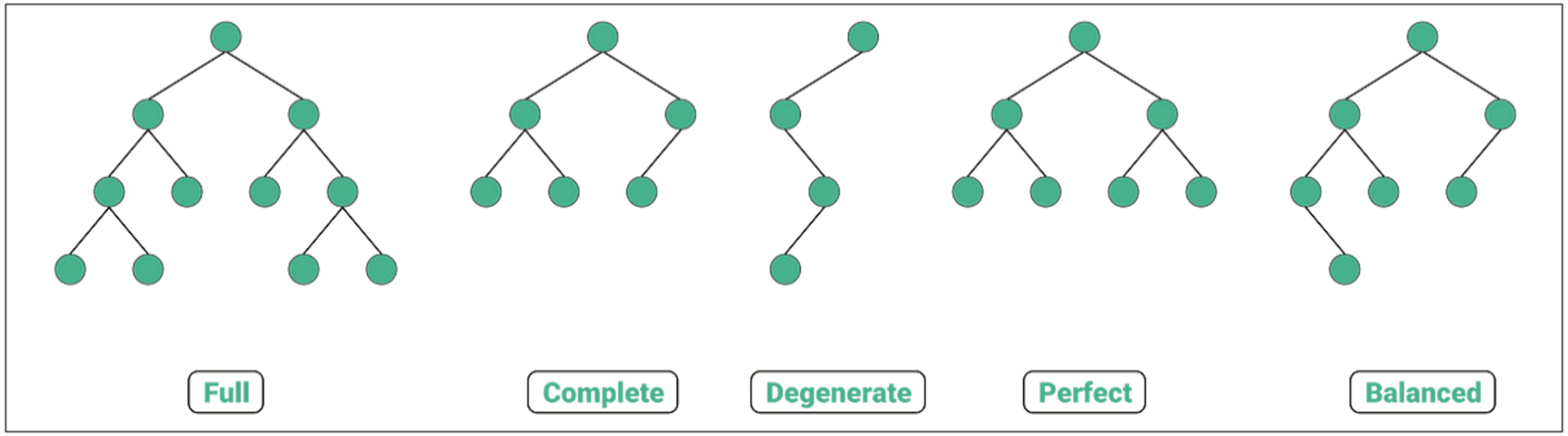
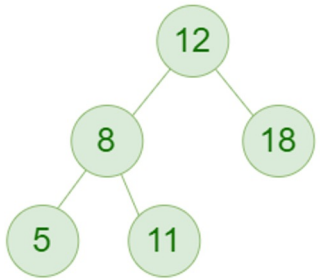


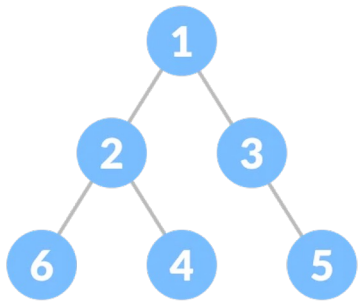
Image credit - https://www.geeksforgeeks.org/

Binary Tree

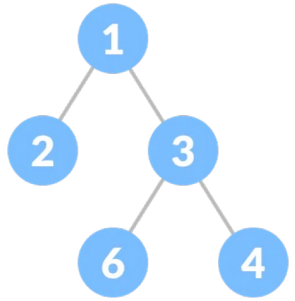
Full and Complete Binary Tree

Full Binary Tree

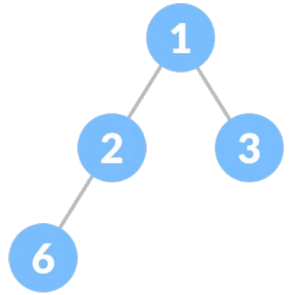




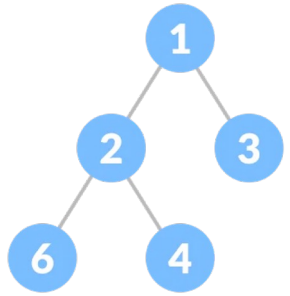
✗ Full Binary Tree
✗ Complete Binary Tree



✓ Full Binary Tree
✗ Complete Binary Tree

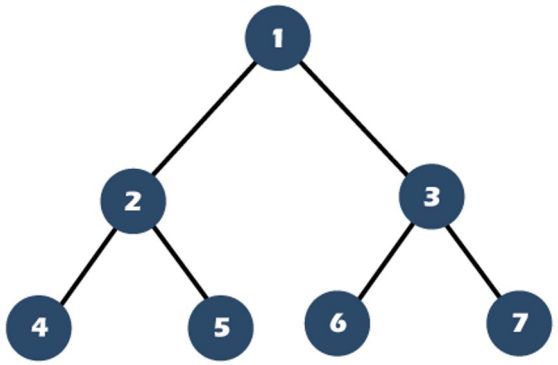


✗ Full Binary Tree
✓ Complete Binary Tree

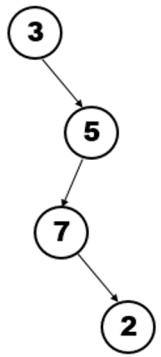


✓ Full Binary Tree
✓ Complete Binary Tree

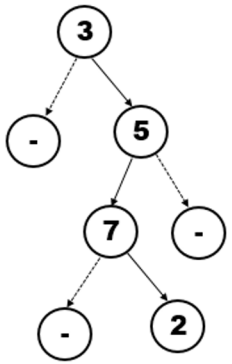
The below tree is a complete binary tree:



