Starting Chapter 4
Trees! (Trees, Glorious Trees)

CptS 223 - Fall 2017 - Aaron Crandall

Today's Agenda

- Announcements
- Chapter 3 summary
- Starting chapter 4 - Trees!

Announcements

- Day of Doom: Mid term #1 - Oct 2nd (it's a Monday)
  - Algorithms & Big-O analysis
  - Linux questions (commands, basic OS concepts)
  - Trees - BST, AVL, Red-Black, B+, misc
  - ADTs - Stack, List, Vector, Queue
- Next MA due out soon && HW1 should go out this afternoon
- PPEL - Career Fair Prep #3 – Workshop: Career Fair Introductions and Informational Interviews - September 15 @ 4:10 pm - 5:30 pm - Sloan 169 https://vcea.wsu.edu/event/career-fair-prep-3-workshop-career-fair-introductions-and-informational-interviews-2/

Thing of the Day

More giant robots!

https://mindyourdirt.com/2016/08/10/james-and-the-giant-atomic-death-robot/

Touching on Chapter 3

- Any questions about chapter 3?
  - Lists
  - Vectors
  - Stacks
  - Queues
  - Implementation
  - Iterators
- Here’s one for you: Would it be a good idea to implement a Queue class using the vector<T> ADT from the STL? What would be a risk?

Let’s do chapter 4

Trees!

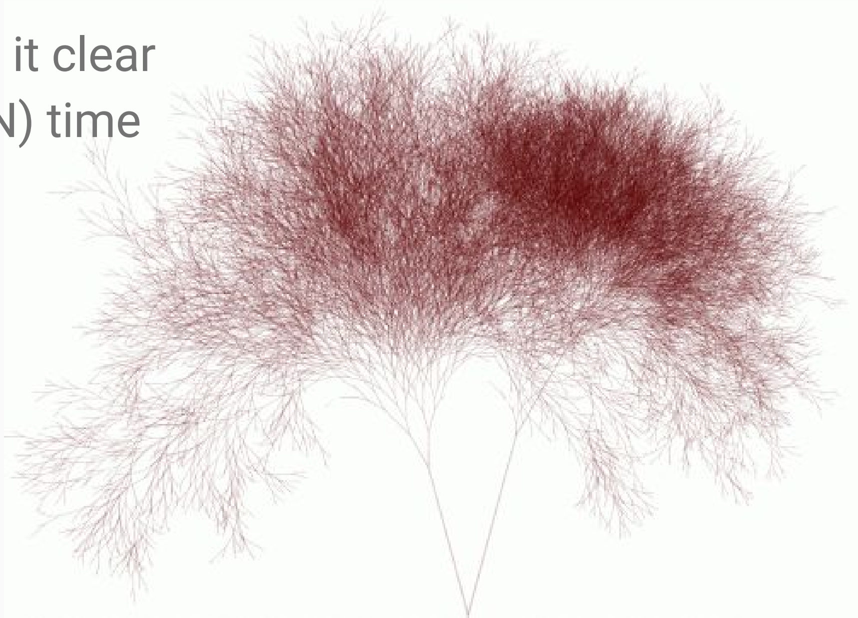
What you’ve all been waiting for.




FYI: I grew up in Portland

Trees - The roots

- Binary search tree (and variants)
- Used for filesystems - UNIX style makes it clear
- When done properly, do things in O(log N) time



