# Finally: Trees!

A tawny frogmouth

There's one at the Baltimore aquarium!

Many slides are lifted from Gibson

# Trees!  Now we're getting fancy!


In data structures, trees are not the leafy plant trees, but a logical branching structure.

Some "trees" you've already probably experienced:

- Organization charts
- Family trees
- Decision trees

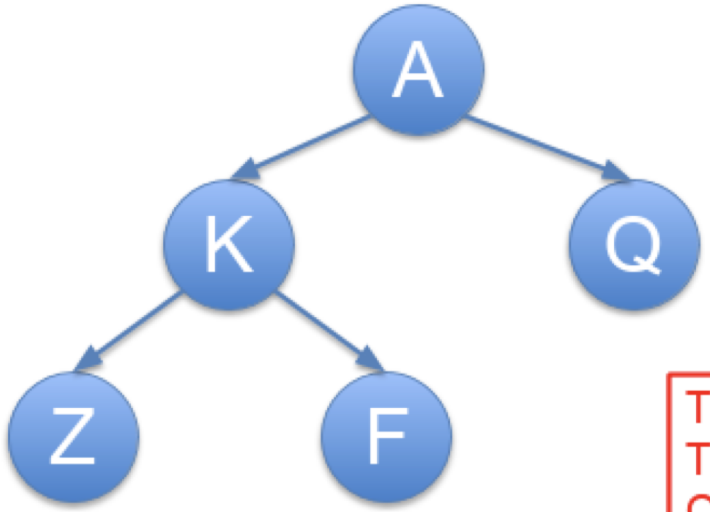
# Nodes revisited


Remember nodes?  No?  A node is a data structure that contains data AND some organizational information (usually in the form of array indices or pointers)

In a linked list, the organizational data is the next (and previous) pointer(s).

In a tree, it is a list of *children* (and possibly a pointer to the parent).

# The family metaphor




The children of A are K and Q.
The children of K are Z and F.
Q, Z, and F have no children.

Parents? Siblings? Grandparents? Cousins? Ancestors? Descendents?

# Angel’s Oak of South Carolina

# More terminology


Each node has *exactly one parent*, except for one node, which is called the ***root***.

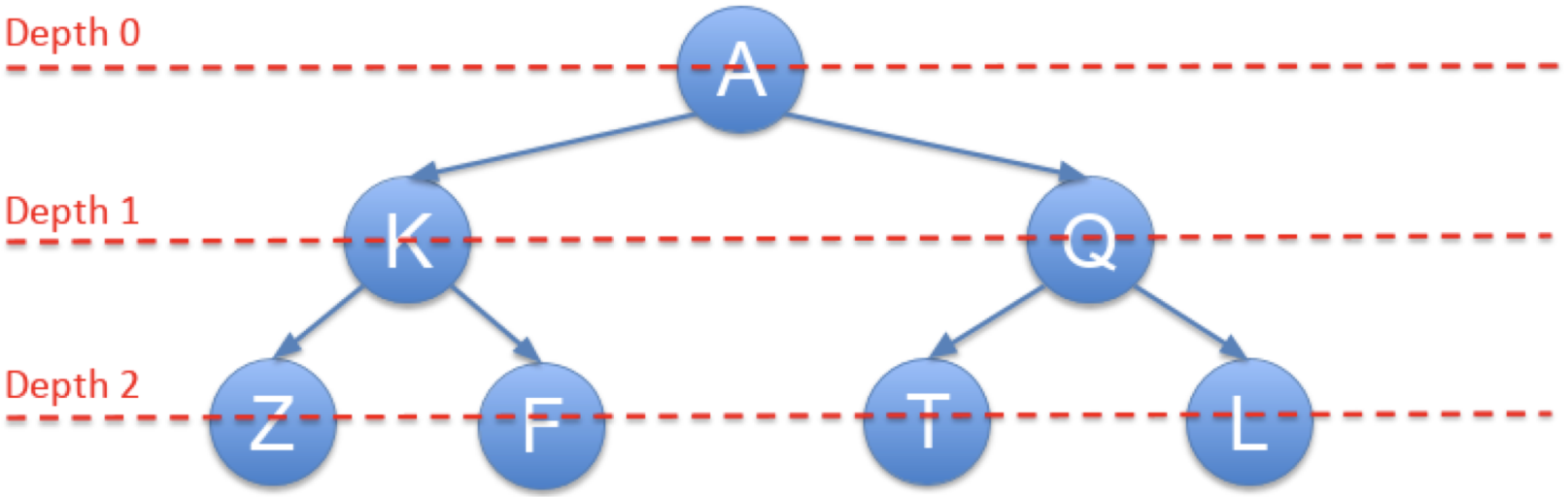
All nodes have zero or more children. Those with zero children are called *leaves.* They are also called *external nodes*.

