Introduction to Computer Science II
CS171

Ed Goetze

Emory University

January 18, 2006
1 CS171: Introduction & Preliminaries

2 Algorithms
   • Definition
   • Examples
From the Course Catalog:
A continuation of CS170. Emphasis is on the use and implementation of data structures, introductory algorithm analysis, and object oriented design and programming with Java. The course will also introduce the basics of procedural programming with C.

Primarily Algorithms & Data Structures
- Algorithm Analysis
- Explore Fundamental & Common Algorithms
- Understand Relationship between Data Structures & Algorithms
- Explore Fundamental Data Structures, Abstract Data Types
CS171 Prerequisites

- **Formal Prerequisites**
  - CS170,
  - AP credit, or
  - An equivalent introduction to Java

- **Assumed Knowledge/Skills**
  - Ability to write & compile basic Java programs
  - Familiarity with Object Oriented Programming (Polymorphism, Inheritance, etc.)
  - Familiarity with Unix
Edward Goetze
MSc W413
404.727.5125
goetze@mathcs.emory.edu
http://www.mathcs.emory.edu/~goetze
Hours: MWF 1:00p – 2:30p & by appointment
Texts

Required Text


Additional Text

A *C* programming reference is highly recommended, e.g., *The C Programming Language*, by Kernighan & Ritchie.
Grading

1st Exam  15%
2nd Exam  15%
Final Exam  20%
Homework  50%
Exams

1st Exam  Monday, February 28th (tentative)
2nd Exam  Friday, April 14th (tentative)
Final Exam  Monday, May 8th, 4:30p–7:00p

Note: The final exam date is not negotiable. Plan accordingly!
# Homework

## Written Assignments
- Conceptual problems (e.g., from the book)
- Based on current reading & lectures.
- No programming involved.
- Similar to test questions.

## Programming Assignments
- Writing or Modifying Java or C
- Use department computer accounts
- Details shortly
- Policy on Computer Assignments

Approximately 15 assignments – about one due per week
Course Policies

- Statement of Policy on Computer Assignments (http://www.mathcs.emory.edu/SPCA) of the Department of Mathematics and Computer Science
- Emory University Honor Code
- Just like CS170

In particular, this means you should take care to protect the confidentiality of your homework files. Apparent honor code violations will be referred to the Emory Honor Council. An automated system may be used to help detect plagiarism.
Students will be graded partially on the basis of their programming assignments. These programming assignments are to be treated as examinations, and are expected to be your individual work. While discussions with other students in the course may be permitted or encouraged by your instructor, you should write your program yourself. The mathlab representatives are available to explain error messages, discuss briefly technical details with which you may not be familiar, and give short suggestions as to how you might detect logic errors. The reps should not, however, be asked to write part or all of your program. Your instructor (and any teaching assistants assigned to the course) will be glad to help you to the extent that he or she feels reasonable.

Submissions based on other students solutions in prior offerings of the course specifically violate these guidelines, as do submissions prepared with the help of an outside "tutor".

You should take precautions to protect the confidentiality of your work: preserve the secrecy of your password, do not make files or directories sharable, pick up your printouts promptly and dispose of printouts where they will not tempt other students. All work should be done either in your "priv" directory or in the class hand in directory of your University account.

All submissions should include a comment statement near the top of the program of the form:

**THIS CODE IS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING**

**A TUTOR OR CODE WRITTEN BY OTHER STUDENTS - YOUR NAME**

Cases of apparent plagiarism or collusion will be referred to the Honor Council.
Course Outline

**Ch 1,2: Introduction & Principles of Algorithm Analysis**
- Big Oh Notation
- Basic Examples
- Union-Find Example

**Ch 3: Elementary Data Structures**
- Arrays
- Linked Lists
- Stacks
- Queues
- Strings
Course Outline (continued)

Intro to C
- Pointers
- Memory Organization

Ch 4: Abstract Data Types

Ch 5: Recursion, Trees, Dynamic Programming

Ch 6-10: Sorting, Queues, Heaps, etc.
Learn to Program **Efficiently**

In CS170, learned the basics (Unix, Java, programming). Here, want to begin learning Efficient Programming. This means studying Algorithms.
### Study of Algorithms

#### Algorithm Analysis
- What’s an Algorithm?
- What’s an **Efficient** Algorithm?
- How can we methodically compare algorithms?