Defining Trees Recursively

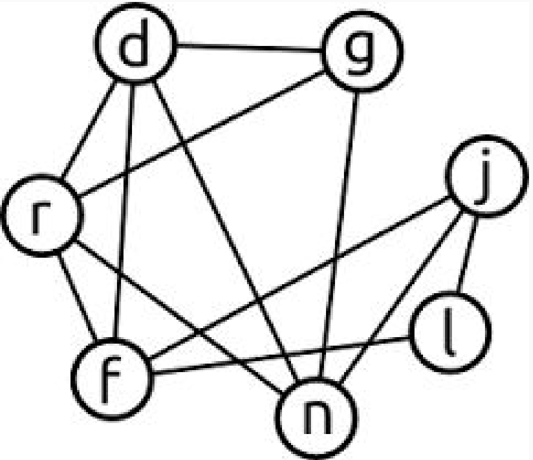
- Collection of nodes
- Collection may be empty []
- Otherwise, has a distinguished node, r, called the root
  - With zero or more nonempty subtrees, T_1, T_2,... T_k, each of which is connected by a directed edge from r
- A tree is a connection of N nodes, 1 root, and N-1 edges
- There are no cycles (loops) in trees
  - More of this in the chapter on graphs
  - Trees are actually Directed Acyclic Graphs (DAGs)

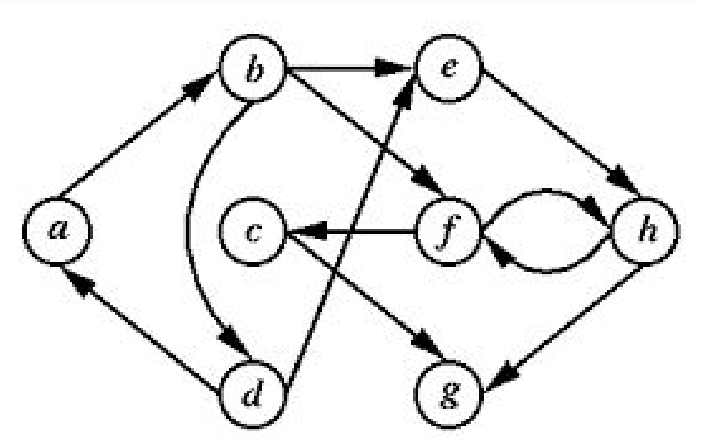
Terms

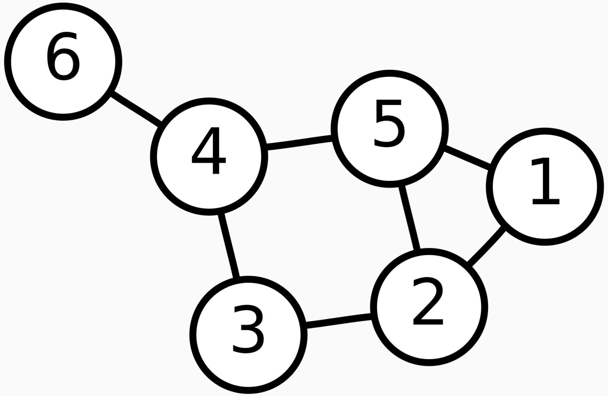
Root: Top node in a tree
Edge: Connection between two nodes in a graph
Child: A node pointed to by another one by a directed edge
Parent: A node with a directed edge towards another node
Leaves: Nodes with no children
Siblings: Nodes with the same parent
Path: Sequence of nodes $n_1, n_2 \ldots n_k$ such that n_i is the parent of $n_{(i+1)}$
Path Length: number of edges on the path
Depth: length of the unique path from the root to a given node
Height: length of the longest unique path from a node to a leaf

Tree vs. Graph
(we'll see graphs in detail later)

