Basic Data Structures II

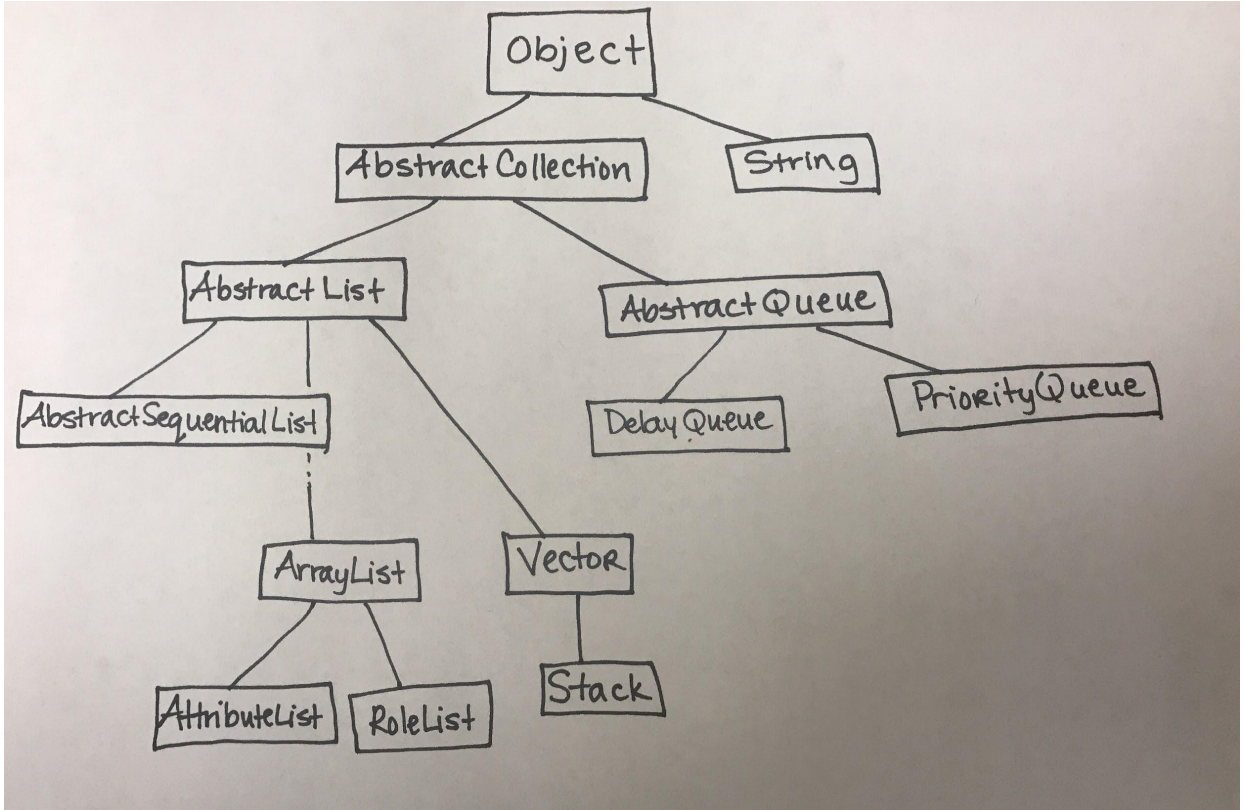
- Rooted Trees
- Structural Induction

What is a (rooted) tree?

What is a (rooted) tree?

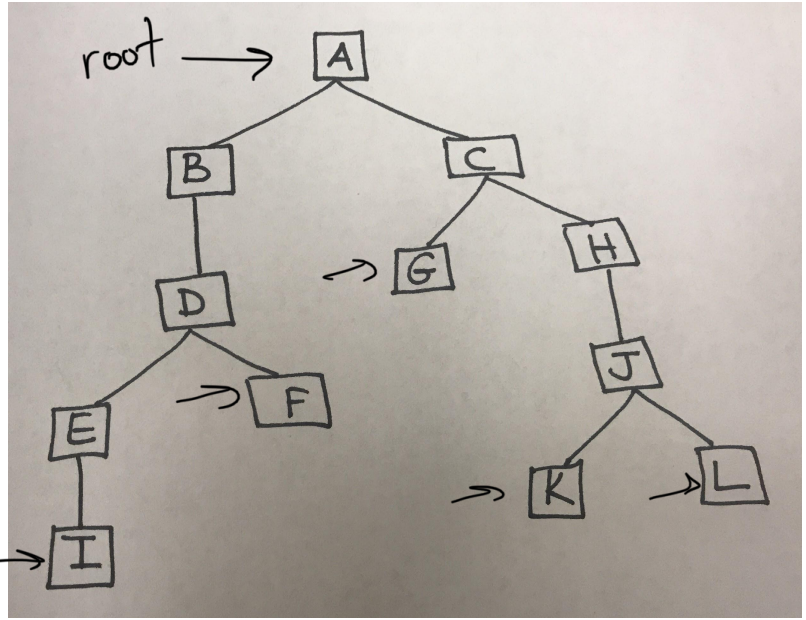
- An abstract data structure (ADT) that models a hierarchical structure
- Terminology: node, root, parent, child, sibling, internal/external node, leaf, ancestors, descendants, depth, height, subtree, etc.

Example: Java classes and inheritance



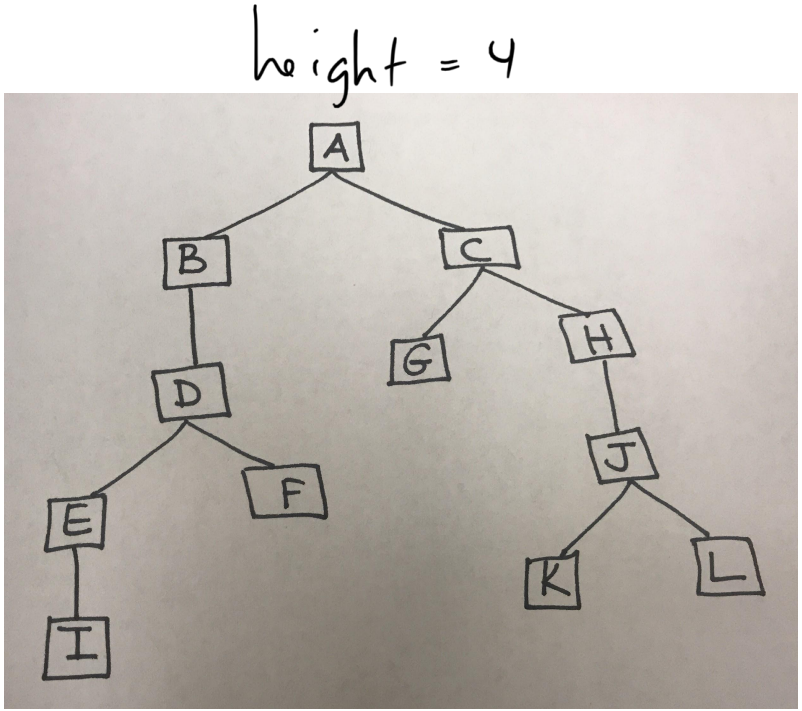
Some terminology

- *size*: # of nodes
- *root*: node at the top
- *parent*: the node directly above another node
- *child*: a node directly below another node
- *internal node*: a node w/ at least one child
- *external node/leaf*: a node w/ no children



