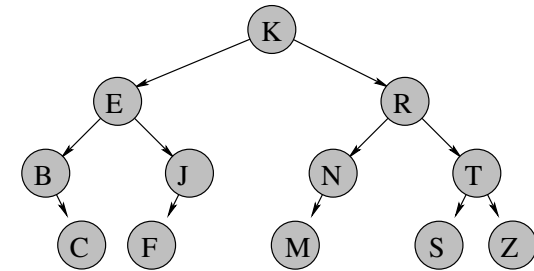


# Trees: Preliminaries

## Section 4.1

Ali Erkan  
Ithaca College

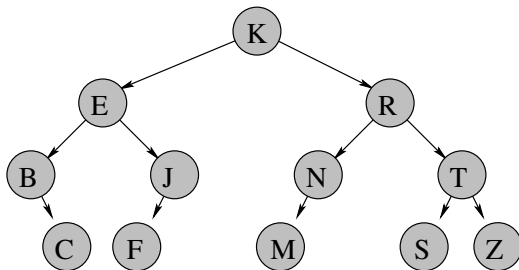
### Definitions



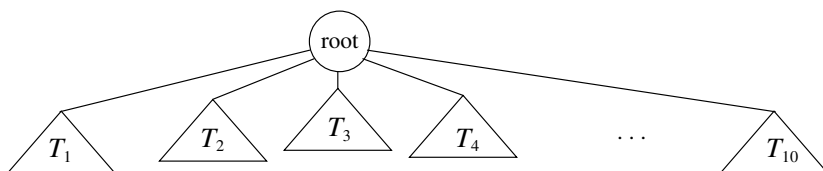
- ▶ **Root:** (A) The only node in a tree that has no parent
- ▶ **Internal node:** (A) A node that has one or more children
- ▶ **Leaf node:** (A) A node that has no children
- ▶ **Ancestor:** (A) Parent, grand-parent, great grand-parent, ..., of a node
- ▶ **Descendant:** (A) Child, grand-child, great grand-child, ..., of a node
- ▶ **Binary Tree:** (A) A tree where each node can have at most two children

### Definitions

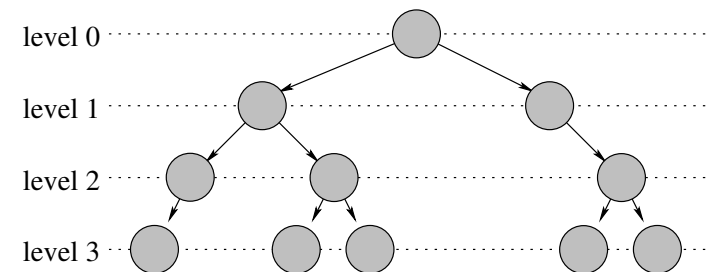
A **Data Structure** (DS), not Abstract Data Type (ADT), that creates a **parent-children** relation between its nodes



Can be defined recursively...

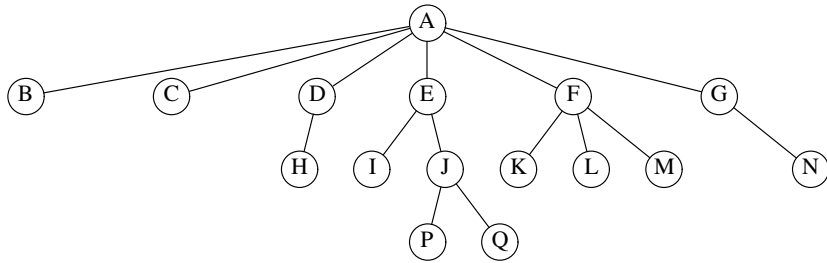


### Definitions



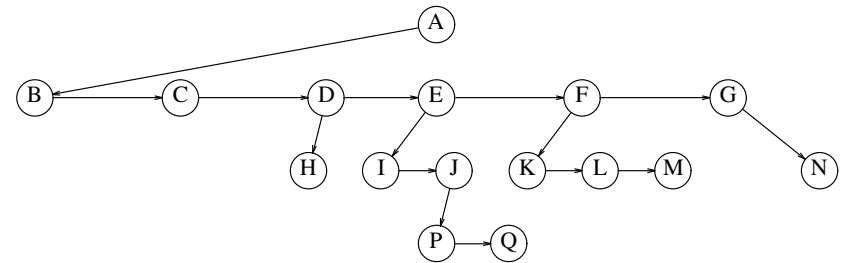
- ▶ **Node level:** The number of generations between the node at the root:
  - Root is level (A) 0
  - The children of root are level (A) 1
- ▶ **Node height:** The number of generations between the node and its most distant descendant:
  - The height of a leaf node is (A) 0
  - The height of the parent of a leaf is (A) 1