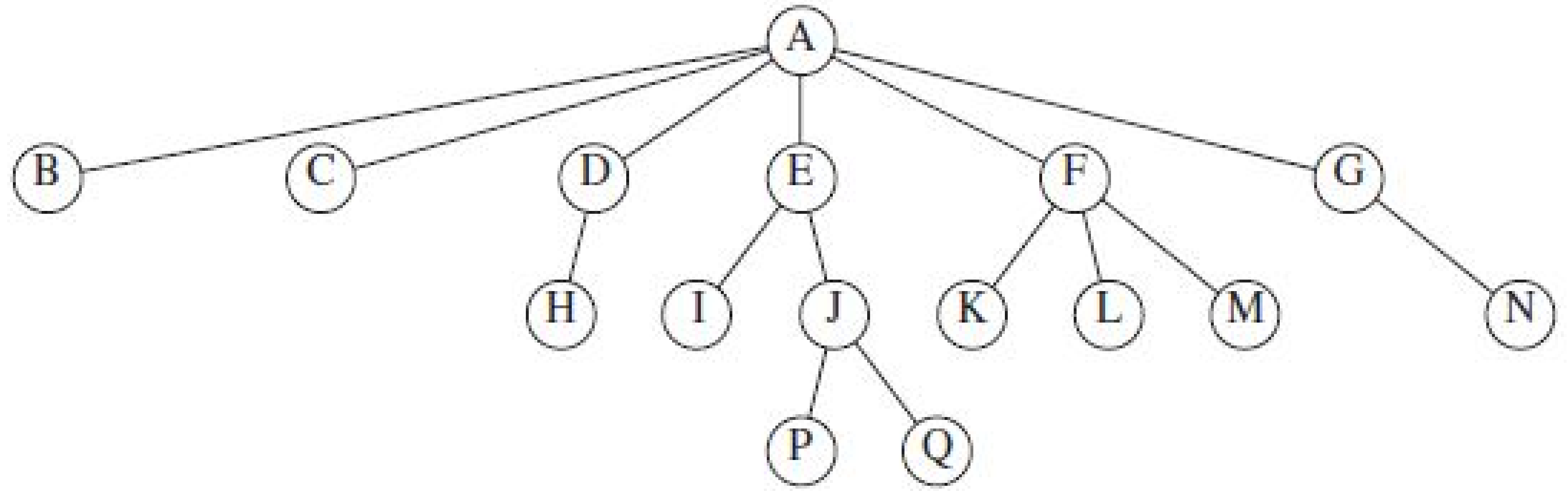
- Trees are graphs, but special ones:
  - A tree is an directed graph in which any two vertices are connected by exactly one path. In other words, any connected directed acyclic graph is a tree.







Examples of the terms



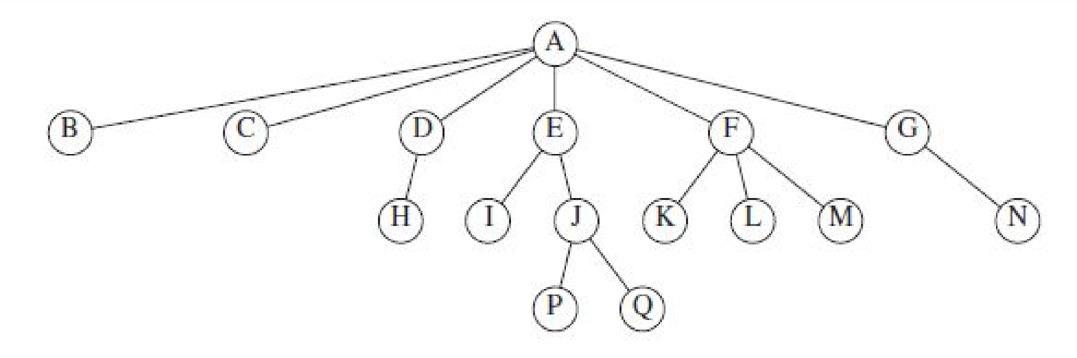
Consider a BST as an ADT

- ADT operations:
  - Add
  - Remove
  - Size
  - Contains
  - Union
  - Find
  - Empty

- Special operations:
  - height()
  - depth()
  - findMin()
  - findMax()
  - inOrder()
  - preOrder()
  - postOrder()
  - levelOrder()

Basic binary tree implementation

```
struct node
{
  int key_value;
  struct node *left;
  struct node *right;
};
```



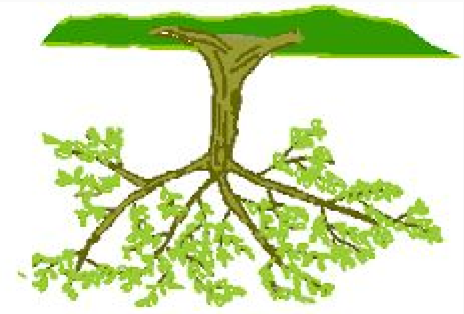
How do we get arbitrary numbers of children for a node?