Nodes that do have children are called *internal nodes.*

# Into the depths



- *Depth* of a node: The number of ancestors excluding itself.

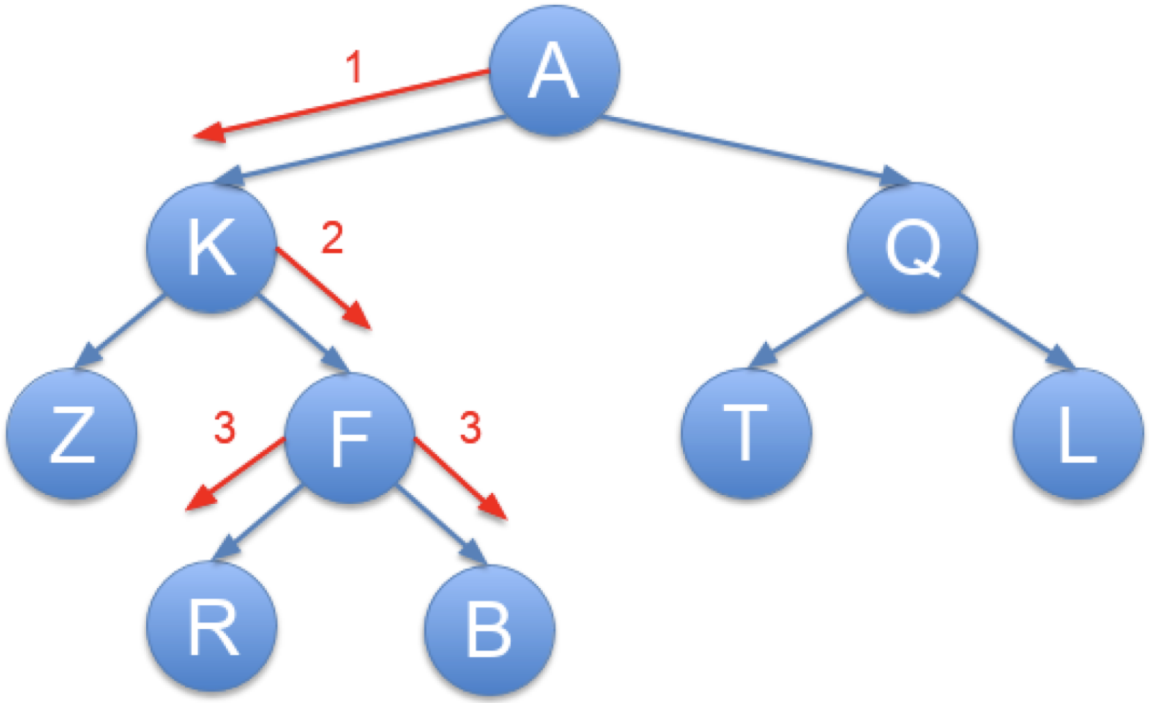



Count number of edges between root and node for depth

# From the depths to the heights


- *Height* of a tree: Number of edges between root and farthest leaf

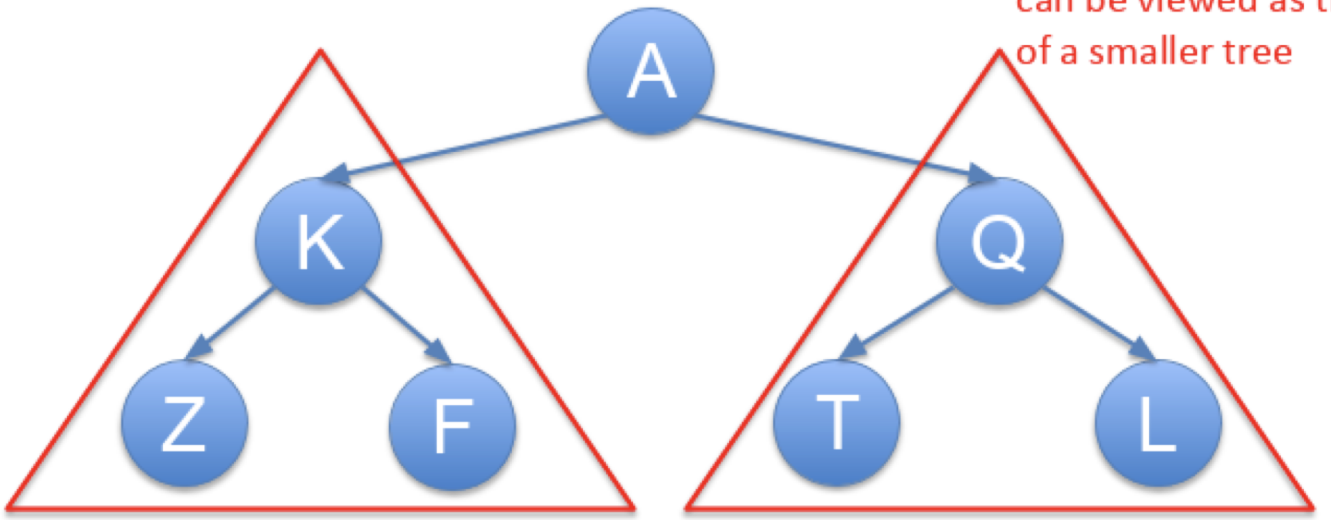
**What is the height of this tree?**



# Treeception

- *Subtree*: A tree that consists of a child and the child's descendants