## Definitions Through Questions...



- ▶ If a tree has  $N$  nodes, how many edges does it have? **A**  $N - 1$
- ▶ Show one **path**. **A** A, D, H
- ▶ Show one pair of **siblings**. **A** K, L
- ▶ What is the **height** and **depth** of  $J$ ? **A** 1, 2
- ▶ What is the **height** and **depth** of  $E$ ? **A** 2, 1
- ▶ What is the **height** and **depth** of  $Q$ ? **A** 0, 3
- ▶ What is the **height** and **depth** of the root? **A** 3, 0

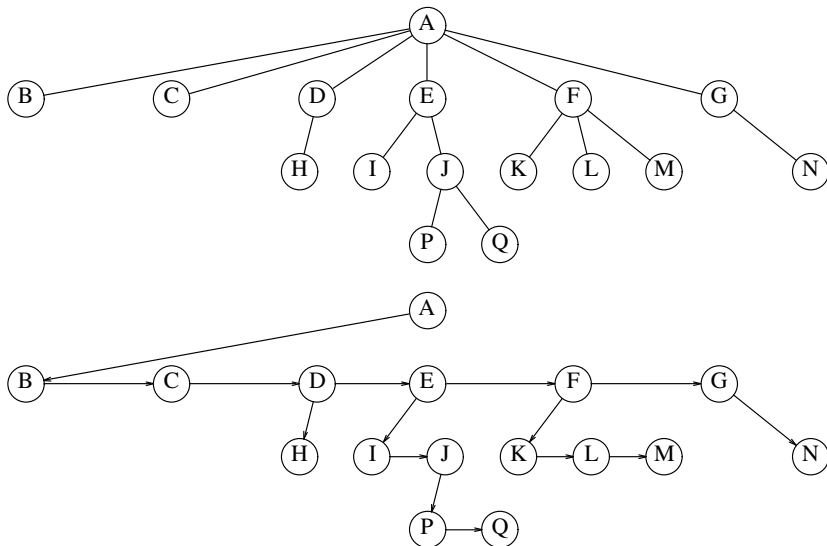
## How Could We Implement That?



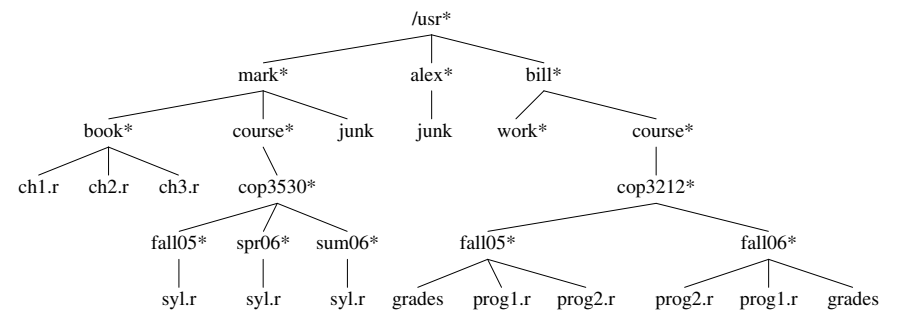
```

class TreeNode {
    Object element;
    TreeNode firstChild;
    TreeNode nextSibling;
}
    
```

## How Could We Implement A General Tree?



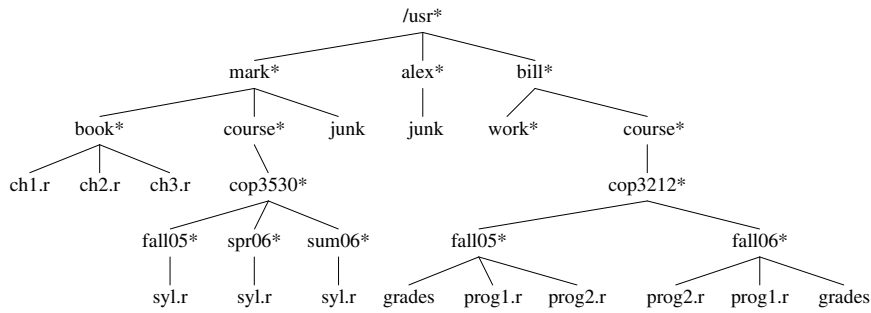
## UNIX File System: Recursive Traversal Driver



```

1 public void listAll() {
2     listAll( 0 );
3 }
    
```

