TRAVERSING TREES

Climbing trees for fun and profit!

TRAVERSAL

Goal - visit each node once and only once

Three operations

- **visit** current node
- traverse to left node
- traverse to right node

TWO MAIN TYPES

Depth-first

Down first - visit child, then next descendent

Breadth-first

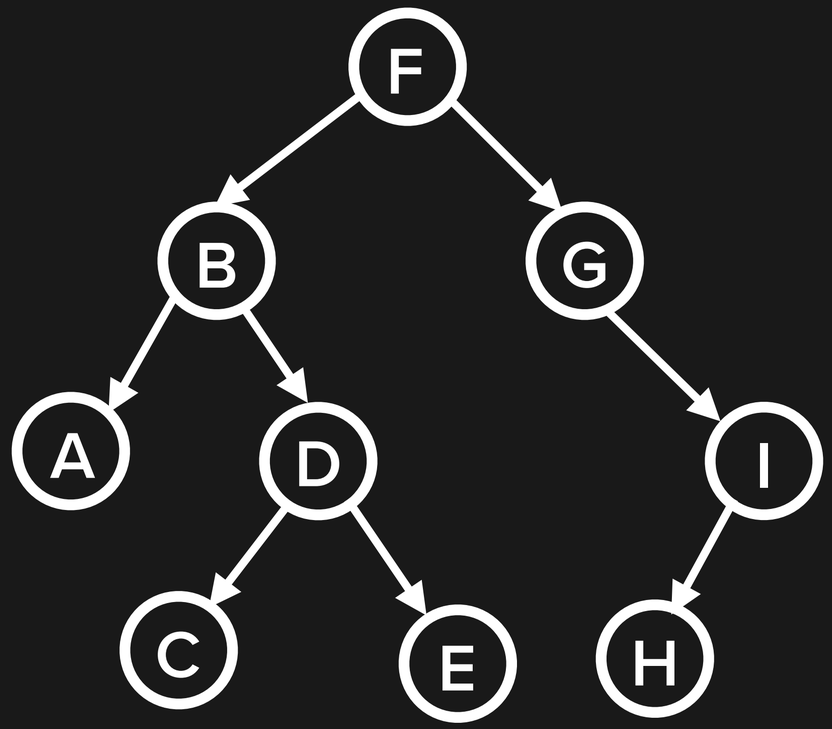
Across first - visit all siblings before going deeper