Perhaps a vector<*node> solution?

```
struct node
{
  int key_value;
  vector<*node> children;
};
```

Any significant impacts using this approach?

- Struct size?
- Vector behaviors?



CompSci trees.
Perhaps we’re in Australia?

Book’s solution - Pointer traversal!

```
struct TreeNode
{
        Object element;
        TreeNode *firstChild;
        TreeNode *nextSibling;
};
```

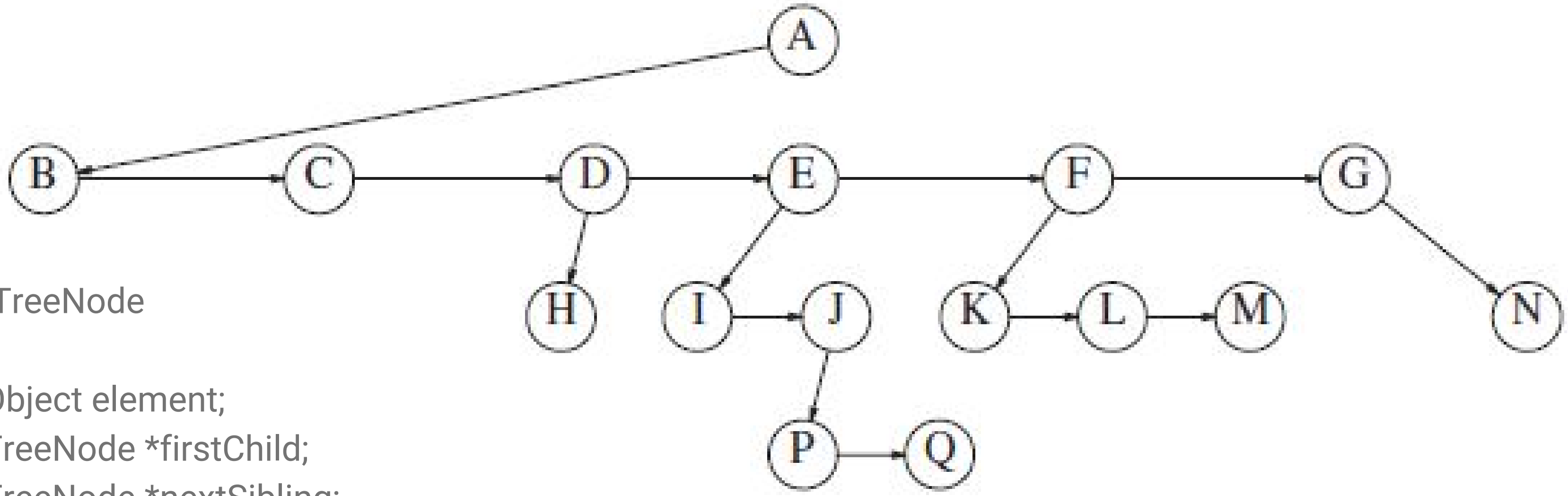
(I want green things back!)

This works great.

- Unlimited children for a node!
- Unlimited siblings for a node!
- Very pointery

But what does that do to the structure of the tree?

