COIN 315: Data Structures

Dr. Paul E. West

Department of Computer Science
Charleston Southern University

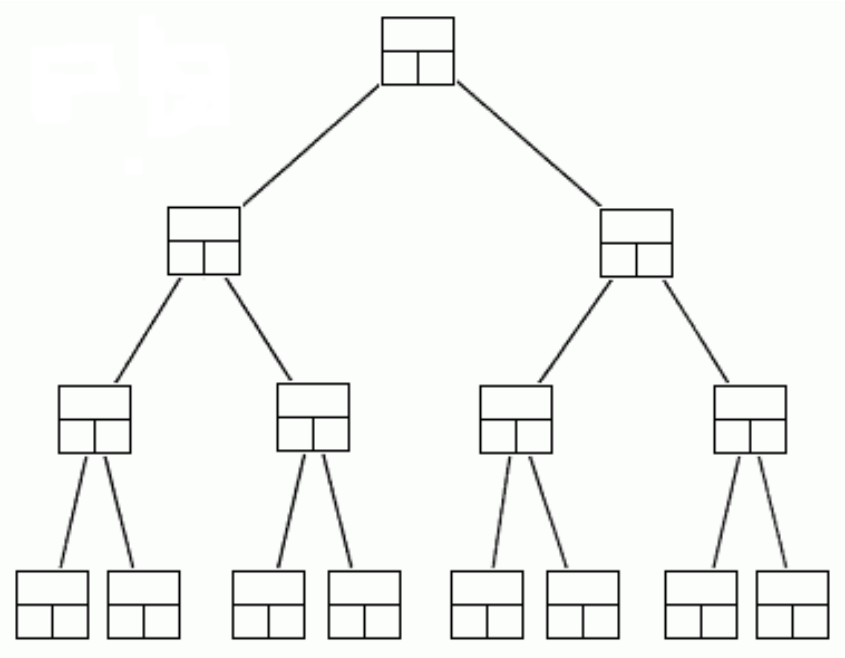
August 24, 2015

About the Professor

- PhD from Florida State University in Computer Science
- Faculty Experience:
  - Charleston Southern: Assistant 2015-Present
  - College of Charleston: Adjunct 2013-2014
- Work Experience:
  - Google (2014): Android Bluetooth/Wi-Fi/Telephony
  - SPAWAR (2009-2014, 2015): Communication systems
  - Various Government Contracts: (2016 - Present)
  - DenimGroup (2004-2005): Start-up; web design and network security

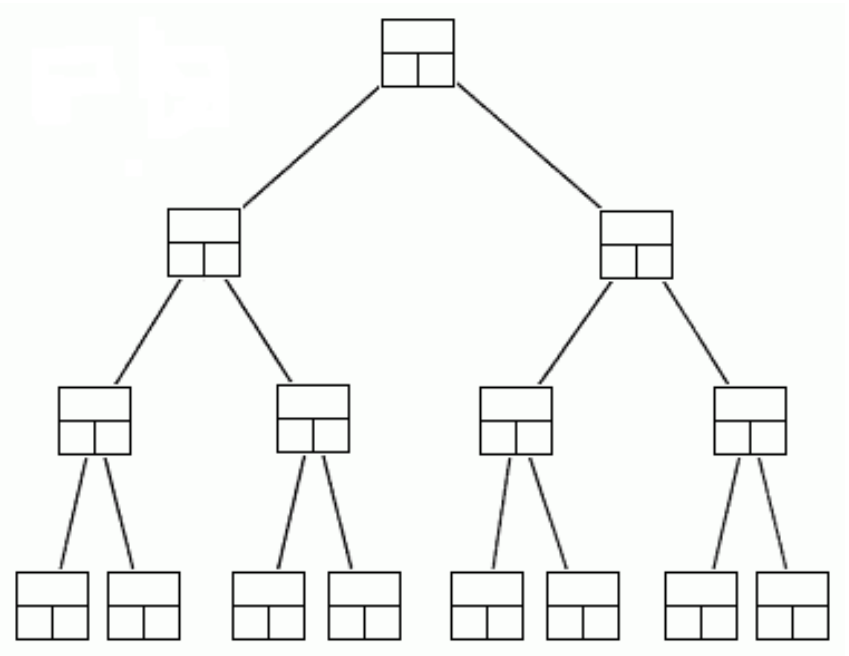
What is a Data Structure?