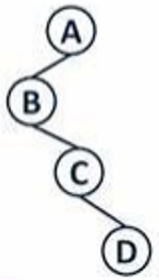
How it looks



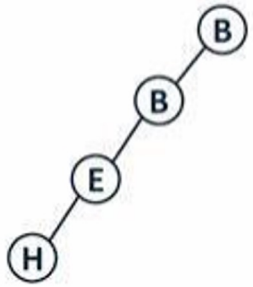
How it actually is



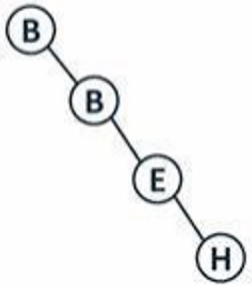
Pathological Tree



Left Skewed Tree

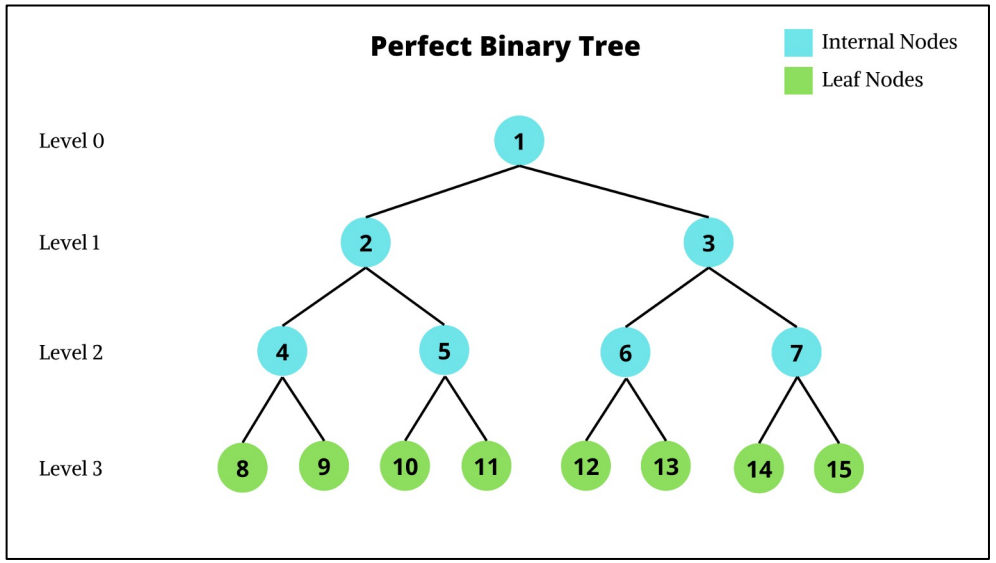


Right Skewed Tree

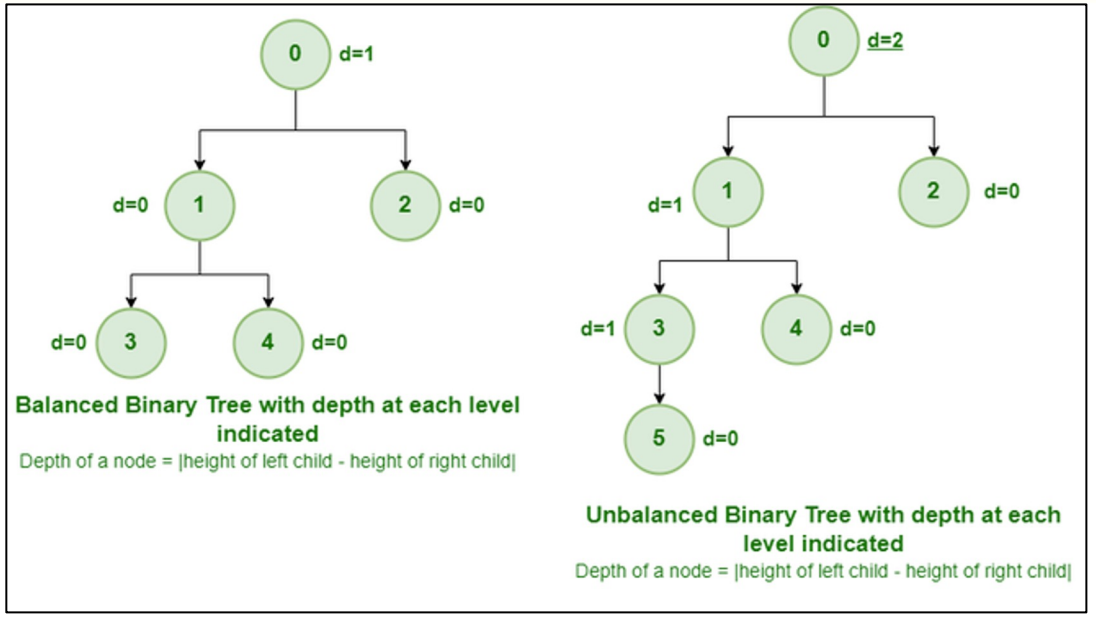


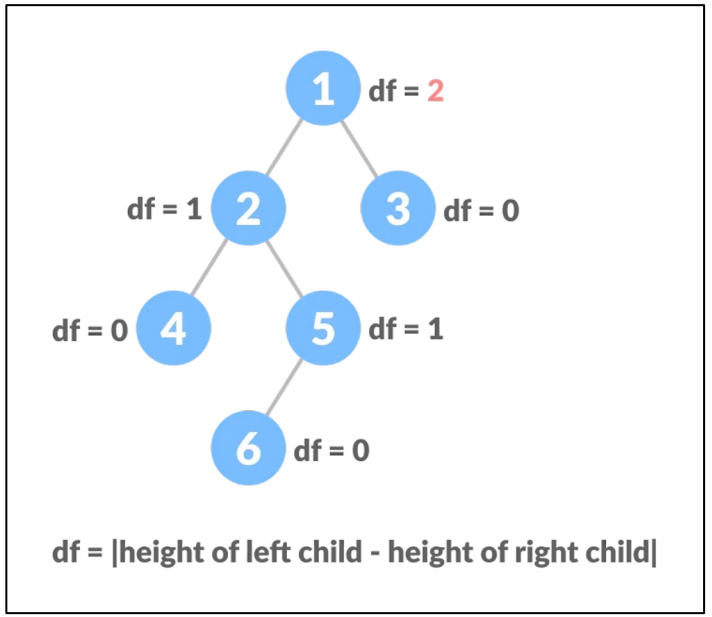
Binary Tree

Perfect Binary Tree and Balanced/Unbalanced Trees



A perfect binary tree is a type of binary tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.





A binary tree is called balanced if every leaf node is not more than a certain distance away from the root than any other leaf. That is, if we take any two leaf nodes (including empty nodes), the distance between each node and the root is approximately the same. In most cases, "approximately the same" means that the difference between the two distances (root to first leaf and root to second leaf) is not greater than 1, but the exact number can vary from application to application.

Tree vs Graph Data Structure

TREE	GRAPH