DEPTH-FIRST SEARCH

Always traverse left subtree before right

Three types of visitation

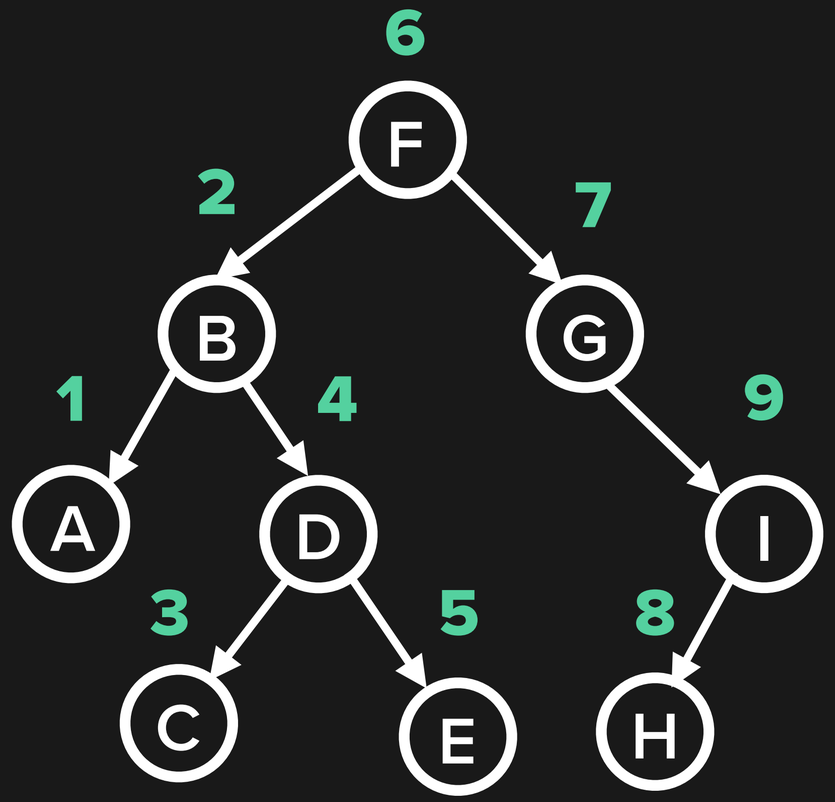
- Pre-order
- In-order
- Post-order



IN-ORDER DFS

```
def in_order_dfs(node):
    if node is not None:
        in_order_dfs(node.left)
        visit(node)
        in_order_dfs(node.right)
```