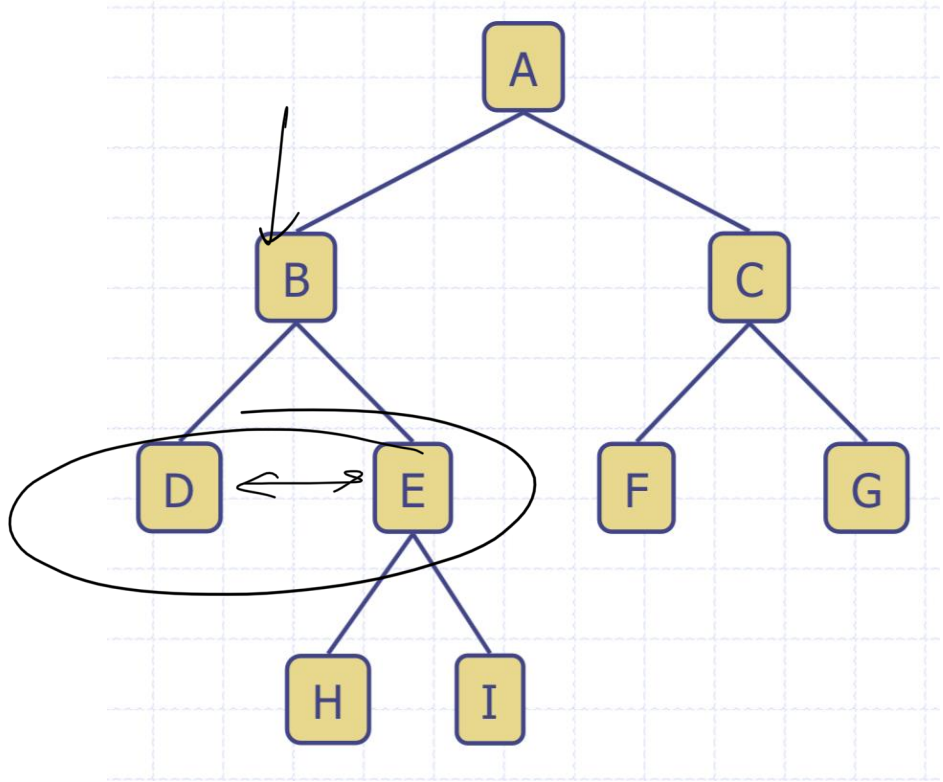
Some terminology

- *height of a node:* the longest distance from the node to a leaf
- *height of the tree*: height of Root
- *depth of a node*: the distance from the node to the root (level)



Binary Tree

- Every node has at most two children
- New terminology: ***leftChild***, ***rightChild***, ***sibling***

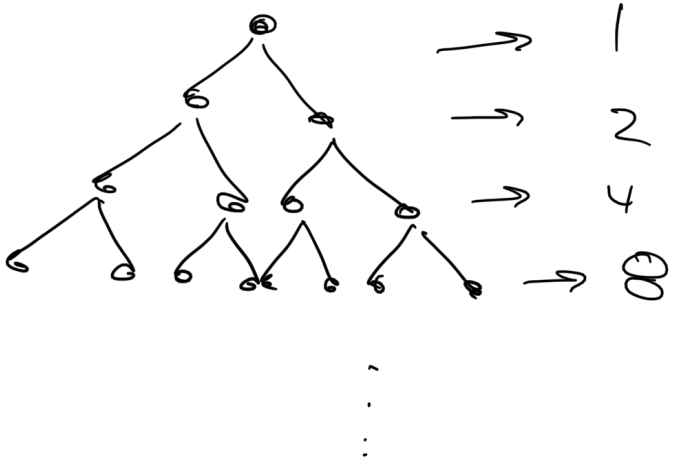


Picture from [2]

Binary Tree

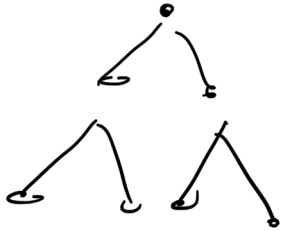
Given a binary tree of height *h*, what is the maximum number of leaves it can have?

2^h



Binary Tree

Given a binary tree of height *h*, what is the maximum number of nodes it can have?



1

2

4

8

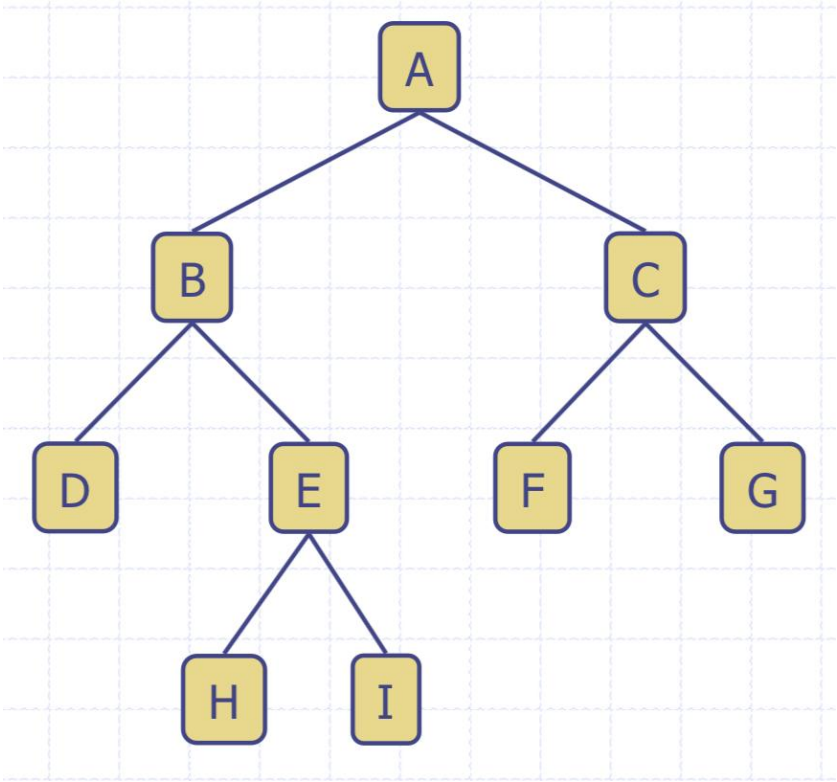
2^h

$$\sum_{k=0}^{h} 2^k = 2^{h+1} - 1$$

Full Binary Tree

- Every internal node has two children.
- Is this a full binary tree?

YES

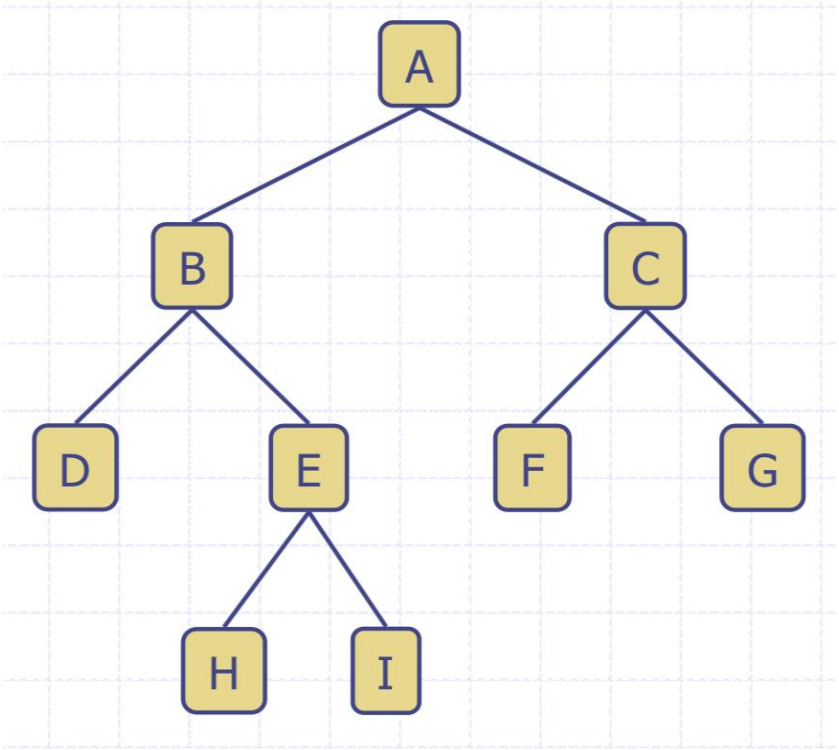


Picture from [2]

Properties of a full binary tree...