Considered **recursive** because each sub-tree can be viewed as the root of a smaller tree




Subtree 1
Includes K, Z, and F

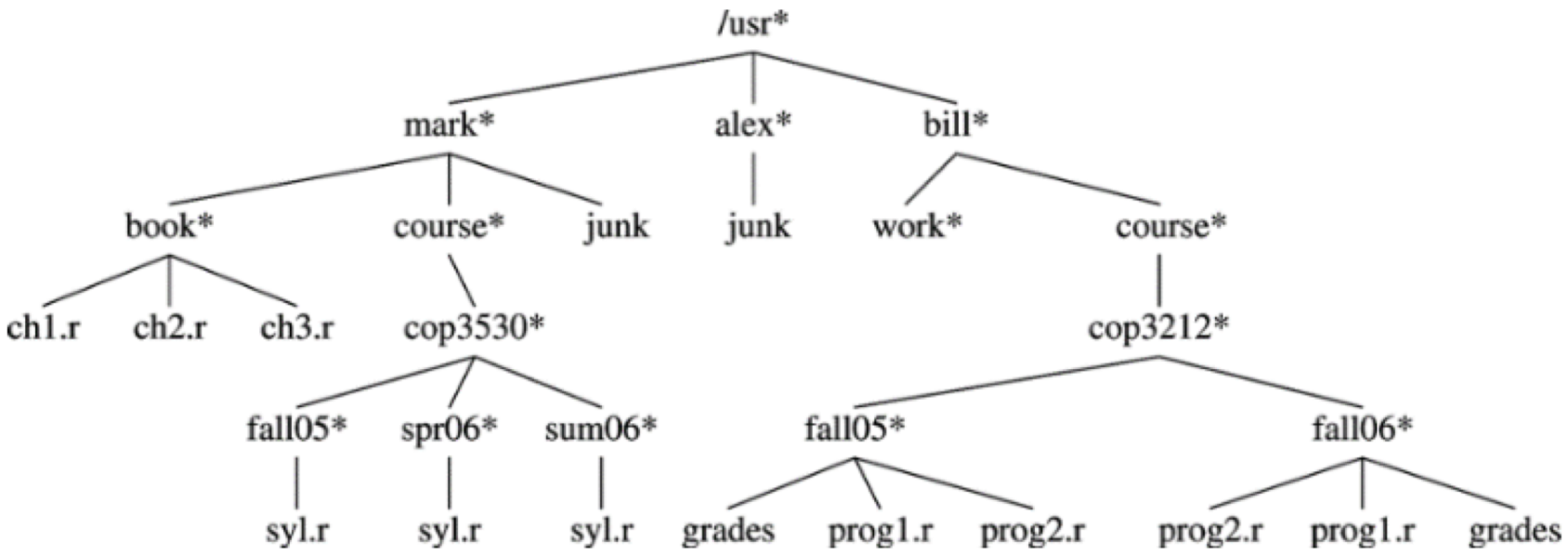
Subtree 2
Includes Q, T, and L

# Tree flavors


- Regular tree
- Regular binary tree
- Binary search tree
- Balanced binary search tree
  - AVL tree
  - Red-black tree

# Regular trees

Basic file systems are trees (when might this be untrue?)