There exists a hierarchical structure, and the top node is called the root node.	The concept of hierarchy leading to a unique root node does not apply here.
Is an acyclic graph.	Cycles can exist.
Must be a connected graph.	Isn't necessarily a connected graph.
Data representation is similar to a tree, with concepts like branches, roots, and leaf nodes.	Data representation is similar to a network
Applications: decision trees, implementing heaps to find max/min numbers, sorting.	Applications: Finding shortest path, navigation, route optimization.

Image credit - https://www.interviewkickstart.com/

Comparison	Tree	Graph
Relationship of node	Only one root node. Parent-Child relationship exists.	No root node. No Parent-Child relationship exists.
Path	Only one path between two nodes	One or more paths exist between two nodes
Edge	N – 1 (N = Number of nodes)	Can not defined
Loop	Loop is not allowed	Loop is allowed
Traversal	Preorder, Inorder, Postorder	BFS, DFS
Model type	Hierarchical	Network

Image credit - https://hyosup0513.github.io/

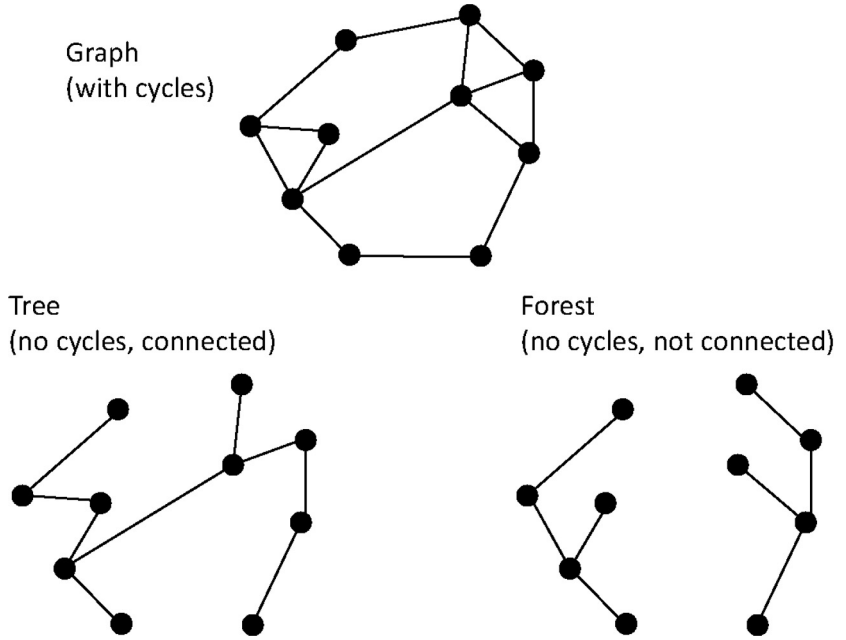
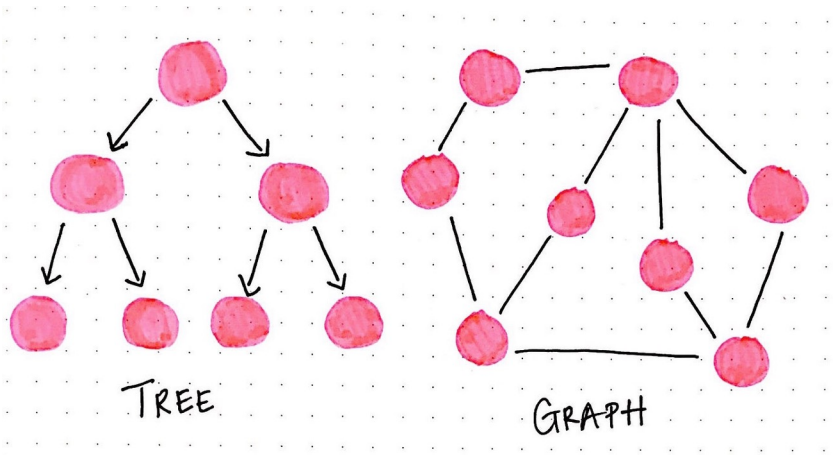