Given ***e*** external nodes, ***i*** internal nodes, ***n*** total nodes, and ***h*** the height of the tree:

- $e = i + 1$
- $e <= 2^h$
- $n = 2e - 1$
- $h <= i$
- $h <= (n-1)/2$
- $h >= \log_2 e$
- $h >= \log_2(n + 1) - 1$



Picture from [2]

Structural Induction

Recursive Definitions of Structures

- Base Case: Consider the smallest instance of the structure.
- Inductive Step: Consider how to build a new instance from existing smaller instances.

Definition of Full Binary Tree:

Basis Step. A tree w/ just the root is a FBT.

Inductive Step. Let T_1 and T_2 be FBTs.

Then $T_3 = T_1 \cdot T_2$. (T_1 and T_2 are joined together with a new root) is a FBT.

Definition. Give a recursive definition for the total number of external nodes $e(T)$ of a full binary tree.

Definition. Give a recursive definition for the total number of external nodes $e(T)$ of a full binary tree.

Basis Step. When T is just one node, then $e(T) = 1$.
Inductive Step. Let T_1 and T_2 be full binary trees. You can form a new binary tree T_3 by making a new node and letting T_1 and T_2 be children of that node, which would then be the root of the new tree. (i.e. $T_3 = T_1 \cdot T_2$)

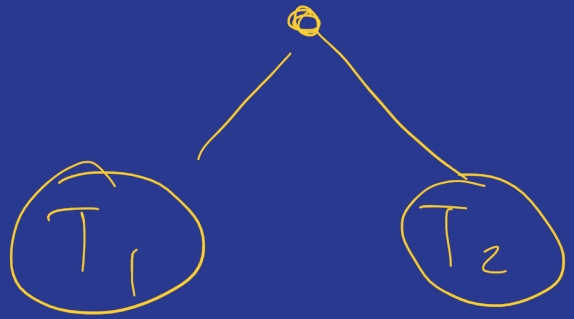
Then $e(T_3) = e(T_1) + e(T_2)$.

Definition. Give a recursive definition for the the number of internal nodes $i(T)$ of a full binary tree.

Definition. Give a recursive definition for the the number of internal nodes *i(T)* of a full binary tree.

Basis Step. When T is just one node, then $i(T) = 0$.
Inductive Step. Let T_1 and T_2 be full binary trees. You can form a new binary tree T_3 by making a new node and letting T_1 and T_2 be children of that node, which would then be the root of the new tree. (i.e. $T_3 = T_1 \cdot T_2$)

Then $i(T_3) = i(T_1) + i(T_2) + 1$.



Definition. Give a recursive definition for the total number of nodes $n(T)$ of a full binary tree.

Definition. Give a recursive definition for the total number of nodes *n(T)* of a full binary tree.

Basis Step. When T is just one node, then $n(T) = 1$.
Inductive Step. Let T_1 and T_2 be full binary trees. You can form a new binary tree T_3 by making a new node and letting T_1 and T_2 be children of that node, which would then be the root of the new tree. (i.e. $T_3 = T_1 \cdot T_2$)

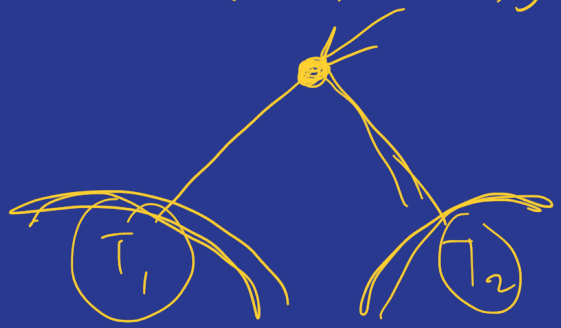
Then $n(T_3) = n(T_1) + n(T_2) + 1$.

Definition. Give a recursive definition for the height $h(T)$ of a full binary tree.

Definition. Give a recursive definition for the height *h(T)* of a full binary tree.

Basis Step. When T is just one node, then $h(T) = 0$.
Inductive Step. Let T_1 and T_2 be full binary trees. You can form a new binary tree T_3 by making a new node and letting T_1 and T_2 be children of that node, which would then be the root of the new tree. (i.e. $T_3 = T_1 \cdot T_2$)

Then $h(T_3) = \max\{h(T_1), h(T_2)\} + 1$.



Proof. Prove by structural induction that a full binary tree with e external nodes has $2e-1$ total nodes.

$n = 2e - 1$

$n(T) = 2e(T) - 1$

Basis Step. Let T be a FBT with just one node.

$e(T) = 1$ and $n(T) = 1$,

so $n(T) = 2e(T) - 1$

Inductive Step. Let T_1 and T_2 be FBTs and let $T_3 = T_1 \cdot T_2$. Assume as IH that

$n(T_1) = 2e(T_1) - 1$ and $n(T_2) = 2e(T_2) - 1$.

$n(T_3) = n(T_1) + n(T_2) + 1$

$= 2e(T_1) - 1 + 2e(T_2) - 1 + 1$ by the IH

$= 2(e(T_1) + e(T_2)) - 1 = 2e(T_3) - 1$

$2^{x+1} + 2^{y+1} \leq 2^{\max(x,y)+1}$

Proof. Prove by structural induction that if *T* is a full binary tree, then $n(T) <= 2^{h(T)+1} - 1$.

Basis Step. Let T be a FBT w/ just one node.

$n(T) = 1 \leq 2^{h(T)+1} - 1 = 2^{0+1} - 1 = 1$ ✓

Inductive Step. Let T_1 and T_2 be FBTs and let $T_3 = T_1 \cdot T_2$. Assume as the IH that $n(T_1) \leq 2^{h(T_1)+1} - 1$ and $n(T_2) \leq 2^{h(T_2)+1} - 1$.