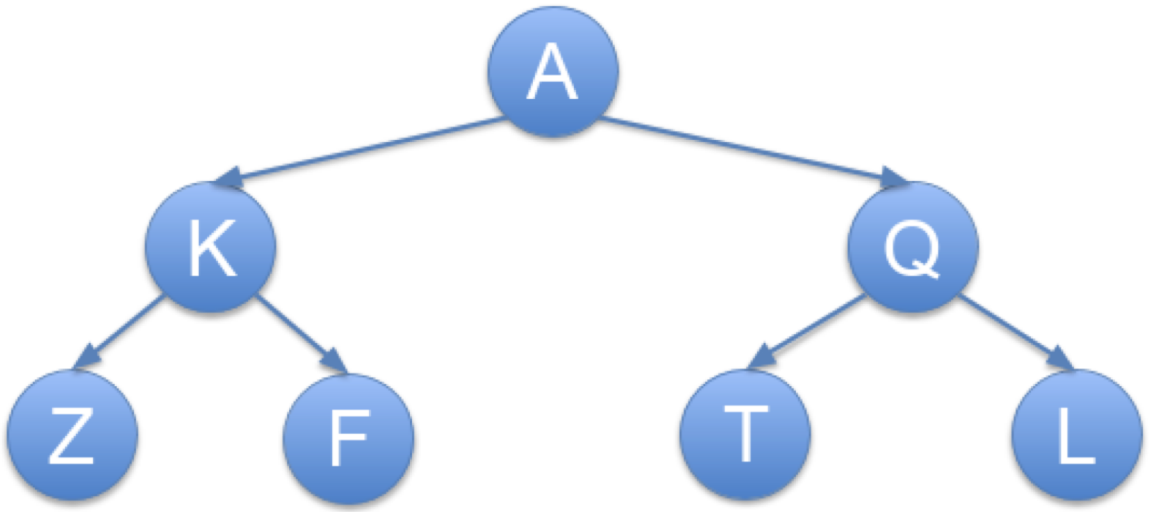
# Baobab trees

Native to Madagascar.