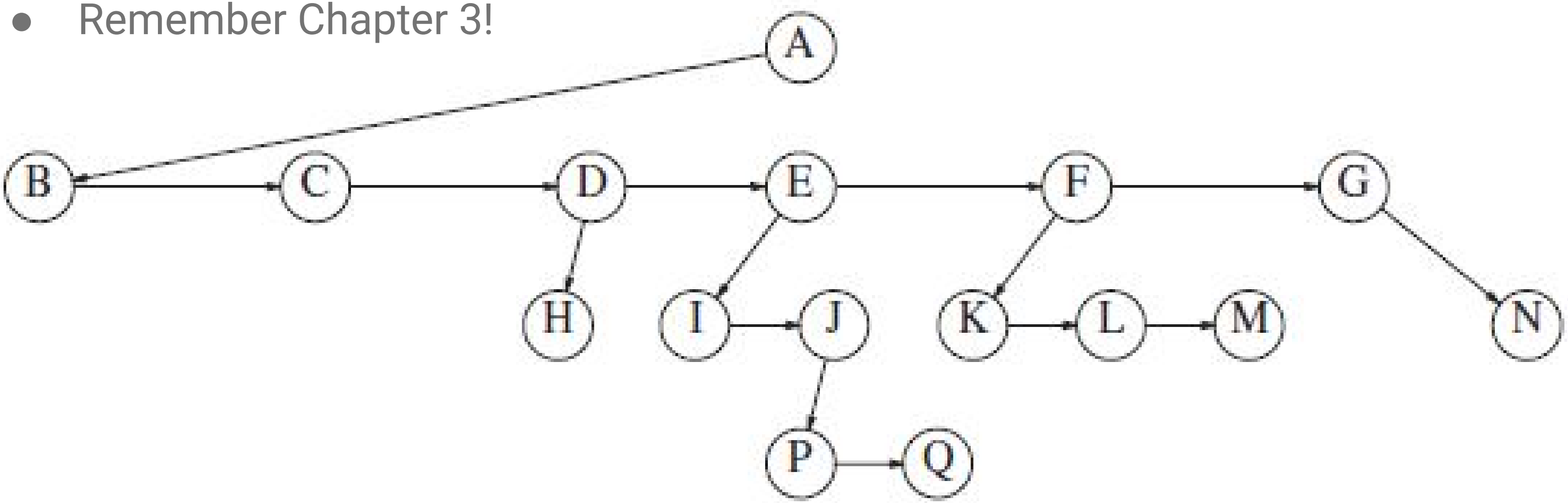
The result is very different




```
struct TreeNode
{
        Object element;
        TreeNode *firstChild;
        TreeNode *nextSibling;
};
```

What are the implications of this for runtime? This is a tree with 16 nodes, fyi

- Consider the layout of the nodes
- Remember Chapter 3!



Linux tricks of the day!

Command: mtr

- Meters a route to a network host: mtr google.com

Command: wc

- Counts words (or lines, or chars) in a file: wc [filename]
- To count lines: wc -l [filename]