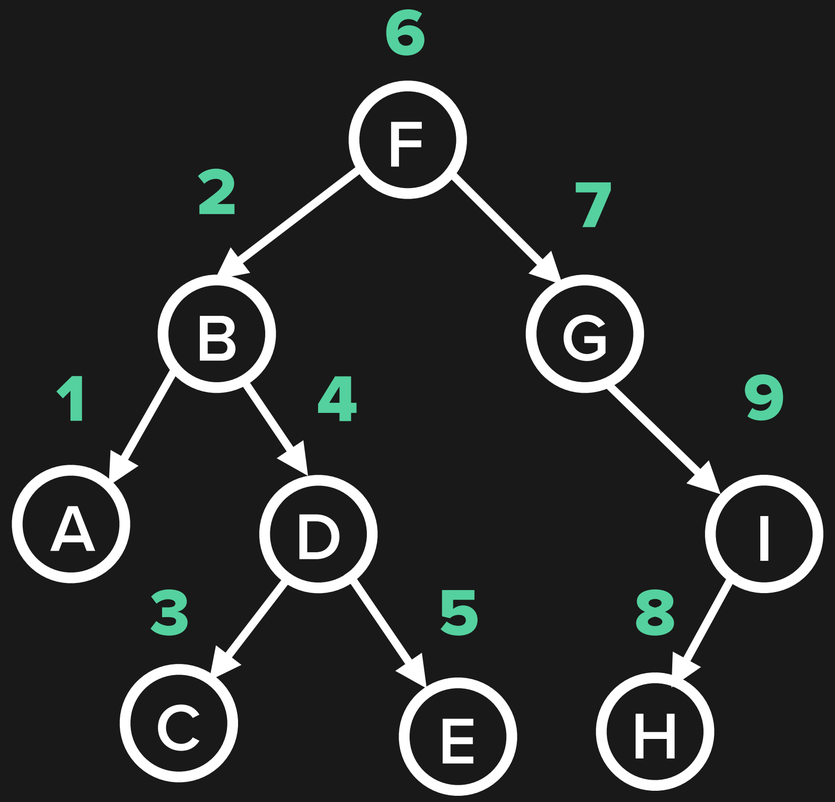
A B C D E F G H I



IN-ORDER DFS

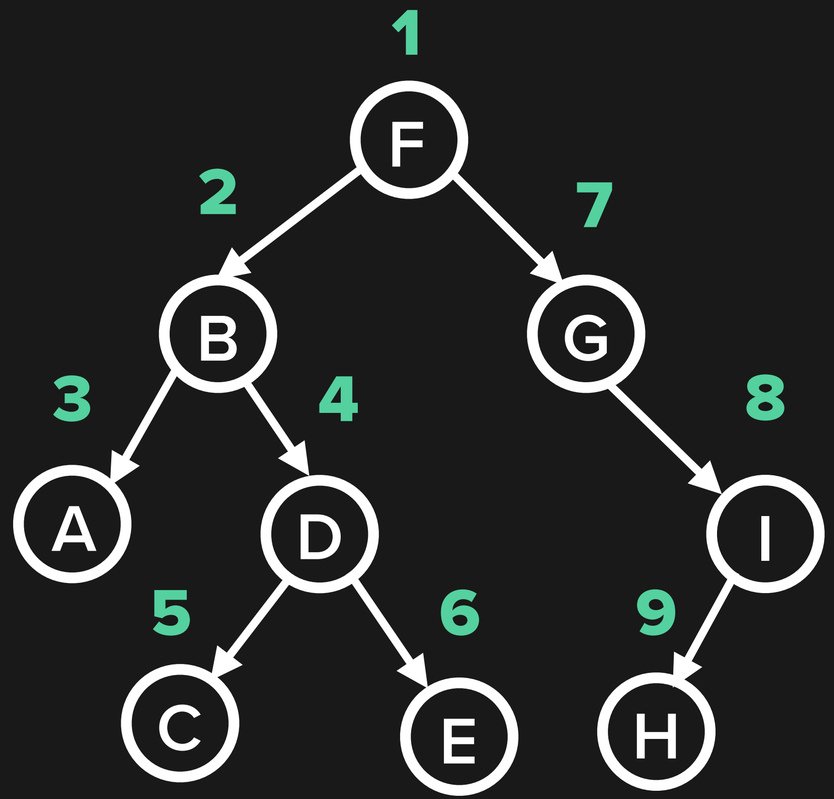
```
def in_order_dfs(node):
    if node is not None:
        if node.left is not None:
            in_order_dfs(node.left)
        visit(node)
        if node.right is not None:
            in_order_dfs(node.right)
```

A B C D E F G H I



PRE-ORDER DFS

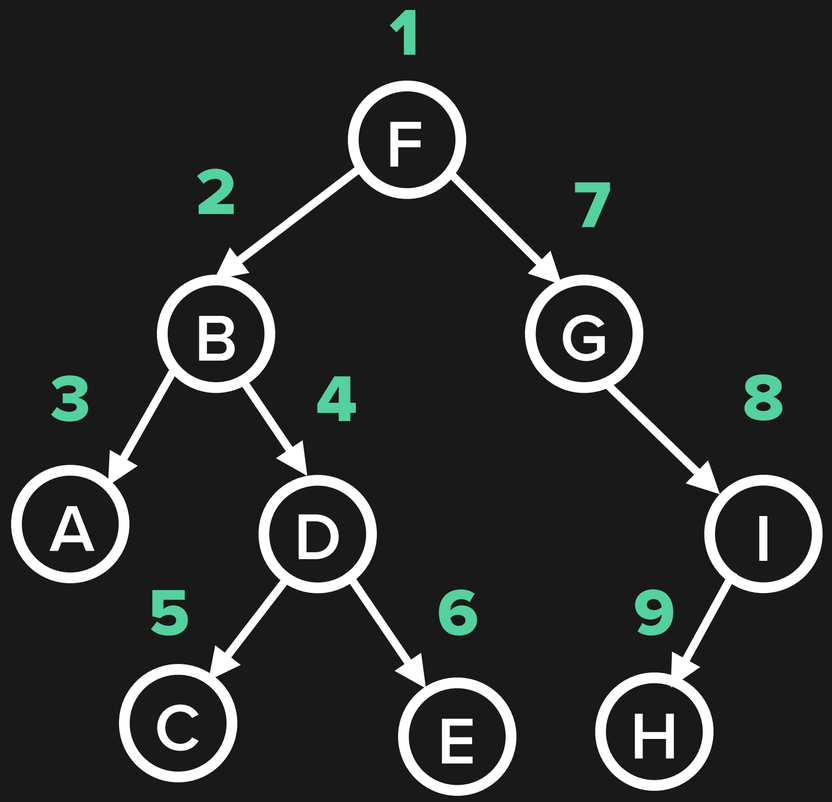
```
def pre_order_dfs(node):
    if node is not None:
        visit(node)
        pre_order_dfs(node.left)
        pre_order_dfs(node.right)
```



