CSCI 2270
Data Structures and Algorithms
Recursive Sorting Algorithms, merge sort and heap sort

Elizabeth White
elizabeth.white@colorado.edu
Office hours: ECCS 112/128
Wed 9:30am-11:30am
Thurs 10am-11am

Administrivia

Colloquium writeup: yes, you can double space, but don’t count the references or cover page as one of the 6 pages.

You may also use the video of a talk *from this semester* on Vimeo, if it’s available: http://vimeo.com/channels/cucs.

Administrivia

Read Sorting chapter, pp.388-411

Heaps: Read pp. 585-606 in the text.

Test next week: know recursion lab, string alignment, linked lists, binary trees and binary search trees, quadratic sorting algorithms, and previous material

Review sheet is up.

Last homework will be on graphs in Java

If you have a reasonably tough project you would rather do, email me a proposal by Wednesday AM and come talk to me in office hours this week; I will do this if I can grade you by interview in the last week of classes.

Course goals: data structures/algos

Recognize the relationship between how we store data and how we can process it

Containers: ways to store data

    (sorted or unsorted arrays, lists, trees, graphs, etc.)

    Each of these is a different data structure

Different containers may take more or less time to perform tasks

    (search of unsorted vs. sorted array, for instance)

This is the fundamental connection between data structures and the algorithms that work on them.

Course goals: memory

Understand how the computer represents our data in memory

- local variables in functions,
- this and other pointers,
- constructors, destructors,
- new and delete, heap vs. stack,
- deep and shallow copies,
- reference parameters and return types,
- const/non-const functions, parameters, and return types

Memory bugs are very tricky to find or figure out, but if you know how these things work, you can often figure out when common things are going wrong in your code.

Course goal: mental compiler

Code analysis:

Recognize which functions are getting called

Tracing code through multiple functions by hand

Tracing code with gdb

Tracing a recursive function until it hits a base case

Counting operations to estimate the big-O run time of an algorithm on a given data structure

Being able to recognize when you could do a task in fewer operations or using less memory than the current code requires

Course goal: software development

Understand why you need good coding habits:

- Write the simplest functions you can
- Test functions right away
- Reuse code whenever possible
- Split code into public and private variables/functions
- Make code work for many different data types

Course goals

Almost none of this material is specific to C++ only

But C++ exposes you to the details of all of it very directly

Understanding it in C++ means you can pick up new languages more quickly

Faster sorting algorithms

Merge sort

Heap sort

Quick sort