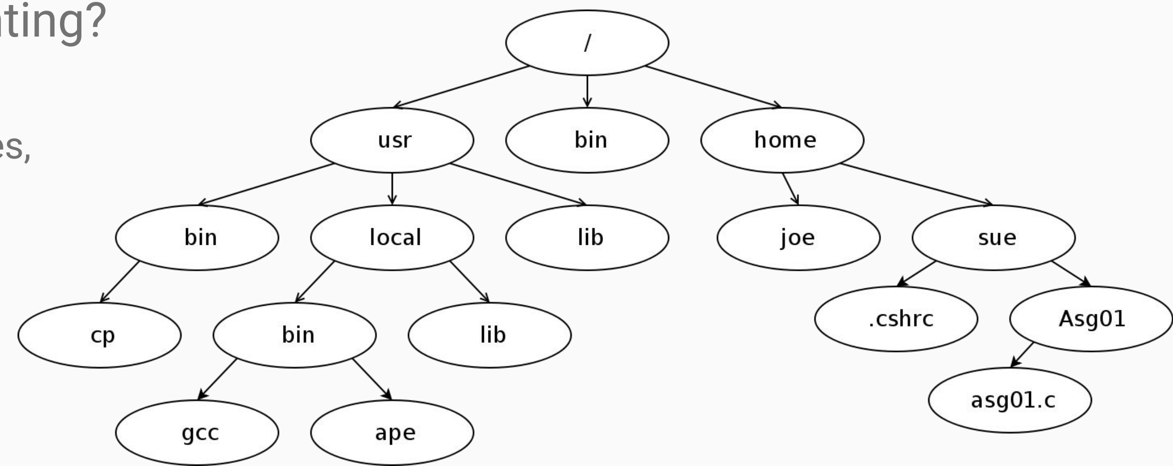
Command: last

- Shows last people to login to the computer: last

Command: pstree

If there’s time, consider the UNIX filesystem

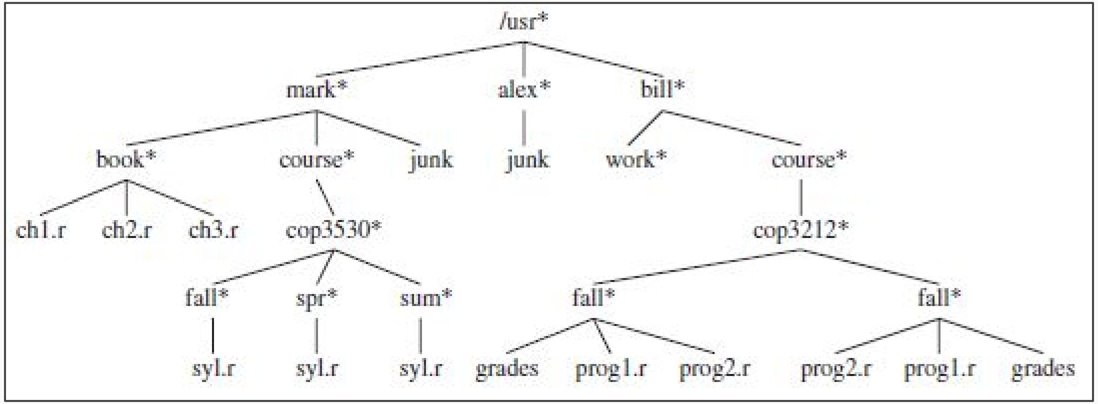
- It’s a tree! (as are most filesystems)
- Ever wonder why I keep spouting out about “the ROOT of the filesystem?”
  - Yes, it was designed and named by computer scientists, not marketers
- What is each “/” representing?
- A node holds what?
  - Pointers to: directories, files, anything else? (metadata!)



Example tree - and a question!

How can nodes be one of *two* things and still be in the same tree?

Directory OR File?



What are they inserting to the tree by? Also, the fully qualified path is kind of the true identity of the file, not just the filename.

<Show tree command here>

Limits of the UNIX filesystem

- Well… the limits of the current Linux filesystem***s***
- Linux supports MANY filesystems: ext4, ext3, ReiserFS, NTFS, HFS+, BSD https://en.wikipedia.org/wiki/Category:File_systems_supported_by_the_Linux_kernel
- There's many filesystems out there, just FYI: https://en.wikipedia.org/wiki/List_of_file_systems
- When a filesystem spec says that it supports X# of directories or files, what does that mean?

Trees get bigger!

OS	Filesystem	Files per dir	Total Files/volume	Vol Size	Filename len
DOS/Win95	FAT16	512	65,000	4 GB	8.3
Win98	FAT32	65k	268,435,437	2 TB	255
Win2k..Win10	NTFS (B-tree)	4,294,967,295	4,294,967,295	256 TiB	255
Linux 2.6-3.0	EXT3	32,000		32TiB	255
Linux 3.0-4.x	EXT4	4 billion	4 billion	1 EiB	255
LInux 4.x	Btrfs (B-Tree)	2^64	2^64	16 EiB	255

For Friday! - Read Chapter 4.0-4.4