F B A D C E G I H

PRE-ORDER DFS

```
def pre_order_dfs(node):
    if node is not None:
        visit(node)
        if node.left is not None:
            pre_order_dfs(node.left)
        if node.right is not None:
            pre_order_dfs(node.right)
```

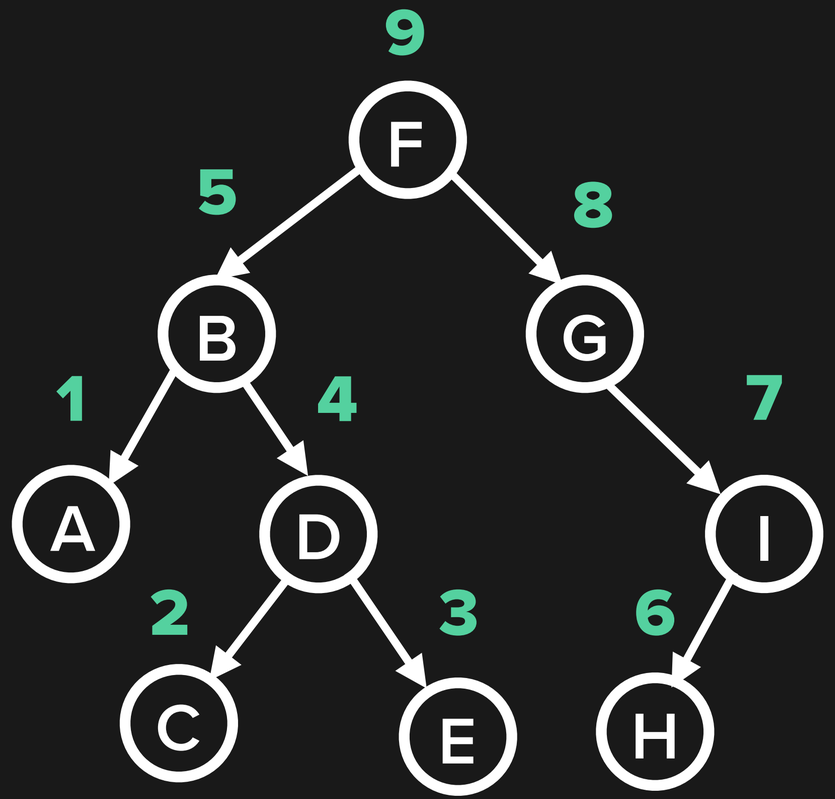


F B A D C E G I H

POST-ORDER DFS

```
def post_order_dfs(node):
    if node is not None:
        post_order_dfs(node.left)
        post_order_dfs(node.right)
        visit(node)
```