Image credit - https://www.cambridge.org/



Graph Traversal

Graph Traversal is the process of visiting every vertex and edge exactly once in a well-defined order.
During a traversal, the vertices that have been visited are tracked.
Tree traversal is a special case of graph traversal. A non-verbal description of three graph traversal algorithms: randomly, depth-first search, and breadth-first search.

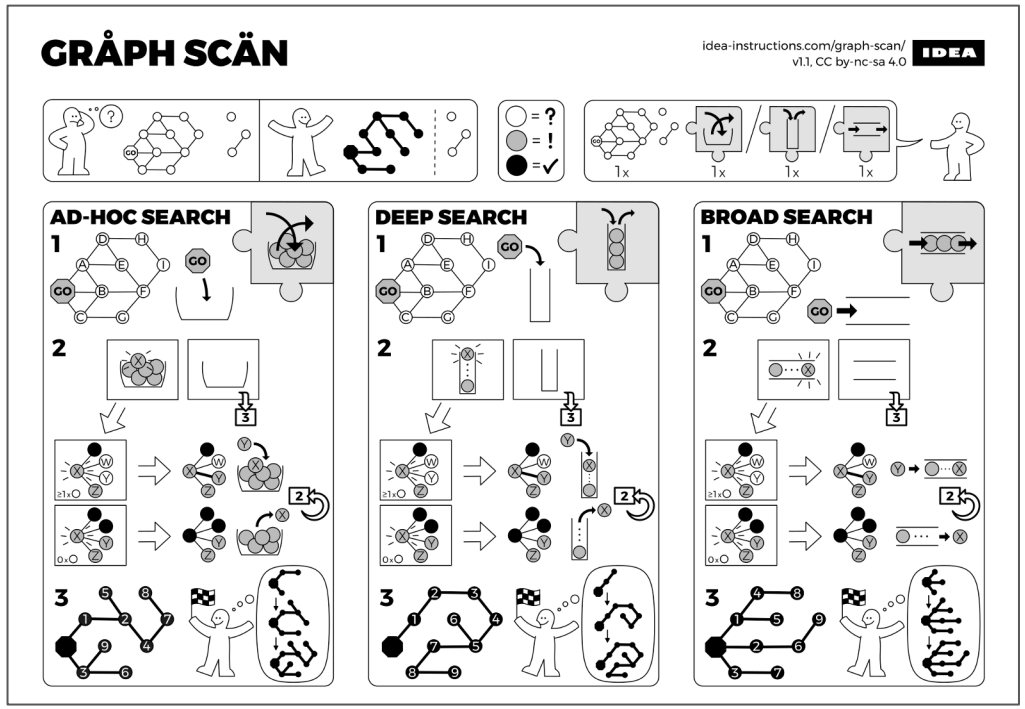


Image credit - https://en.wikipedia.org/

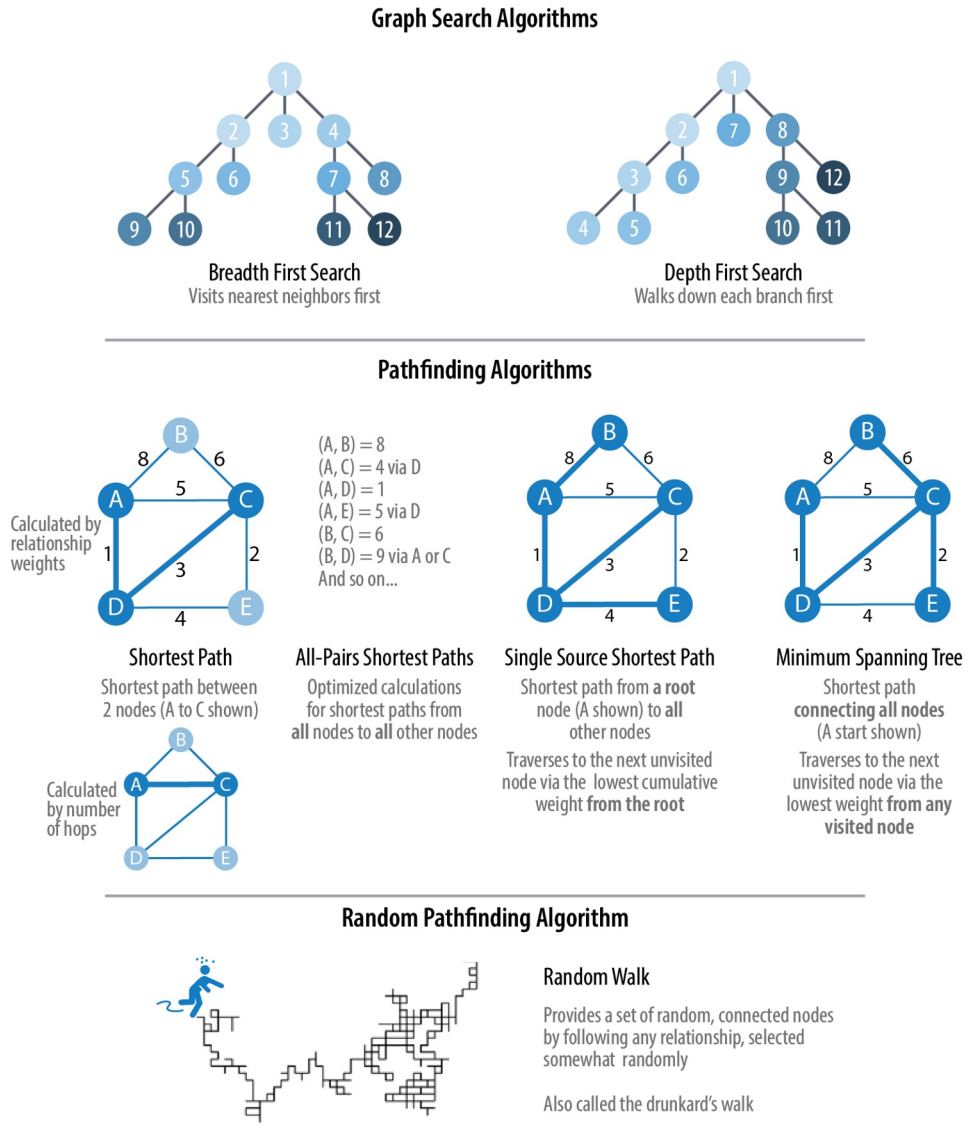


Image credit - https://www.oreilly.com/

Stack and Queue

Queue. A queue is the more intuitive of the two. It works much like waiting in line to buy a ticket. The first element to get to the data structure is the first one to be served.

Stack. A stack, however, works like a stack of trays in a cafeteria. You always grab from the top, not the bottom. This means you will have a relationship where the most recent element will be the first serviced.

- **Breadth-First Search (BFS)** uses a Queue data structure
(FIFO)

- **Depth-First Search (DFS)** uses a Stack data structure
(LIFO)