$n(T_3) = n(T_1) + n(T_2) + 1$

$\leq 2^{h(T_1)+1} - 1 + 2^{h(T_2)+1} - 1 + 1$ by the IH

$\leq 2 \cdot 2^{\max(h(T_1), h(T_2)) + 1} - 2 + 1$

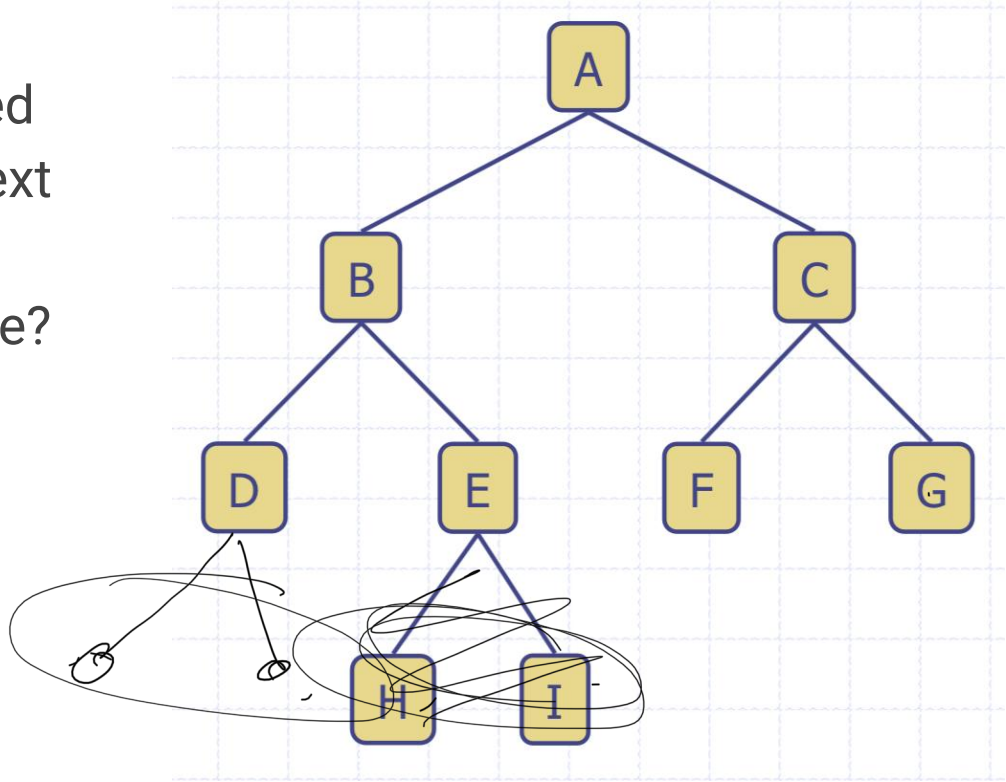
$= 2 \cdot 2^{h(T_3)} - 1$

$= 2^{h(T_3)+1} - 1$

$n(T_3) \leq 2^{h(T_3)+1} - 1$

Complete Binary Tree

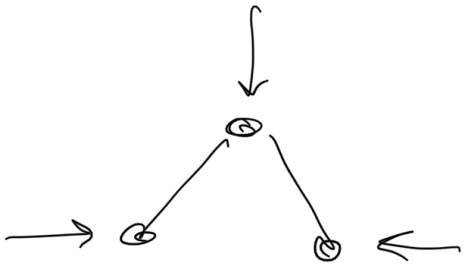
- Each level is completely filled up before going on to the next level.
- Is this a complete binary tree?



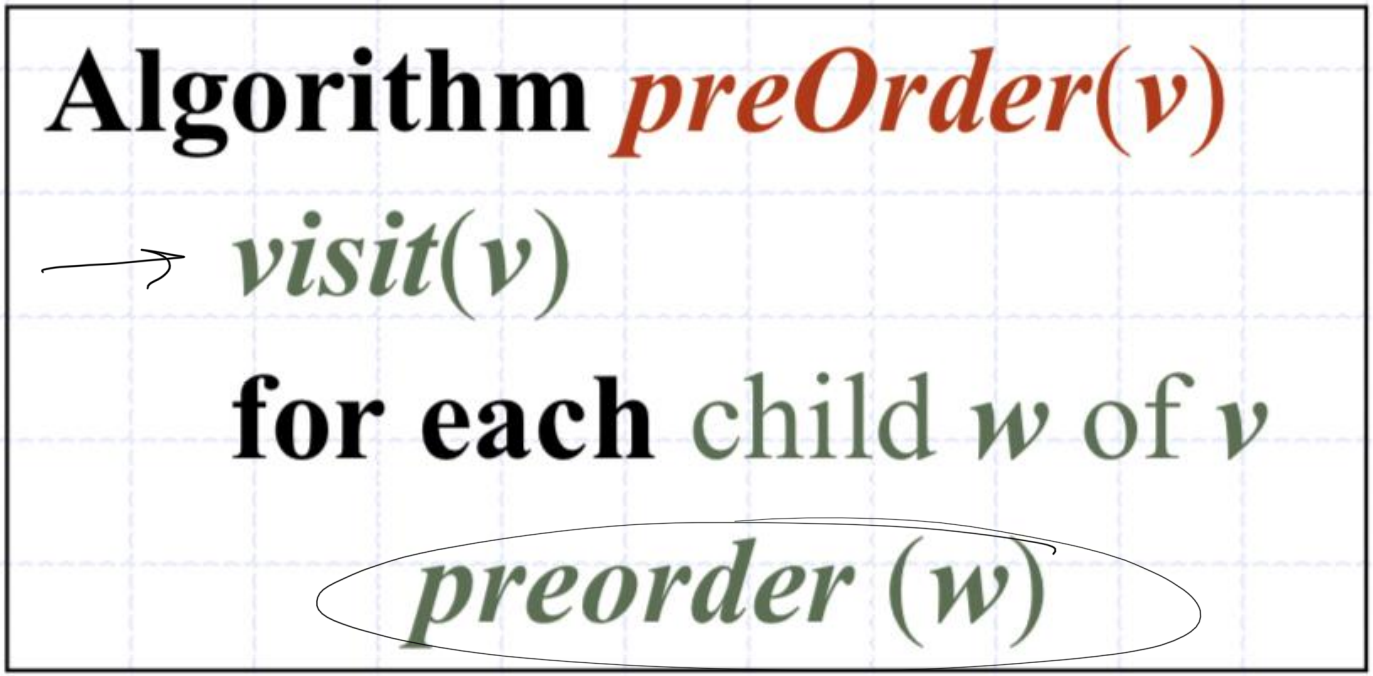
Picture from [2]

Basic Tree Traversals

- Preorder/DFS: parents before children—a node is visited before its descendants
- Postorder: children before parents—a node is visited after its descendants
- Inorder: A node is visited after its left subtree and before its right subtree.
- BFS: Level by level.