# Binary trees



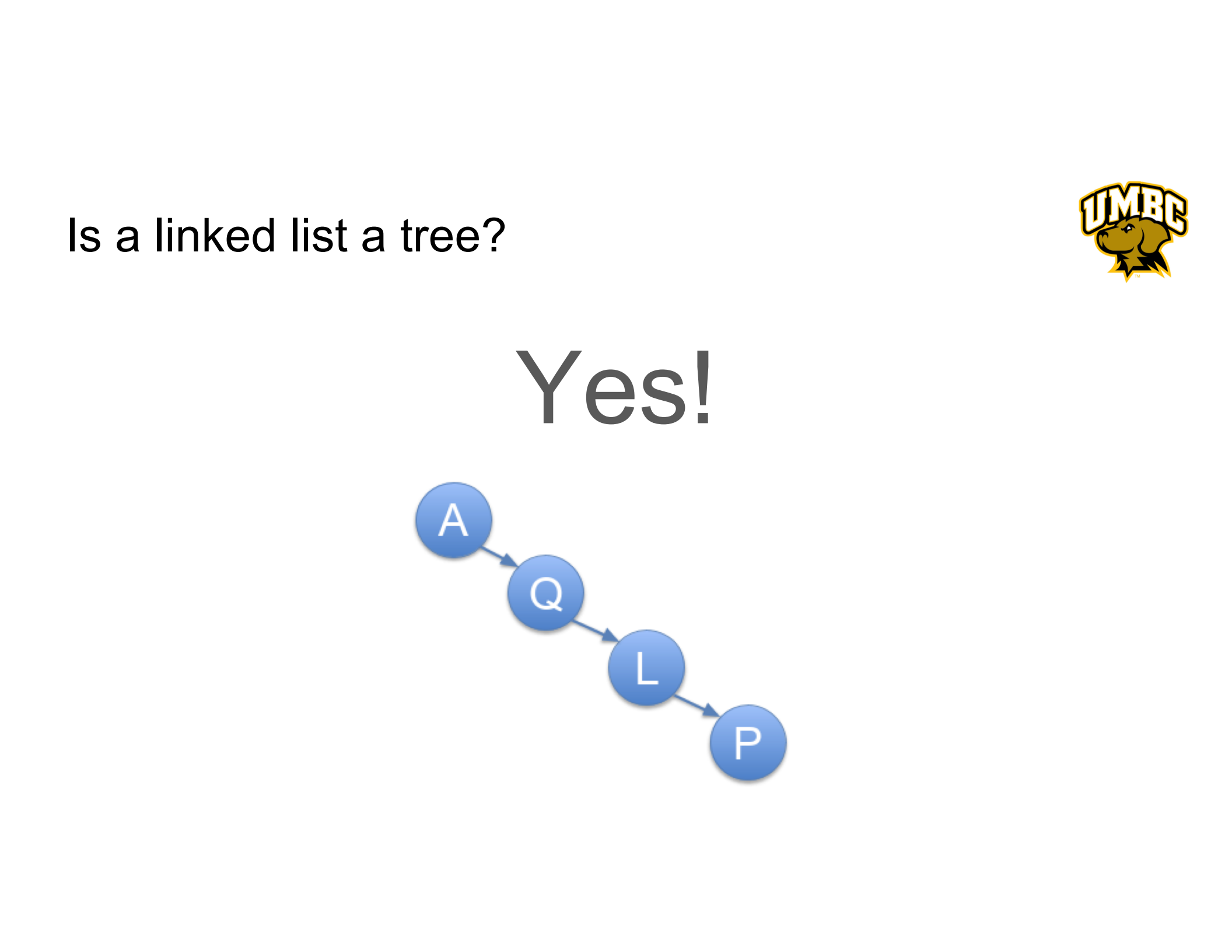

At most two children.
(Why do we have a special case for this?)



Is a linked list a tree?


?

Is a linked list a tree?

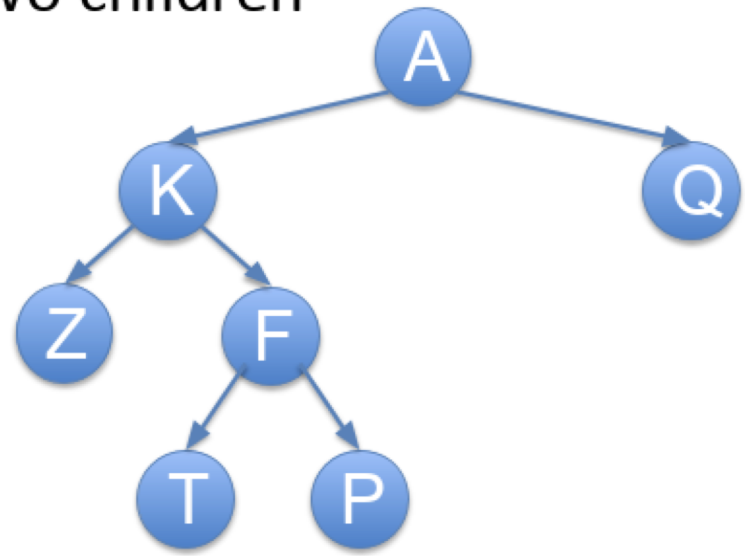



Yes!

