Data Structures and Algorithms

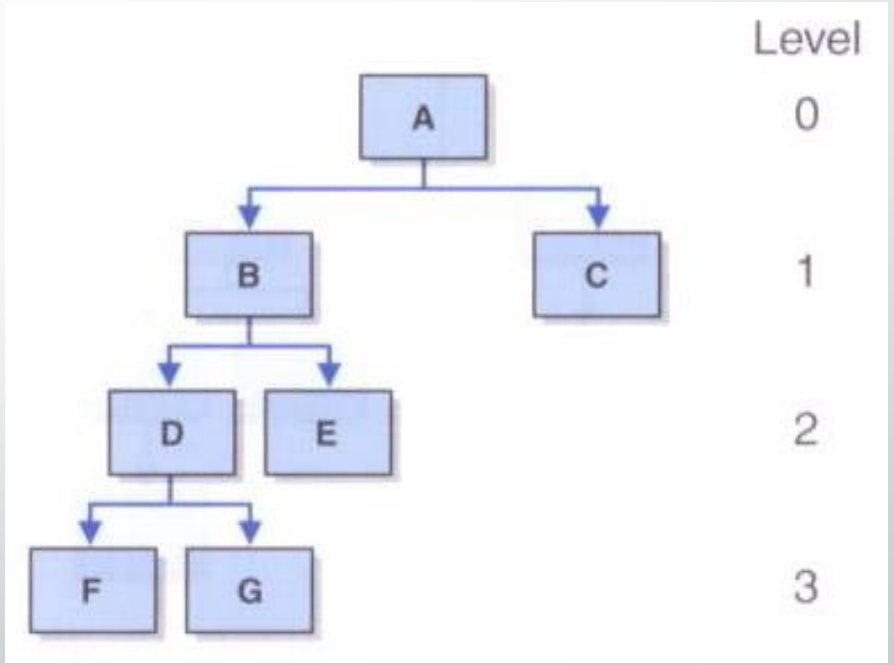
More Tree Traversals

Amilcar Aponte

amilcar@cct.ie

Binary Trees

- Trees in which nodes may have at most two children are called ***binary trees***



Traversals

- Depth-First Traversals
- Breadth-First Traversals

Depth-First Traversals

- In this case we are visiting the nodes of the tree as far down the edges as possible.
- We'll be visiting full branches before we visit sibling node. We've done this already
  - InOrder
  - PreOrder
  - PostOrder

- https://www.khanacademy.org/computer-programming/depth-first-traversals-of-binary-trees/934024358

Breadth First Traversal

- In this case, we're going to visit the width of the tree, before going down any branch
- Coding this is a bit more elaborated because it can't be done recursively.
- Instead, we need to use a queue to store the order in which nodes will be visited.

Breadth-First Tree Traversal

- The process is not recursive, since we are not traversing entire subtrees at once.
- We use a queue to produce a FIFO (i.e., first-in first-out) semantics for the order in which we visit nodes.

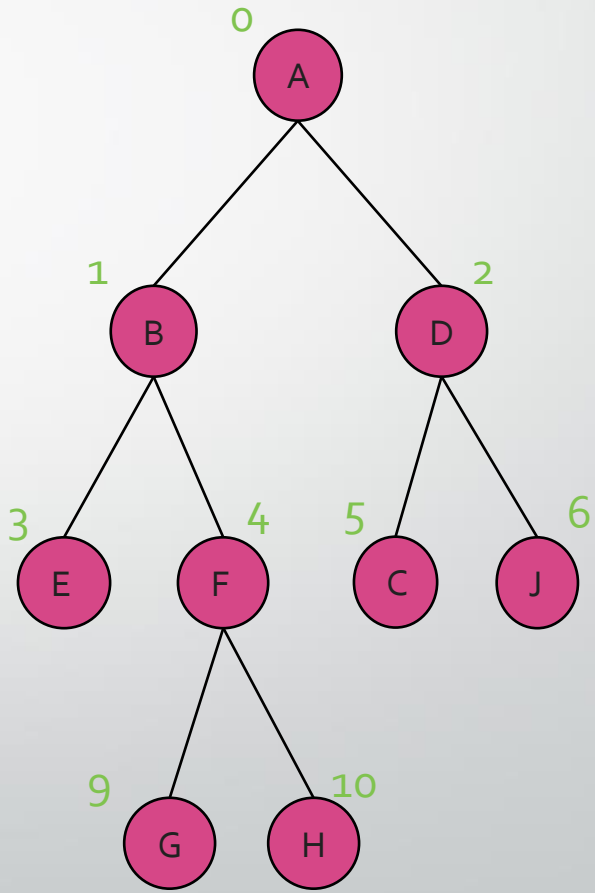
```
Algorithm breadthfirst( ):
    Initialize queue Q to contain root( )
    while Q not empty do
        p = Q.dequeue( )                    { p is the oldest entry in the queue }
        perform the "visit" action for position p
        for each child c in children(p) do
            Q.enqueue(c)    { add p's children to the end of the queue for later visits }
```

Give it a go

- Using the pseudo code from the previous slide,

Breadth-First Traversal

```
public void breadthFirst() {
    IntBSTNode p = root;
    Queue queue = new Queue;
    if (p != null) {
        queue.enqueue(p);
        while (!queue.isempty()) {
            p = (IntBSTNode) queue.dequeue();
            visit(p);
            if (p.left != null)
                queue.enqueue(p.left);
            if (p.right != null)
                queue.enqueue(p.right);
        }
    }
}
```



That’s all folks

- Any questions?