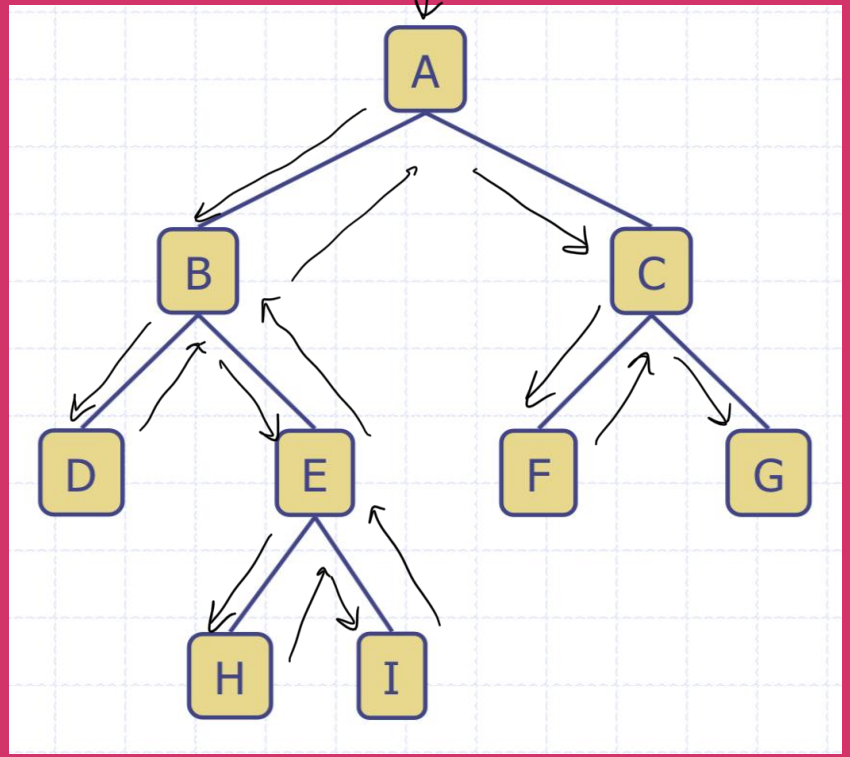


Preorder Traversal



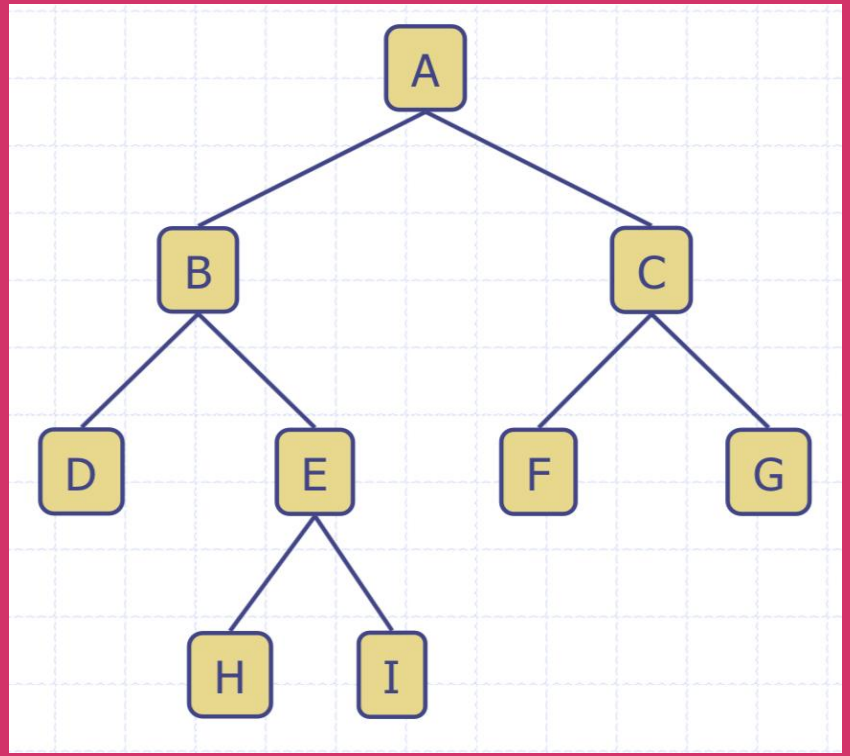
Picture from [2]

Example: What's the pre-order traversal of this tree?



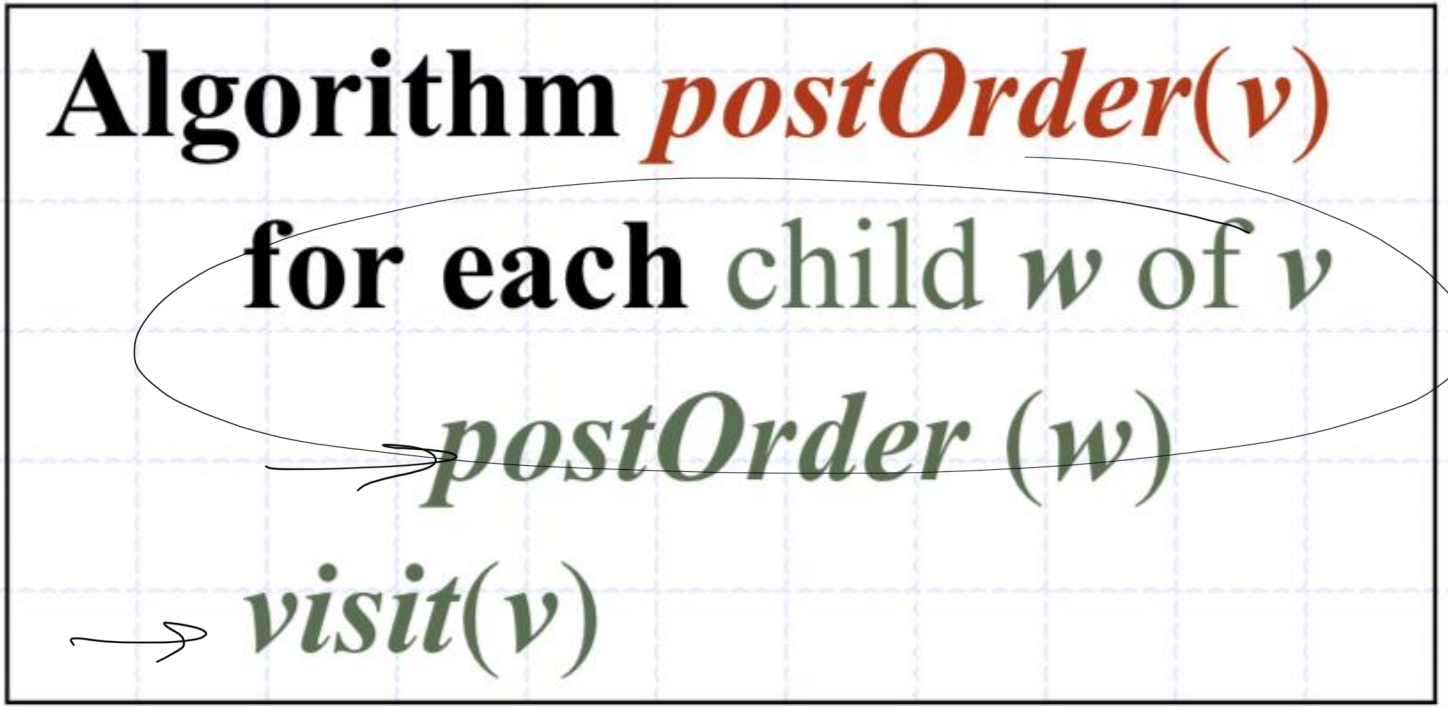
A - B - D - E - H - I - C - F - G

Example: What's the pre-order traversal of this tree?