Rainbow Eucalyptus Tree

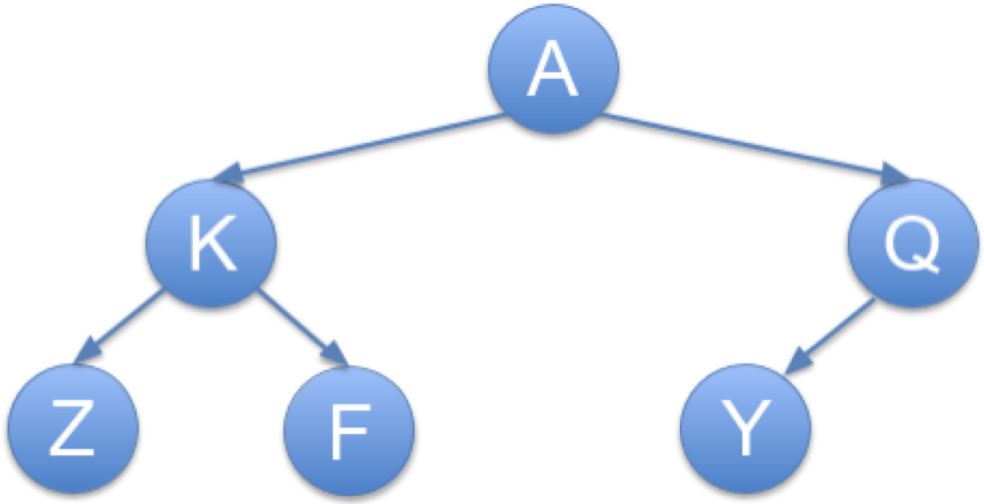
# Full binary tree


A binary tree is *full* if every node has exactly zero or two children



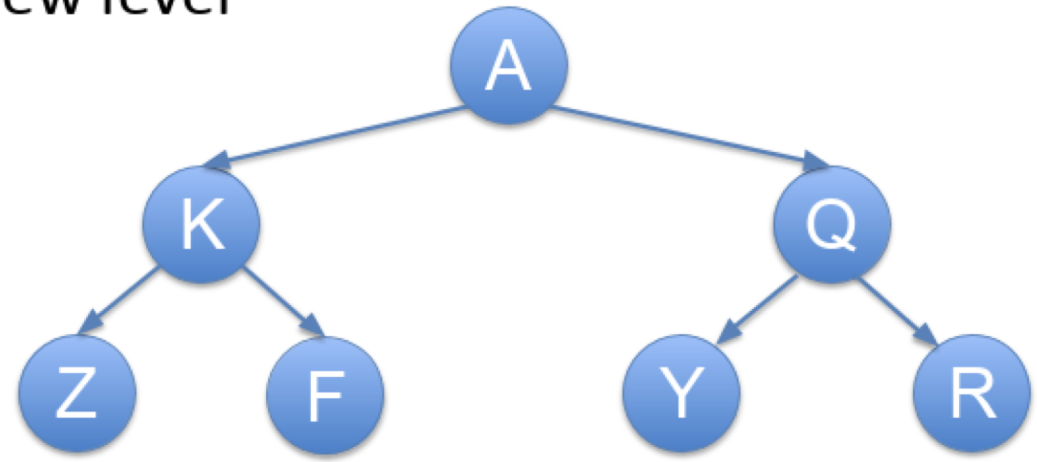
# Complete binary trees

A binary tree is complete if every level except possibly the last is completely filled *and* the nodes of the last level are as far left as possible.

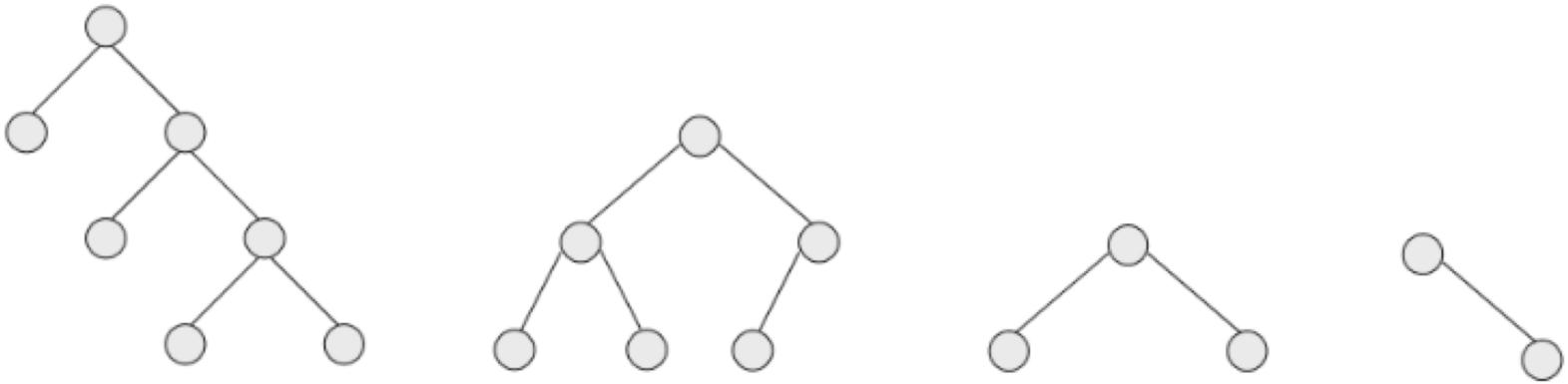


# Perfection!  Perfect binary trees


A binary tree is *perfect* if every leaf is on the same level, and adding one more node creates a new level



# Time to be judgemental of trees



# *Dragon's Blood* trees of Socotra Archipelago


What is the capacity of a binary tree of height H?