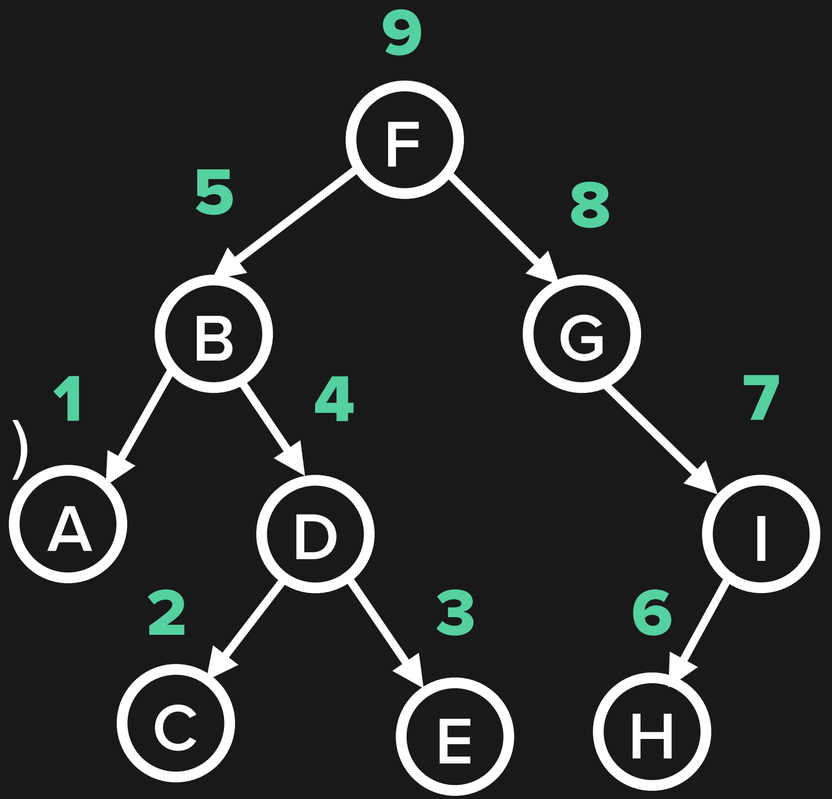
A C E D B H I G F



POST-ORDER DFS

```
def post_order_dfs(node):
    if node is not None:
        if node.left is not None:
            post_order_dfs(node.left)
        if node.right is not None:
            post_order_dfs(node.right)
        visit(node)
```

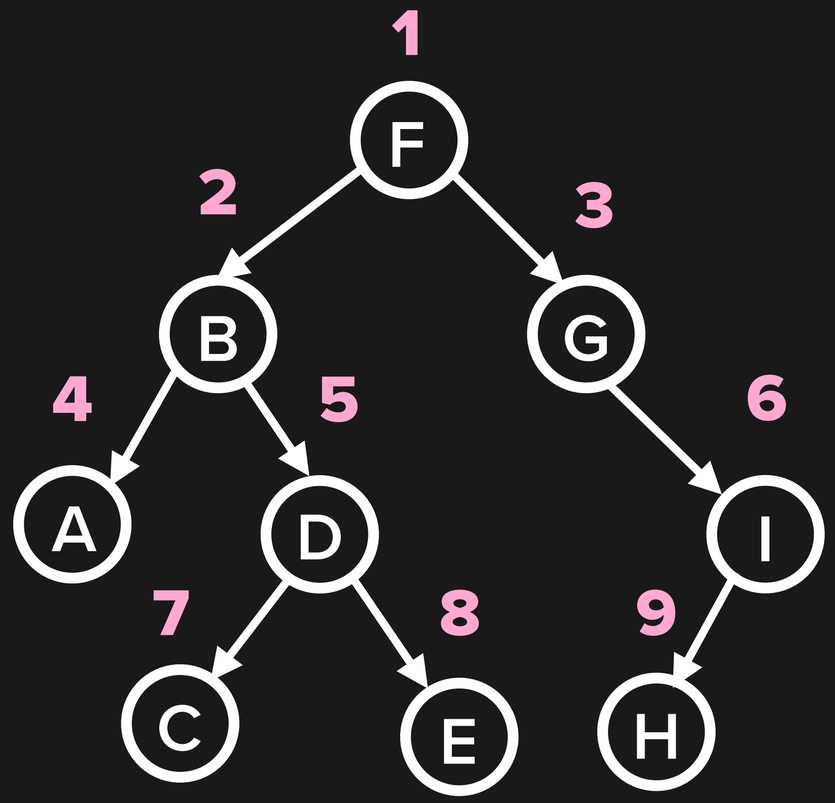
A C E D B H I G F



BREADTH-FIRST SEARCH

```
from queue import Queue

def bfs(root_node):
    queue = Queue()
    queue.enqueue(root_node)
    while len(queue) > 0:
        node = queue.dequeue()
        visit(node)
        if node.left is not None:
            queue.enqueue(node.left)
        if node.right is not None:
            queue.enqueue(node.right)
```



F B G A D I C E H