Trees – Part 2
Traversal

Different approaches to moving through trees

Contents

- Introduction to Tree Traversal
- Depth-First
  - Pre-order
  - Post-order
  - In-order
- Breadth-First
- Summary

Canberra Institute of Technology

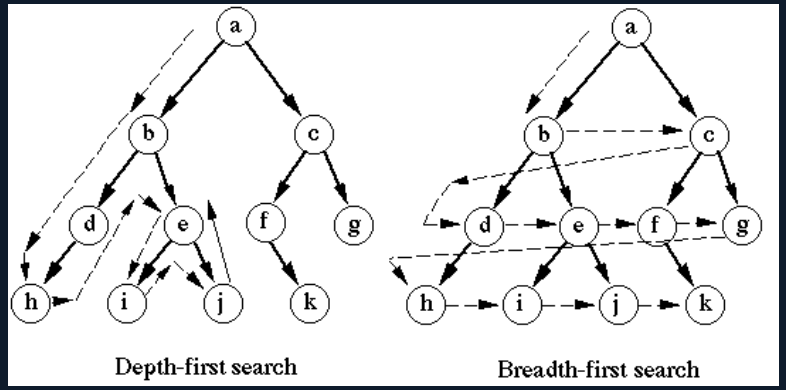
aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Tree Traversal

- Tree traversing is when we search through the nodes in a tree, either to access all the values or to find a specific one.
- Tree traversal is also known as “walking the tree”.
- There are many ways to traverse a tree, the most common are:
  - Depth-first
  - Breadth-first

Tree Traversal

- Depth-first searches down each branch all the way to the bottom before moving across to sibling nodes.
- Breadth-first moves across the siblings first before moving down the branches.



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Depth-First

- There are three different approaches to depth-first traversal.
  - Pre-order: This walk starts with a parent node and then traverses its child nodes
  - Post-order: This walk first traverses down a node's branch and processes the leaf nodes first
  - In-order: This walk visits the left child first, operates on the parent node, and then visits the right child node
- We use recursion to do this!

Depth-First: Pre-order

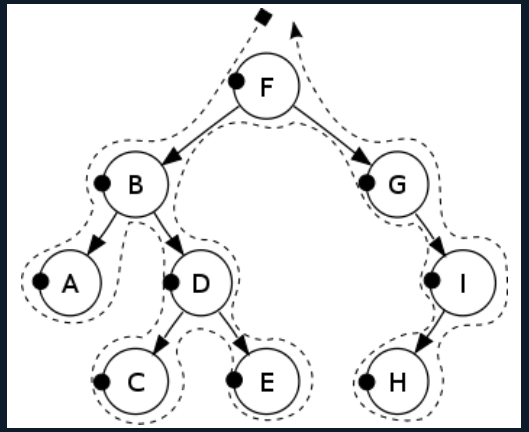
- For pre-order we:
  - Process node
  - Loop through children of node
    - Call pre-order function on each child

```
FUNCTION Preorder

    Print currentNode;

    Child1.Preorder;
    Child2.Preorder;

END FUNCTION
```



Result: F, B, A, D, C, E, G, I, H

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Depth-First: Post-order

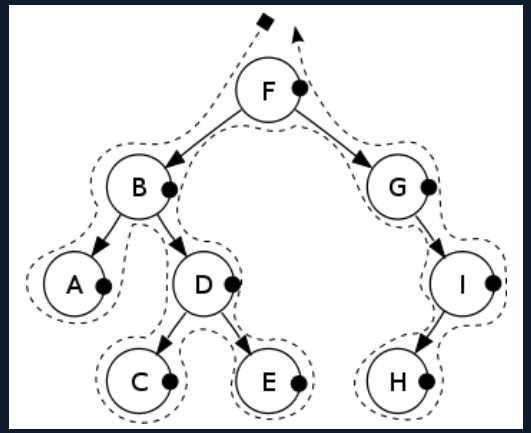
- For post-order we:
  - Loop through children of node
    - Call post-order function on each child
  - Process node

```
FUNCTION Postorder

    Child1.Postorder;
    Child2.Postorder;

    Print currentNode;

END FUNCTION
```



Result: A, C, E, D, B, H, I, G, F

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Depth-First: In-order

- For in-order we:
  - Call in-order function on first child
  - Process node
  - Call in-order function on second child

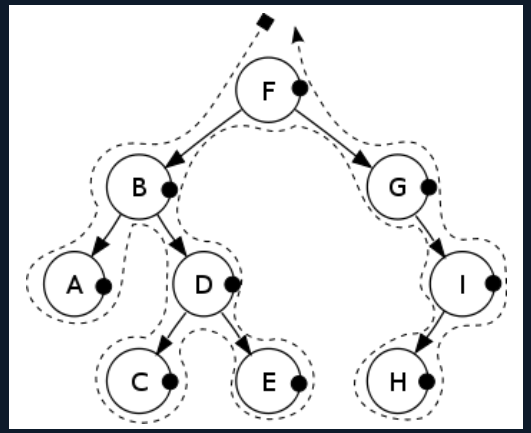
```
FUNCTION Inorder

    Child1.Inorder;

    Print currentNode;

    Child2.Inorder;

END FUNCTION
```



Result: A, B, C, D, E, F, G, H, I

Canberra Institute of Technology

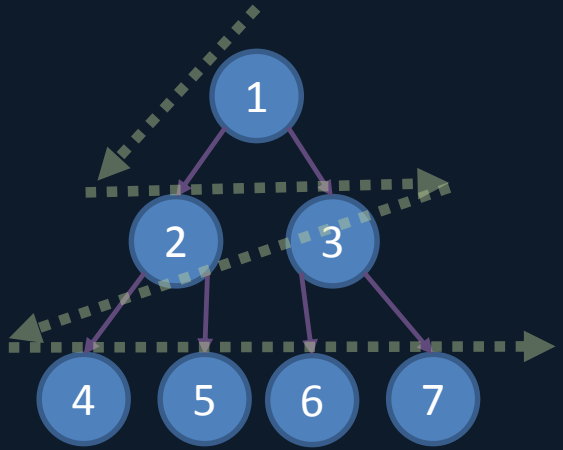
aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Depth-First: Comparison

- When should we use each method?
  - Pre-order: More efficient if we know the node we are searching for is likely to be towards the top of the tree.
  - Post-order: More efficient if we know the node is likely to be in one of the leaves.
  - In-order: Good for flattening the tree into an ordered array or list.

Breadth-First

- Unlike depth-first, there is only one approach to breadth-first searching.
  - Process the parent, then all children before moving on to grandchildren.
  - We use a list or queue to do this.



Breadth-First

- Breadth-first is easy to implement and doesn't require recursion.
- Less efficient when used as a search than depth-first.

```
FUNCTION BreadthFirst

    ADD root to list
    LOOP through nodes in list until empty
        PRINT node
        REMOVE node from list
        ADD node's children to back of list
    END LOOP

END FUNCTION
```

Summary

- Tree Traversal is good for getting all values in a tree or for searching for a specific value.

- There are lots of different ways to walk a tree.
  - Depth-first takes less memory and is often faster to search.
  - Breadth-first is better if looking for a node near the root or if the tree is a lot deeper than it is wide.