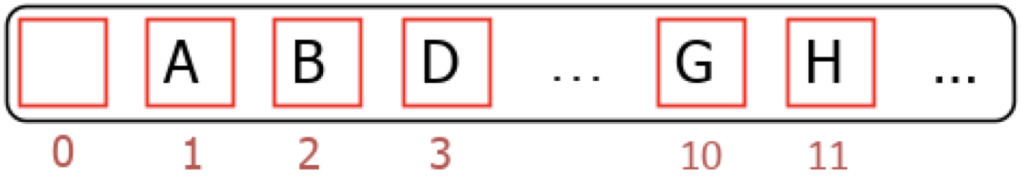
# What's under the hood?

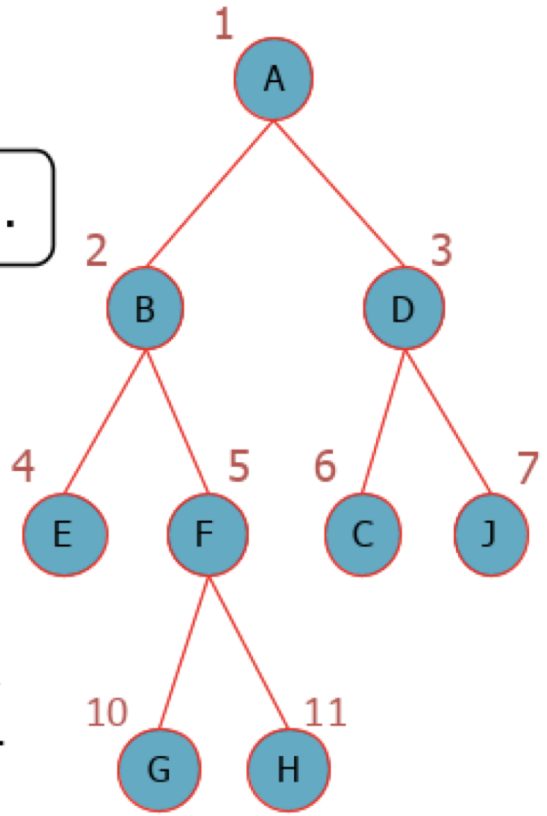

There are two main implementations for a binary tree, one that uses pointers, and another that uses an array.

We're going to live code the linked list one (partially).

# Array-Based Representation of Binary Trees

- Nodes are stored in an array A




- Node v is stored at A[rank(v)]
  - rank(root) = 1
  - if node is the left child of parent(node), rank(node) = 2 · rank(parent(node))
  - if node is the right child of parent(node), rank(node) = 2 · rank(parent(node)) + 1

# Dead Vlei Trees, Namibia

This is an actual photograph!
Not shopped!

# Traversals


A traversal is an algorithm that visits each item of the data structure.

Three types:

1. Preorder
2. Postorder
3. In order

Fortunately, we don’t have to traverse Californian Redwoods


- *Preorder*: Visit root, traverse left, traverse right
- *Inorder*: Traverse left, visit root, traverse right
- *Postorder*: Traverse left, traverse right, visit root

**Algorithm for Preorder Traversal**

1. `if` the tree is empty
2. Return

`else`

3. Visit the root.
4. Preorder traverse the left subtree.
5. Preorder traverse the right subtree.

**Algorithm for Inorder Traversal**

1. `if` the tree is empty
2. Return

`else`

3. Inorder traverse the left subtree.
4. Visit the root.
5. Inorder traverse the right subtree.

**Algorithm for Postorder Traversal**

1. `if` the tree is empty
2. Return

`else`

3. Postorder traverse the left subtree.
4. Postorder traverse the right subtree.
5. Visit the root.

# Tree sources

https://matadornetwork.com/trips/21-beautiful-trees-forests/