depth first search

1. Start with the first node (called the start node) and place it on top of a stack.

2. Keep a list called 'visited' to remember which nodes have already been explored.

3. While the stack is not empty, take out the top node from the stack.

4. If this node has not been visited, mark it as visited.

5. Check if this node is the goal (the target node we are searching for). If yes, stop the search and confirm it exists.

6. If it is not the goal, add all of its children (connected nodes) to the stack. They are added in reverse order to keep the correct DFS path.

7. Repeat the process until the goal is found or the stack becomes empty.

8. At the end, the visited list shows the order in which the nodes were explored.

**1. Inorder Traversal**

* **Definition:** Visit the **left subtree first**, then the **root node**, and finally the **right subtree**.
* **Order:** Left → Root → Right
* **Usage:** In a **Binary Search Tree (BST)**, inorder traversal gives nodes in **sorted order**.

**2. Preorder Traversal**

* **Definition:** Visit the **root node first**, then the **left subtree**, and finally the **right subtree**.
* **Order:** Root → Left → Right
* **Usage:** Useful for creating a **copy of the tree** or for problems where the root must be processed before its children.

**3. Postorder Traversal**

* **Definition:** Visit the **left subtree first**, then the **right subtree**, and finally the **root node**.
* **Order:** Left → Right → Root
* **Usage:** Useful for **deleting a tree** (because children are handled before the root) or evaluating **expression trees**