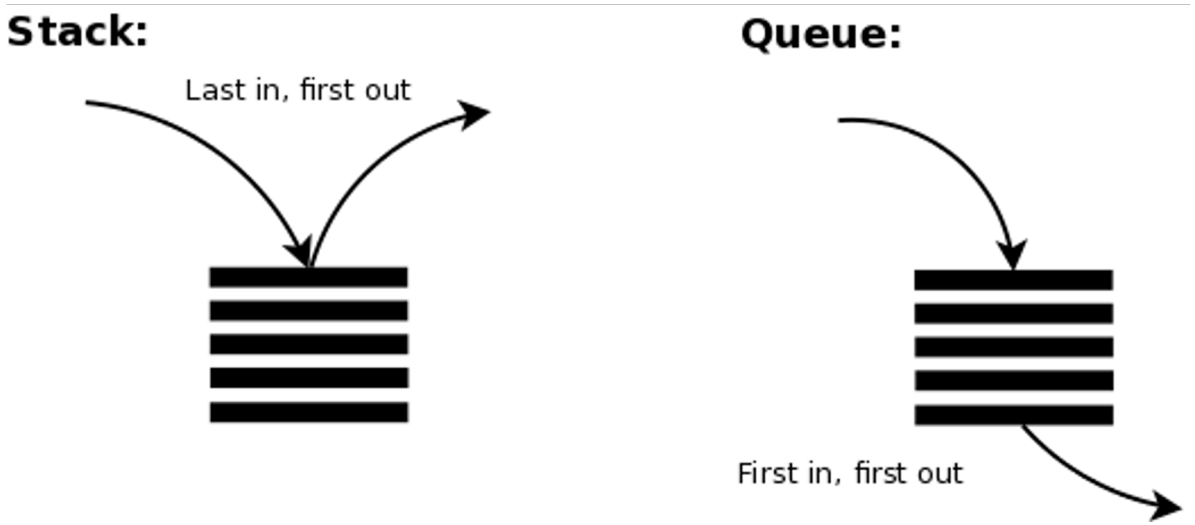


Image credit - https://gohighbrow.com/

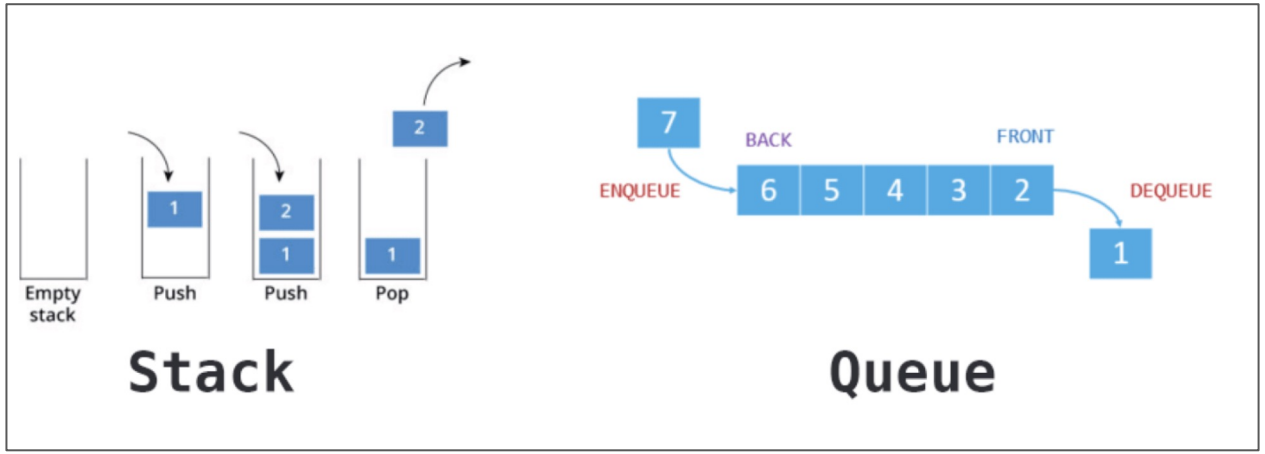


Image credit - https://dev.to/

Breadth-first Search (BFS)

Breadth-first search is an algorithm for traversing or searching tree or graph data structures. It starts at the tree's root or graph and searches/visits all nodes at the current depth level before moving on to the nodes at the next depth level. Breadth-first search can be used to solve many problems in graph theory.

The BFS algorithm works as follows:

1. Declare a queue and insert the starting vertex.
2. Initialize a visited array and mark the starting vertex as visited.
3. Follow the below process till the queue becomes empty:
   a. Remove the first vertex of the queue.
   b. Mark that vertex as visited.
   c. Insert all the unvisited neighbors of the vertex into the queue.

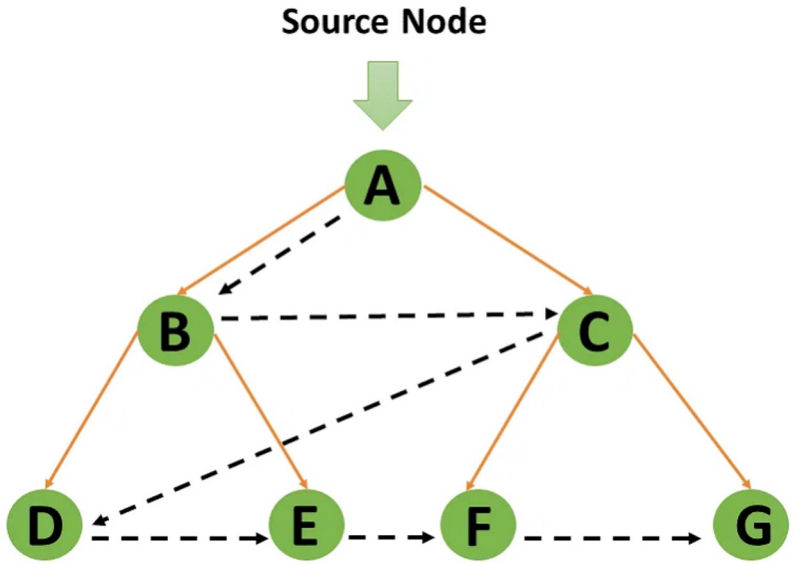


Image credit - https://www.simplilearn.com/

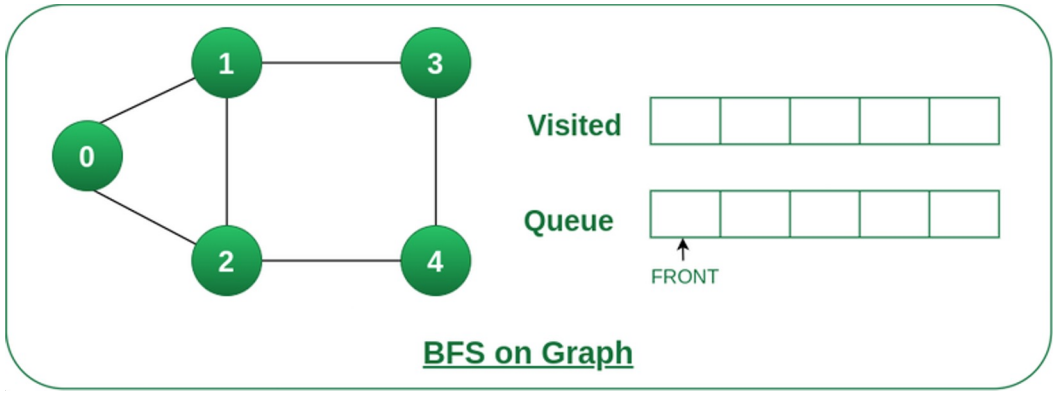


Image credit - https://www.geeksforgeeks.org/

Depth-first Search (DFS)

Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

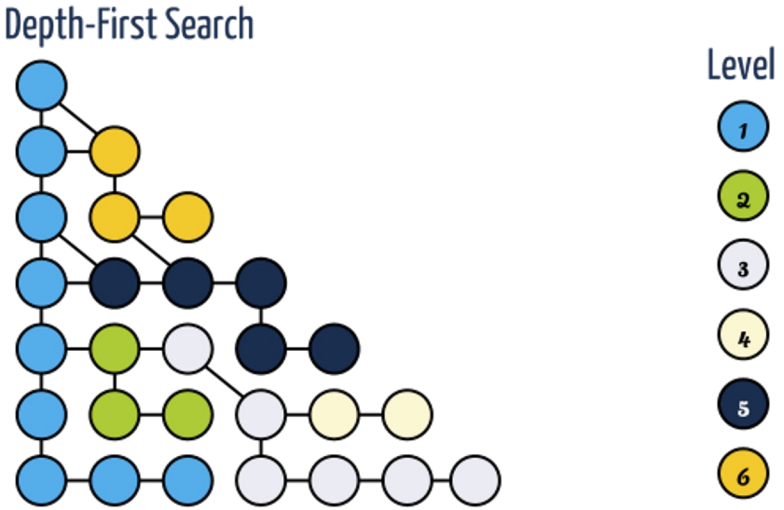


Image credit - http://stoimen.com/

The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.
4. Keep repeating steps 2 and 3 until the stack is empty.

The algorithm first goes in depth and then backtracks to all unvisited successors!

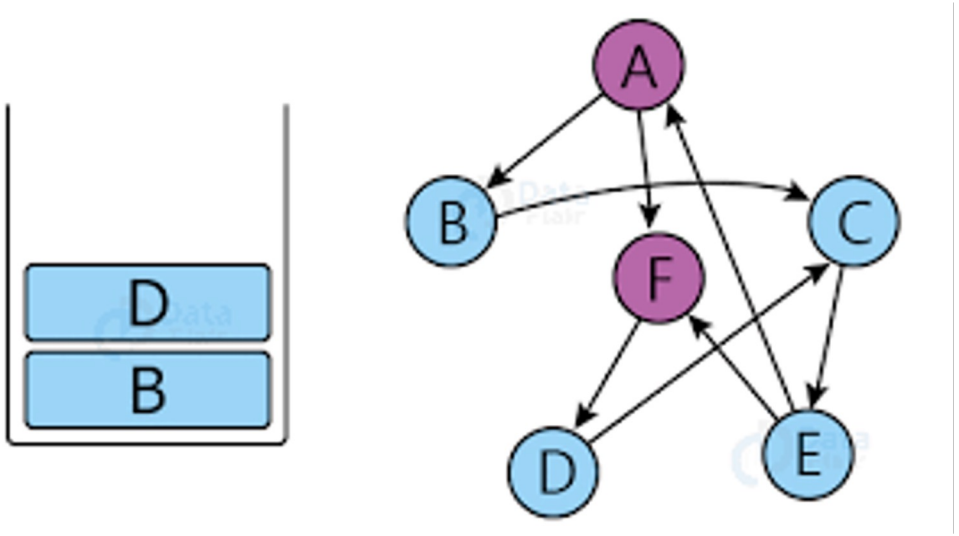


Image credit - https://data-flair.training/

Traversing Binary Trees

Traversal is a process to visit all the nodes of a tree and may print their values too. Because, all nodes are connected via edges (links) we always start from the root (head) node. That is, we cannot randomly access a node in a tree. There are three ways which we use to traverse a tree −

- In-order Traversal
- Pre-order Traversal
- Post-order Traversal

Generally, we traverse a tree to search or locate a given item or key in the tree or to print all the values it contains.