- A data structure is a scheme for organizing data in the memory of a computer.
- Some of the more commonly used data structures include lists, arrays, stacks, queues, heaps, trees, and graphs.



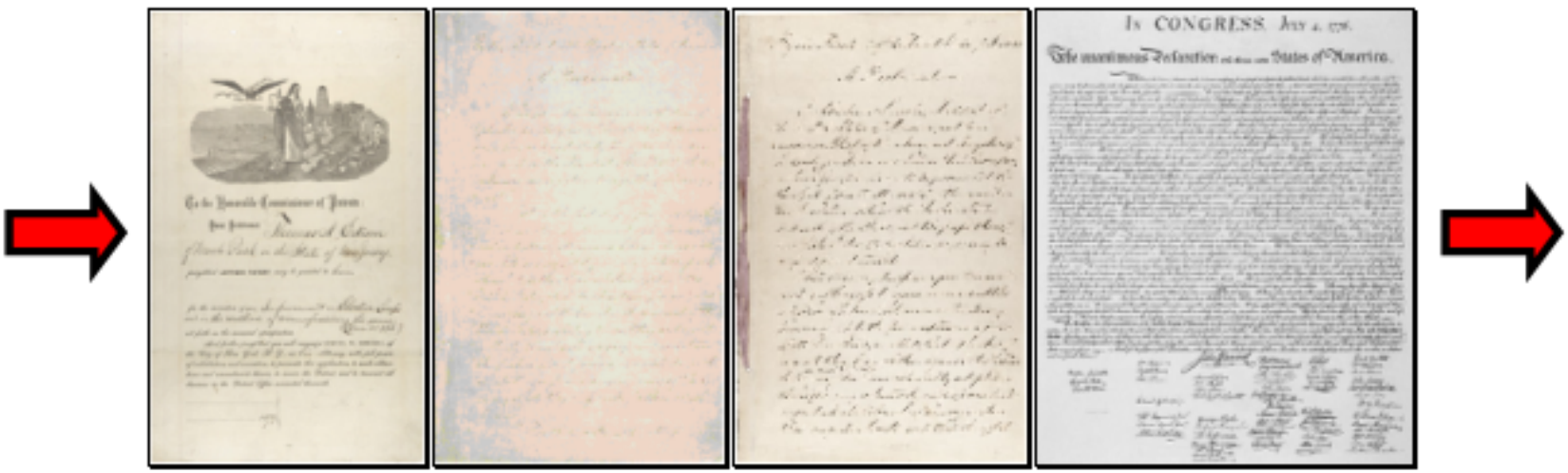
Why?

- The way in which the data is organized affects the performance of a program for different tasks.
- Computer programmers decide which data structures to use based on the nature of the data and the processes that need to be performed on that data.