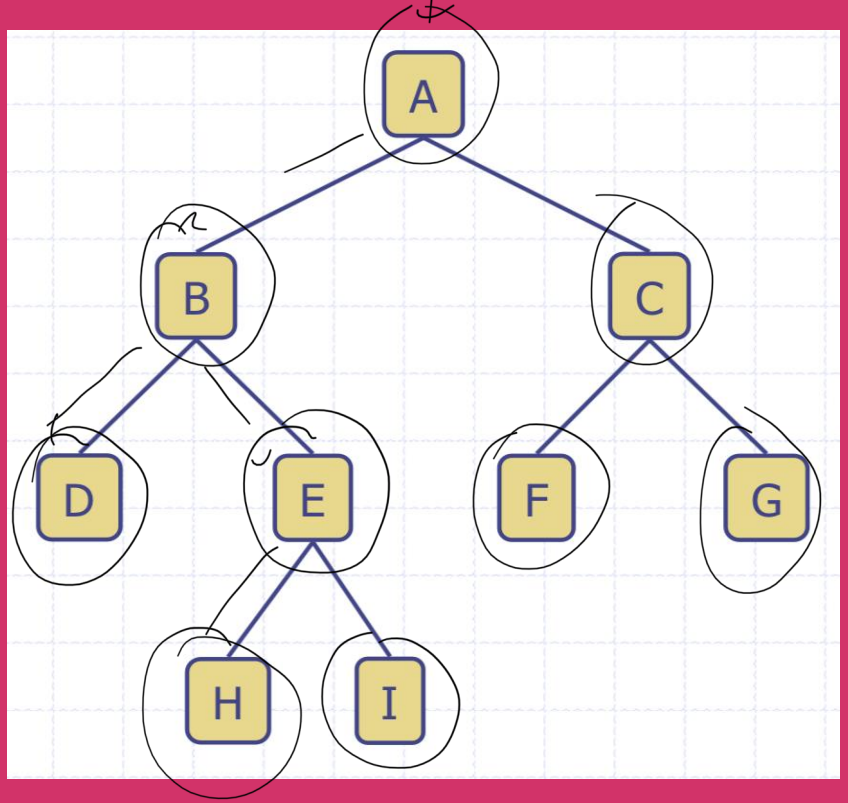
A-B-D-E-H-I-C-F-G

Postorder Traversal

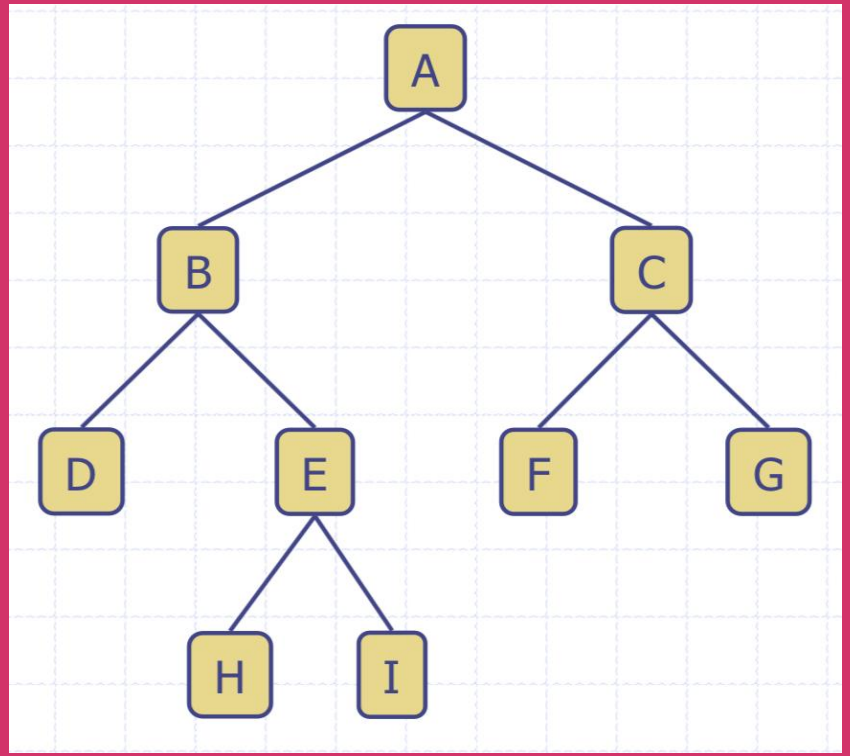


Picture from [2]

Example: What's the post-order traversal of this tree?

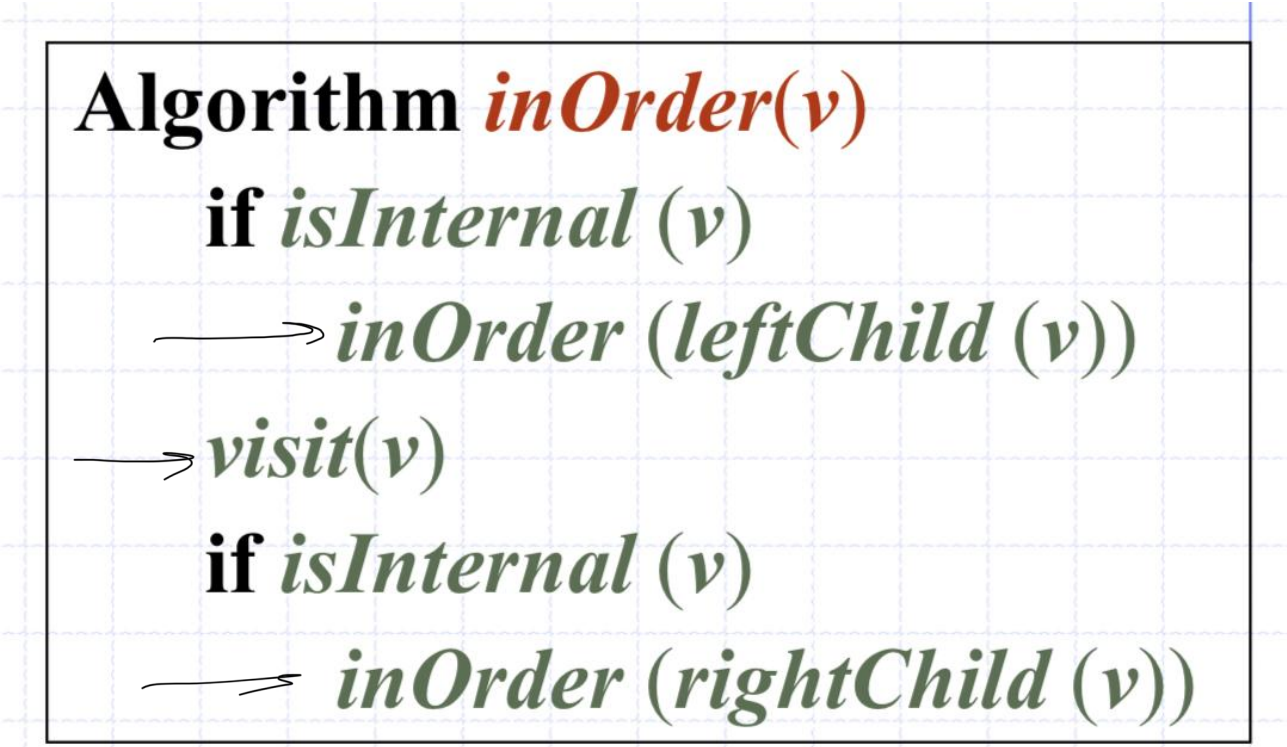


Example: What's the post-order traversal of this tree?