## UNIX File System: Recursive Traversal

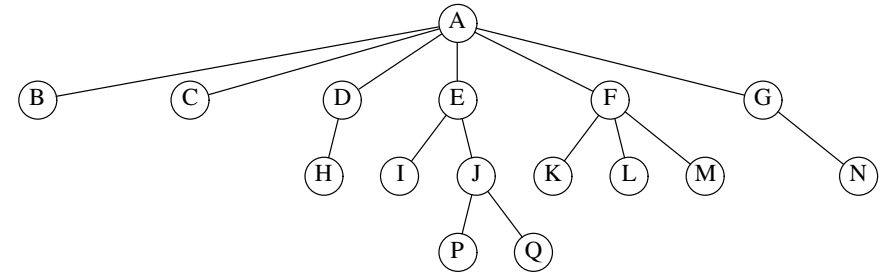


```

1 private void listAll( int depth ) {
2   printName( depth );
3   if ( isDirectory() )
4     for each file c in this directory
5       c.listAll( depth + 1 );
6 }

```

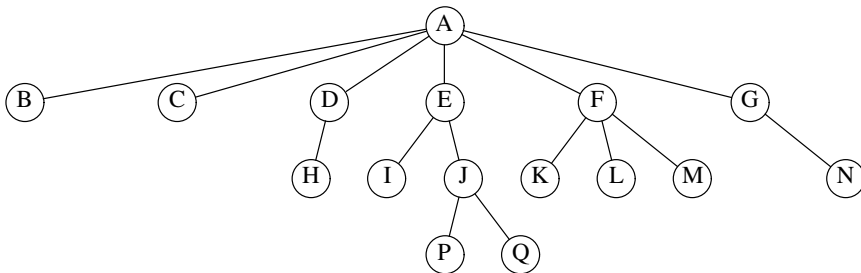
## Tree Traversal



► What is the **postorder traversal** of the above tree?

- **Rule:** Ⓐ Children come before the parent
- **Result:** Ⓐ B C H D I P Q J E K L M F N G A

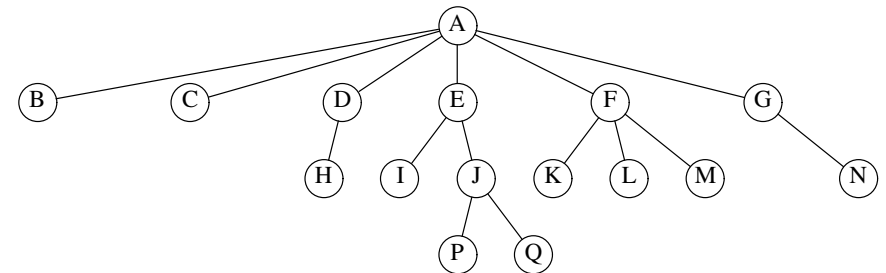
## Tree Traversal



► What is the **preorder traversal** of the above tree?

- **Rule:** Ⓐ Parent comes before children
- **Result:** Ⓐ A B C D H E I J P Q F K L M G N

## Tree Traversal

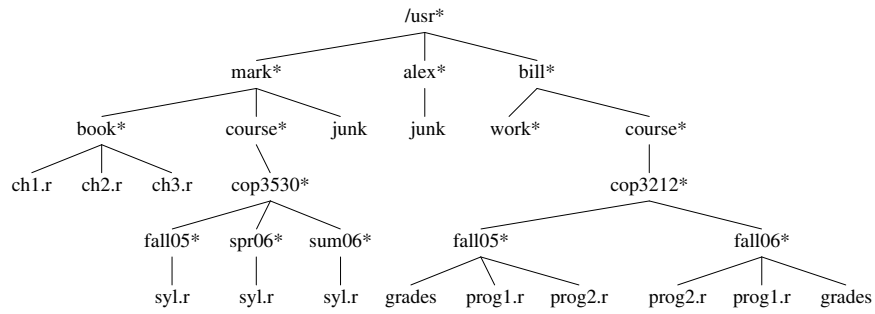


► What is the **inorder traversal** of the above tree?

- **Rule:** Ⓐ Parent processed in between the two children
- **Result:** Ⓐ This is a trick question since inorder traversal is an operation specifically for binary trees. However, inorder traversal can be “forced onto” arbitrary trees by “processing” a parent  $N - 1$  times if that parent has  $N$  children.

# UNIX File System: Recursive Traversal

---



► Given the above view of the file system:

- Give one example of why we may want a **preorder** traversal?  
Ⓐ To print the names of all the files/dirs.
- Give one example of why we may want a **postorder** traversal?  
Ⓐ To find the total size of all the files.