Queue

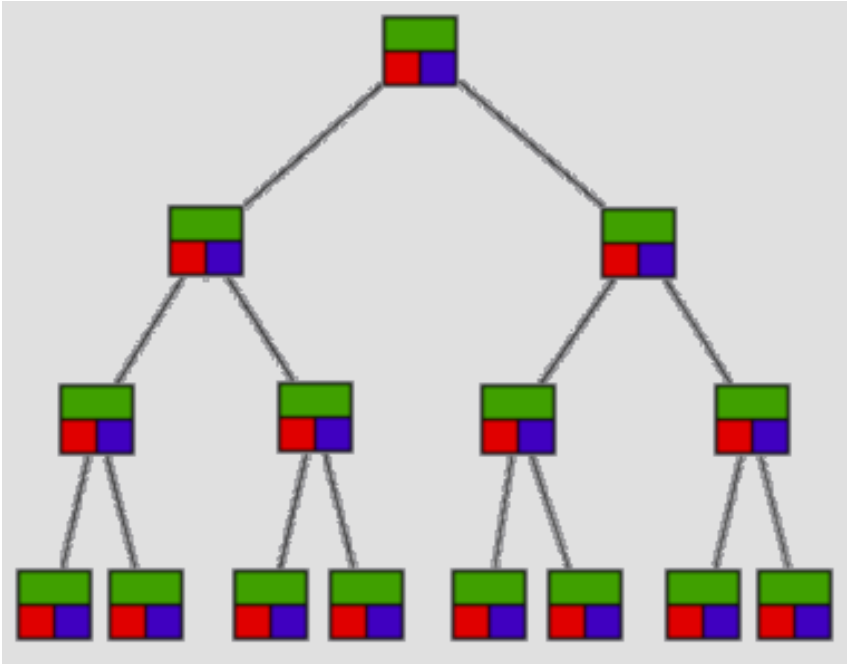
- A queue is an example of commonly used simple data structure. A queue has beginning and end, called the front and back of the queue.



- Data enters the queue at one end and leaves at the other. Because of this, data exits the queue in the same order in which it enters the queue, like people in a checkout line at a supermarket.

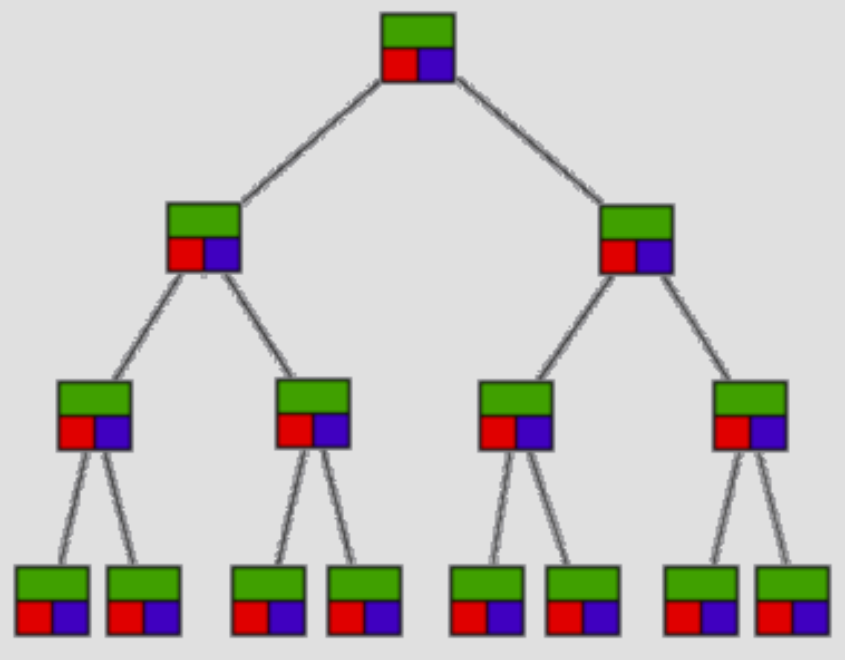
Binary Trees

- A binary tree is another commonly used data structure. It is organized like an upside down tree.
- Each spot on the tree, called a node, holds an item of data along with a left pointer and a right pointer.