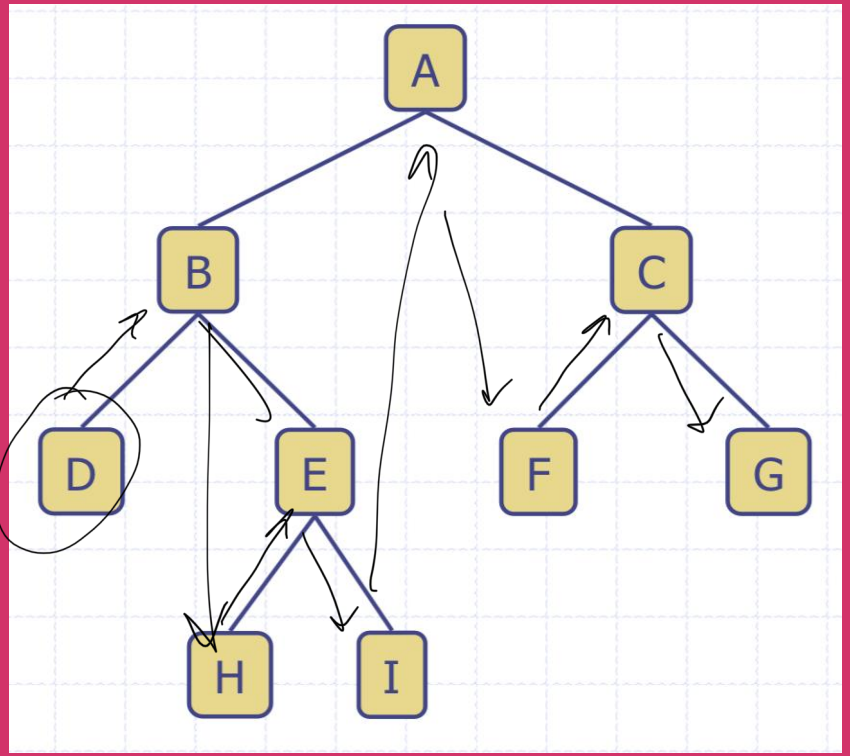
D-H-I-E-B-F-G-C-A

Inorder Traversal

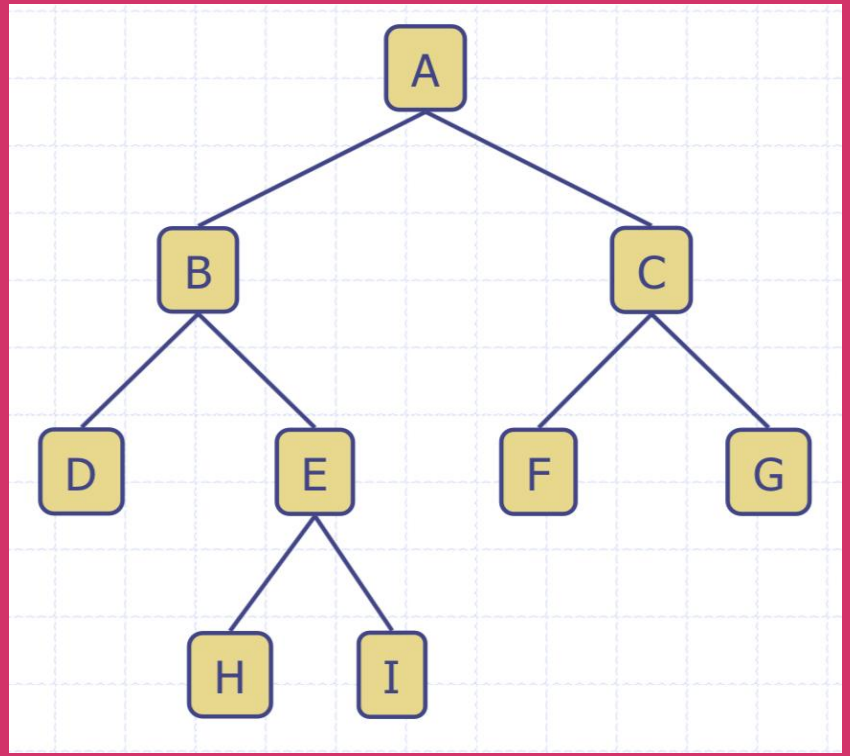


Picture from [2]

Example: What's the in-order traversal of this tree?

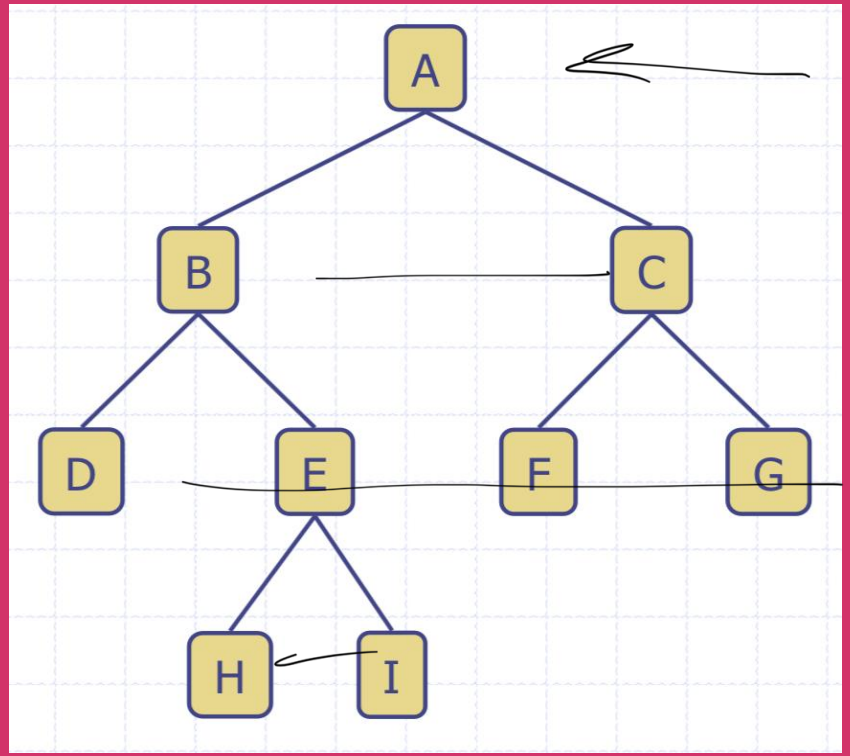


Example: What's the in-order traversal of this tree?



D-B-H-E-I-A-F-C-G

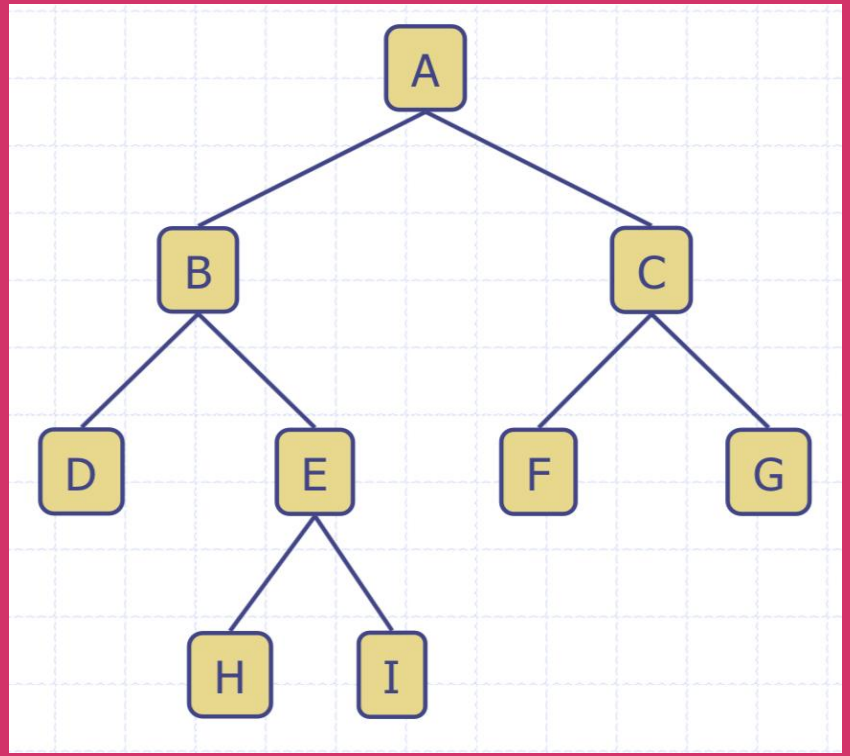
Example: What's the BFS traversal of this tree?