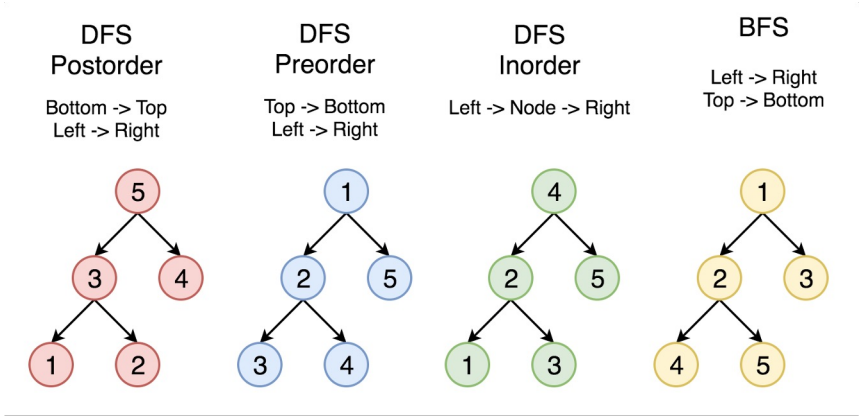


Image credit - https://zhang-xiao-mu.blog/

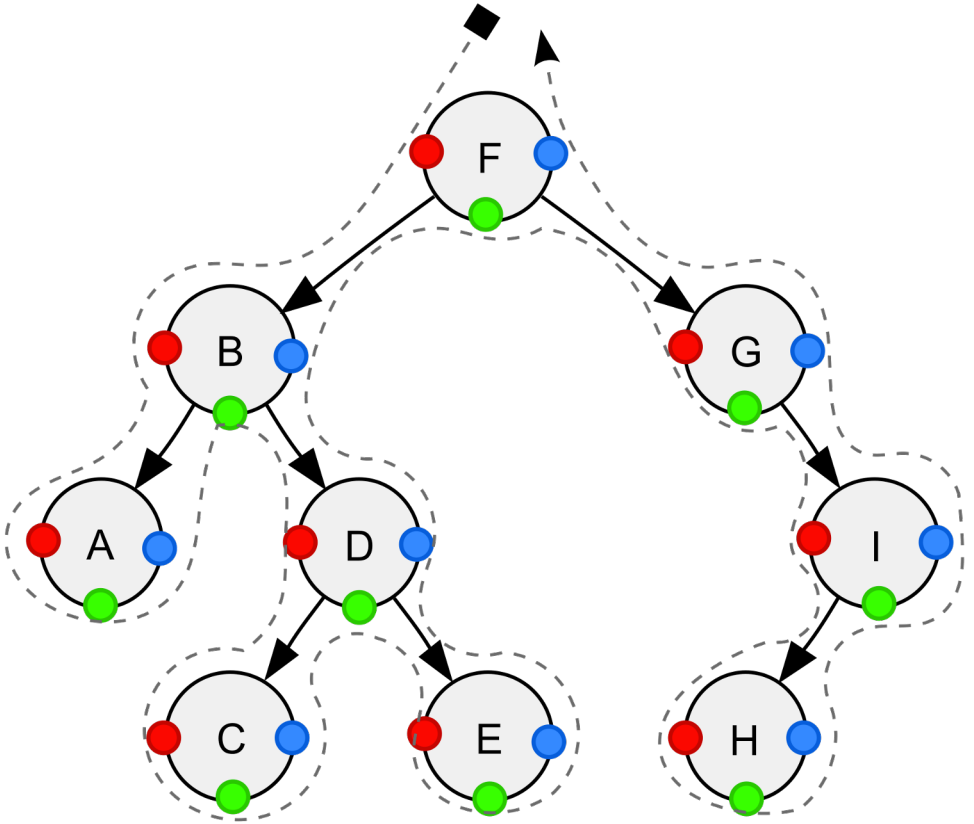


Image credit - https://en.wikipedia.org/

Pre-order Traversal

In this traversal method, the root node is visited first, then the left subtree and finally the right subtree.

```
Until all nodes are traversed –
Step 1 – Visit root node.
Step 2 – Recursively traverse left subtree.
Step 3 – Recursively traverse right subtree.
```

In-order Traversal

In this traversal method, the left subtree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a subtree itself.

If a binary tree is traversed in-order, the output will produce sorted key values in an ascending order.

```
Until all nodes are traversed –
Step 1 – Recursively traverse left subtree.
Step 2 – Visit root node.
Step 3 – Recursively traverse right subtree.
```

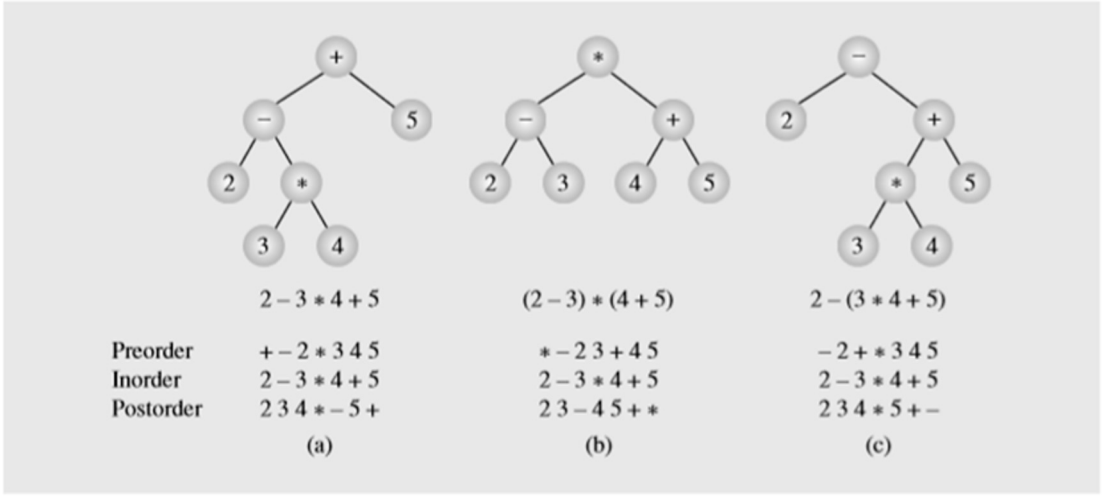


Image credit - https://stackoverflow.com/

Post-order Traversal

In this traversal method, the root node is visited last, hence the name. First we traverse the left subtree, then the right subtree and finally the root node.

```
Until all nodes are traversed –
Step 1 – Recursively traverse left subtree.
Step 2 – Recursively traverse right subtree.
Step 3 – Visit root node.
```