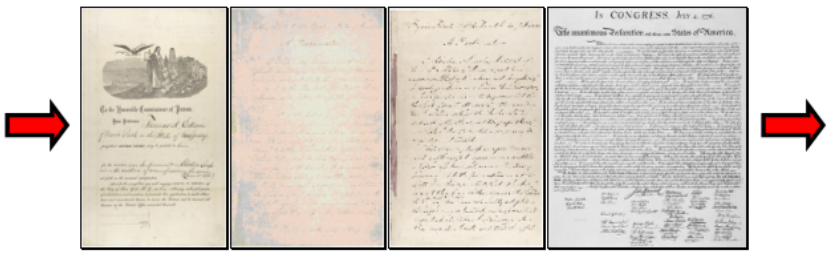
Binary Trees

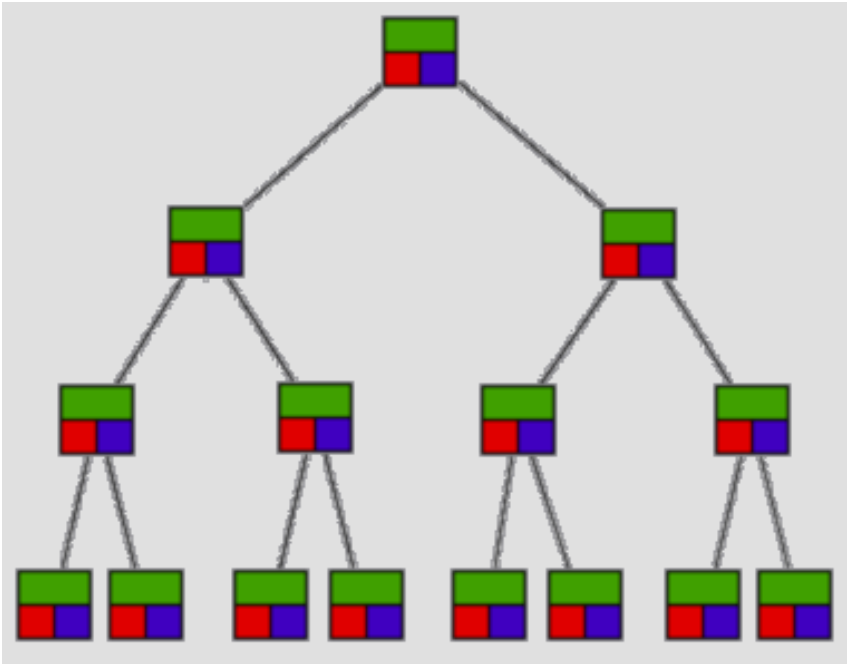
- The pointers are lined up so that the structure forms the upside down tree, with a single node at the top, called the root node, and branches increasing on the left and right as you go down the tree.



Decisions Decisions...

- When do we choose one over the other?