A B C D E F G H I

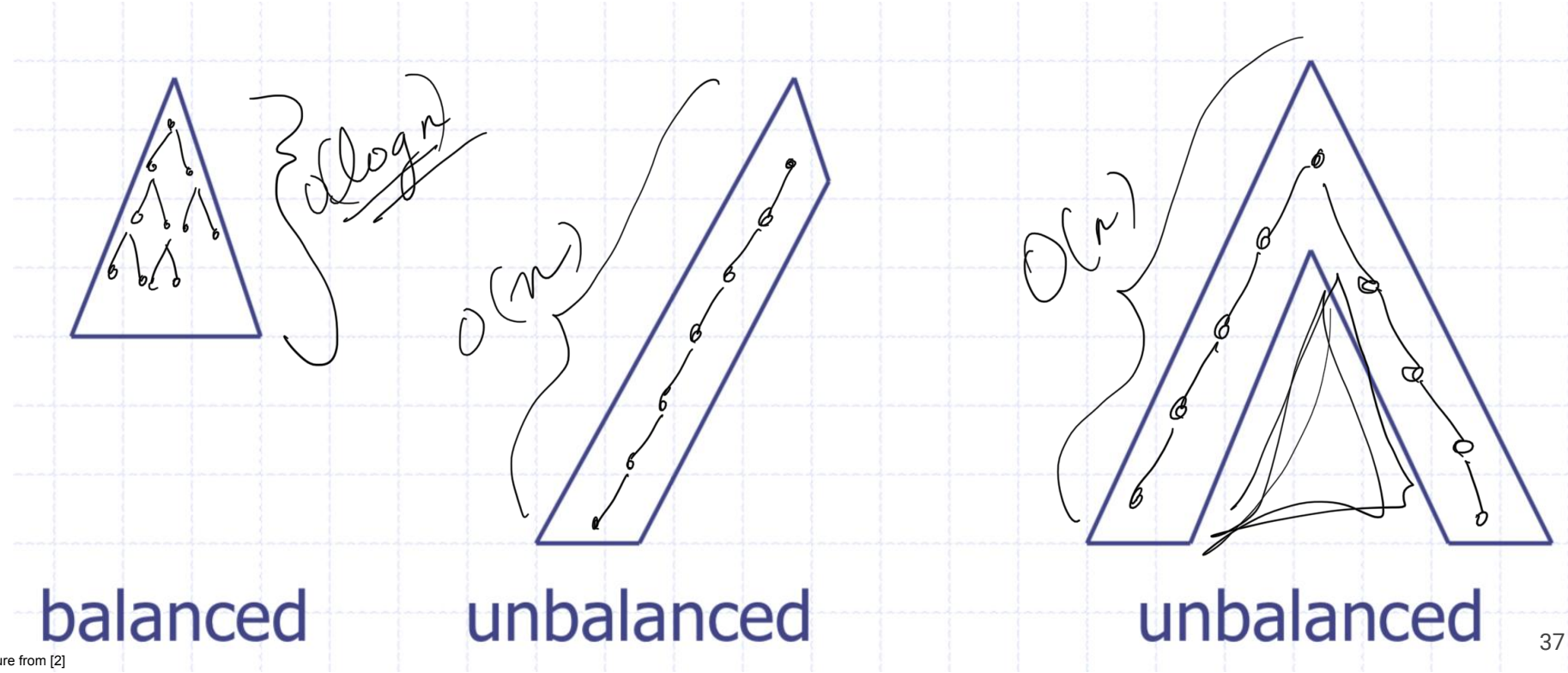
Q: A B C D E

Example: What's the BFS traversal of this tree?



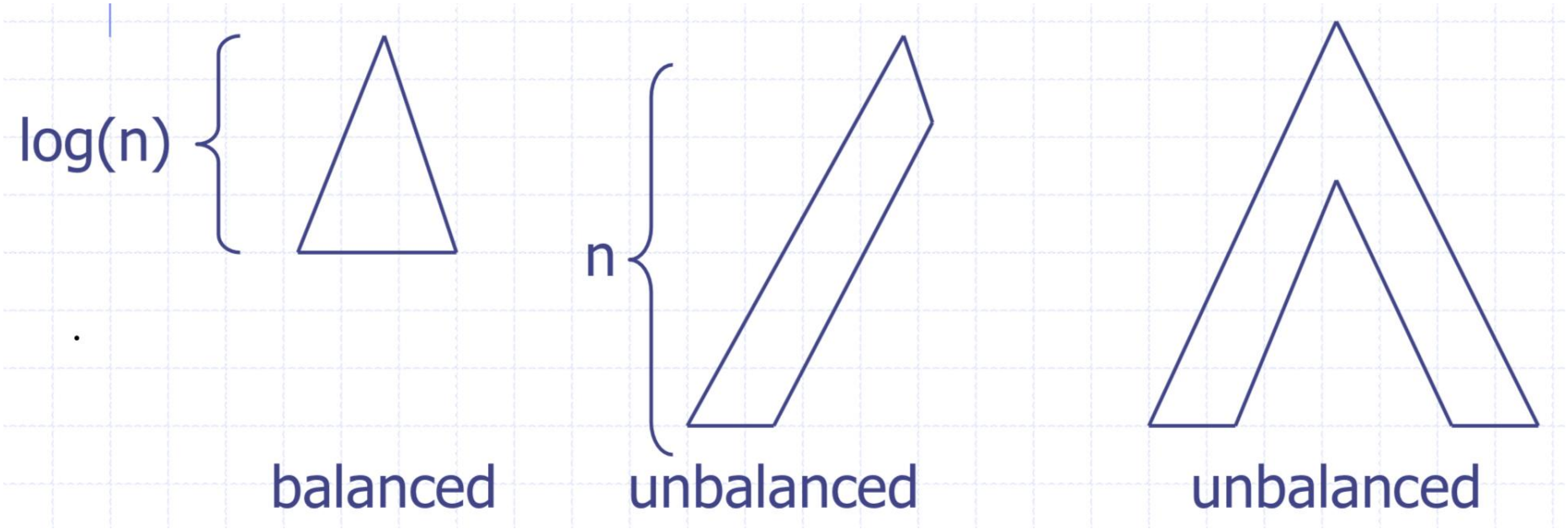
A-B-C-D-E-F-G-H-I

The Importance of Balance: How tall are these trees given ***n*** nodes?



Picture from [2]

The Importance of Balance: How tall are these trees given ***n*** nodes?

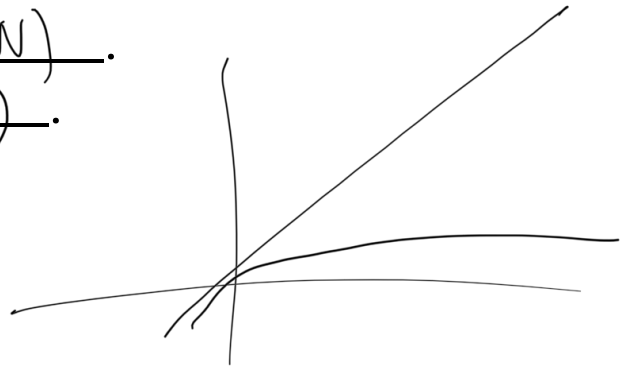


Why does it matter?

Picture from [2]

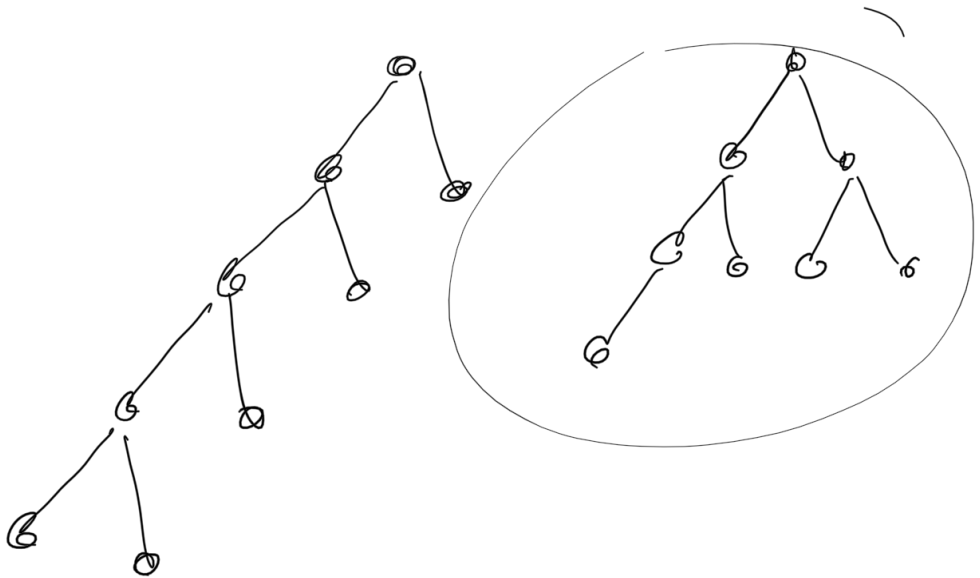
The Importance of Balance: Why does it matter?

- The operations associated with tree data structures often depend on the height of tree (the longest path from root to node).
- If that is the case, then the runtime of such an operation will be proportional to path length, and the maximum possible time for that would be proportional to height of the tree.
- If the tree is unbalanced, that may be O(N).
- If the tree is balanced, that is only O(logN).



Which tree specification that we've discussed guarantees balance?

A. binary
B. full
C. complete



References

[1] *Algorithms, Fourth Edition;* Robert Sedgewick and Kevin Wayne (and associated slides)
[2] *Data Structures & Algorithms in Java,* Goodrich & Tamassia (and associated slides)