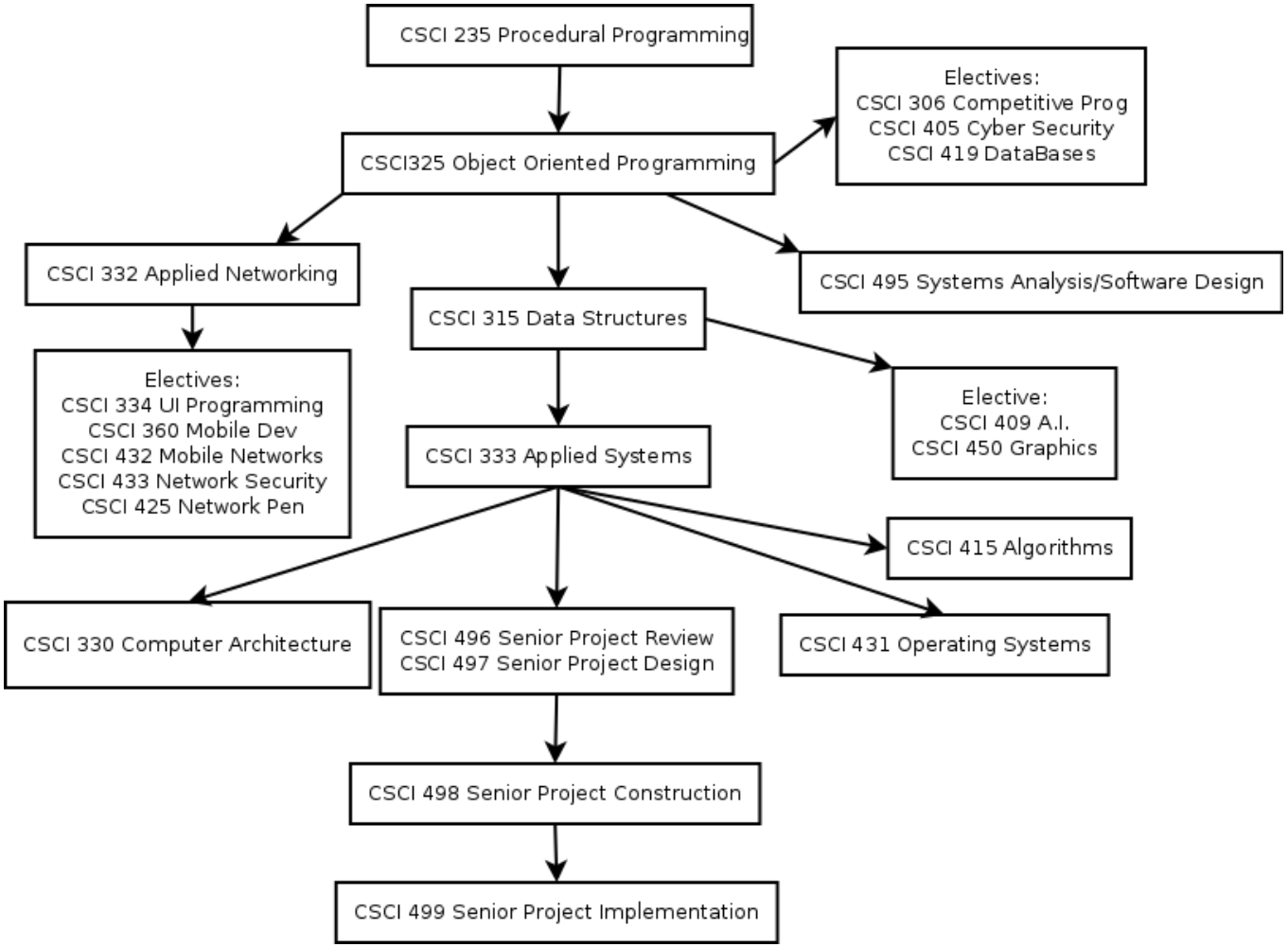
Syllabus

Lets go over the syllabus...

Warning:

- Most students consider this class high(est) workload
- This class really prepares your for work experience

Do Not Drop!


CSCI 235 Procedural Programming
CSCI325 Object Oriented Programming
Electives:
CSCI 306 Competitive Prog
CSCI 405 Cyber Security
CSCI 419 DataBases
CSCI 332 Applied Networking
CSCI 315 Data Structures
CSCI 495 Systems Analysis/Software Design
Electives:
CSCI 334 UI Programming
CSCI 360 Mobile Dev
CSCI 432 Mobile Networks
CSCI 433 Network Security
CSCI 425 Network Pen
Elective:
CSCI 409 A.I.
CSCI 450 Graphics
CSCI 333 Applied Systems
CSCI 415 Algorithms
CSCI 330 Computer Architecture
CSCI 496 Senior Project Review
CSCI 497 Senior Project Design
CSCI 431 Operating Systems
CSCI 498 Senior Project Construction
CSCI 499 Senior Project Implementation

- git: used as our repository of code, lectures, and assignments
- gcc: a compiler toolchain used to translate our C++ programs into machine executable programs
- cxxtest: a test suite to build and automate are testing
- make: a tool to simplify building, testing, cleaning, and other utilities
- text editor: use to write your programs
- bash: our shell/Command Line Interface we use to run our programs
- These tools make up our development environment
- How is this different than an IDE?

Logging into our machines

- Our lab uses Linux (Debian 8)
- Please login with your standard CSCI login
- ***Note:*** These installations of Debian has different GUIs to utilize, you may choose your preference by clicking on the gear when entering your password!
  - I recommend starting with KDE or Gnome.
  - I personally use Enlightenment.
- You may run Linux at home naively or as a VM, I recommend Mint or Ubuntu if you are new.

Terminal

- terminal: (terminal emulator), is a text-only window in a graphical user interface (GUI) that emulates a console.
- It is were we use our CLI
- xterm, gterminal, lxdeterminal
- Run whatever you want!

UNIX Command Format

- UNIX commands can be very simple one word commands, or they can take a number of additional arguments (parameters) as part of the command. In general, a UNIX command has the following form:
  command options(s) filename(s)
- The command is the name of the utility or program that we are going to execute.
- The options modify the way the command works. It is typical for these options to have be a hyphen followed by a single character, such as -a. It is also a common convention under Linux to have options that are in the form of 2 hyphens followed by a word or hyphenated words, such as –color or –pretty-print.
- The filename is the last argument for a lot of UNIX commands. It is simply the file or files that you want the command to work on.

Common UNIX Conventions

- In UNIX, the command is almost always entered in all lowercase characters.
- Typically any options come before filenames.
- Many times, individual options may need a word after them to designate some additional meaning to the command.

Man Page

- The man command allows you to access the MANual pages for a UNIX command.
- To get additional help on any of the commands listed below, you can always type man name_of_command at the command prompt.
- Examples:
  man ls
  man cd

Navigation and Directories

- ls: lists the contents of a directory
  - l: long directory listing
  - a: lists all files, including files which are normally hidden
  - F: distinguishes between directories and regular files
  - h: ? Look it up using man
- pwd: prints the current working directory
- cd: changes directories
  - The difference between relative and absolute paths.
  - Special characters ., .., and ~
- mkdir: creates a directory
- rmdir: removes a directory (assuming it is empty)
- If you get an error that the directory is not empty even though it looks empty, check for hidden files.

Commands

- cat : shows the contents of a file, all at once
- more : shows the contents of a file, screen by screen
- less : also shows the contents of a file, screen by screen
- head : used to show so many lines form the top of a file
- tail : used to show so many lines form the bottom of a file

Commands

- alias: creates an alias for a command.
  - Aliases can be placed in your /.bashrc login script.
  - Example: alias rm 'rm -i'.
- date: shows the date and time on the current system
- who: used to print out a list of users on the current system
- hostname: prints the hostname of the current computer
- whoami: prints your current username

Commands

- The pipe (|) creates a channel from one command to another. Think of the pipe as a way of connecting the output from one command to the input of another command.
- The pipe can be used to link commands together to perform more complex tasks that would otherwise take multiple steps (and possibly writing information to disk).
- Examples:
  - Count the number of users logged onto the current system.
    - The who command will give us line by line output of all the current users.
    - We could then use the wc -l to count the number of lines...
    - who | wc -l
  - Display long listings in a scrollable page.
    - ls | less

Commands

- ps : lists the processes running on the machine.
  - ps -u username lists only your processes.
  - ps -a : lists all processes running on the machine.
  - The PID column of the listing, provides the information required by the kill command.
- kill : terminates a process
  - kill process_id : sends a terminate signal to the process specified by the process_id (PID).
  - In cases where the terminate signal does not work, the command “kill -9 process_id" sends a kill signal to the process.
- nice : runs a process with a lower priority.

Editor War!

- vi: (vee-eye) large learning curve, but can increase productivity
- emacs: has numerous features and more beginner friendly
- “standard editors:”
  - kate
  - gedit
  - tea
  - jed
- to exit vim: pres ESC : x <enter>
- to exit emacs: ctrl-x, ctrl-c
- I recommend starting with a standard one and then branching out.
- I use vi

Git

- Developed by Linus Torvalds
- Version Control Tool
  - Subversion
  - CVS
- Open Source
- Free!
- Just a folder in a directory

Git Golden 5

- clone/init
- Add
- Commit
- Push
- Pull

Joining the Class

- Create a GitHub account (or use an existing one) and login.
- email me you username: pwest@csuniv.edu
- You should (eventually) receive an email to join CSU
- Click the link and join
- Click on the csci-315-fall-2015 repository
- Click fork
- Now clone your repository:
  $ git clone <your repo>
- Lets look through the repository...

Auto Grader

- I will attempt to maintain an auto grader for this class.
- It will function similarly to the one used in CSCI 325.
- **Note:** I will **NOT** hand out the test cases from the auto grader. It is imperative that you learn how to write your own test cases!
- The grade from the auto grader should be pushed to your repository after submission.

- Follow the instructions for Lab 0