#### Explore Fundamental & Common Algorithms
- Basics of Effective Programming
- Illustrate Analytic Techniques
- Prerequisite for Complex Algorithms
- Leads to sound Algorithm Design
What’s an Algorithm

Informal Definition (Cormen, et. al.)
Any well-defined computational procedure that takes some set of values as input and produces some set of values as output.

Really Informal Definition:
A problem solving method on a computer.
What’s an Algorithm (continued)

Really, Really Informal Definition

Problem Solving Strategy (What’s a computer, anyway?)
Introduction to Computer Science II

Ed Goetze

CS171: Introduction & Preliminaries

Algorithms

Definition

Examples

Classical Problem

Algorithm Dependencies

- Computational Technology: What tool is used to execute the algorithm?
- Data Structure: How do you store the values (input, output, intermediate) used in the Algorithm?
- Computational Processing: What do you do with the input values to produce the desired output?

Recording Music

Devise a strategy to allow a musical performance to be reliably recorded & replayed.
Available Technology  Written (musical) language
Data Structure   Sheet Music
Computational Processing  Read sheet music, play music.
Comments  Grave technological limitations.
Recording Music: Solution #2 – Musicbox

Available Technology  Mechanical Engineering
Data Structure  Metal Cylinders
Computational Processing  Studs on metal cylinders strike musical “keys”
Comments  Technological advance results/cause by/cause
Improved Algorithm
Recording Music: Solution #3 – Player Piano

Available Technology  Mechanical Engineering
Data Structure  Paper Cylinders
Computational Processing  Mechanical sensors detect holes in paper, resulting in piano key strike
Comments  Similar technology, better(?) solution.
Recording Music: Solution #4 – Edison Phonograph

Available Technology Mechanical Engineering
Data Structure Wax Cylinders
Computational Processing Grooves in wax detected with mechanical sensor and mechanically amplified.
Comments Similar technology, better-er(?) solution.
Available Technology  Electrical & Mechanical Engineering

Data Structure  Vinyl Records

Computational Processing  Mechanical to electrical to acoustic signal translation.

Comments  Technological advance results/caused by/causes Improved Algorithm
Recording Music: Solution #6 – Tape Player

Available Technology  Electrical & Mechanical Engineering
Data Structure    Magnetic Tape
Computational Processing  Magnetic to electrical to acoustic signal translation

Comments  Improvements: data structure more compact, tougher(?).
Recording Music: Solution #7 – CD Player

Available Technology  Digital Computing

Data Structure  CD

Computational Processing  Digital to Acoustic Signal translation

Comments  Technological advance results/caused by/causes Improved Algorithm
Recording Music: Solution #8 – Compressed Digital Audio Formats

Available Technology  Digital Computing
Data Structure  Compressed Digital Audio Formats
Computational Processing  Digital to Acoustic Signal translation

Comments  Improvement due to (significantly) refined algorithm.
Algorithms always dependent on underlying computational technology

Algorithms rely on both data structure & computational procedure

For (relatively) fixed level of technology, algorithm refinement can profound impact on overall solution.
Data Structures

Studying of Algorithms goes hand-in-hand with study of Data Structures.

Data Structures

- Definitions & Concepts
- Explore the Fundamental, Basic, and Important Data Structures
- Study interplay between Algorithms & Data Structures.
Example (Searching Problem)

Given a set of numbers \( \{a_1, a_2, \ldots, a_n\} \) and a value \( v \) find an index \( i \) such that \( v = a_i \), or determine if no such index exists.

Solution #1: Sequential Search

Put the data into an array, \( a[n] \). Start at the beginning of the array, sequentially traverse array, comparing \( v \) to \( a[i] \).

Solution #2: Binary Search

Put the data into a binary search tree. Start at root, traverse left or right tree by comparing \( v \) to nodes.
Solution Comparison

**Different Data Structures** Array or Binary Search Tree

**Different Computation Procedure** Sequentially traverse the array or Traverse the tree.

Which is Preferred Solution?

- How do you measure "preferred"? Execution time? Another metric?
- How much data is there?
- How often will search be executed?
- How often will data be modified (elements added to or removed from list)?
- How complicated is it to implement the solution?
Another Classic Problem

Sorting Problem
Given a set of numbers \( \{a_1, a_2, \ldots, a_n\} \), reorder the set as \( \{a'_1, a'_2, \ldots, a'_n\} \) where \( a'_i \leq a'_{i+1} \).

Sorting Solutions
Selection Sort, Insertion Sort, Bubble Sort, Quicksort, Heapsort, Mergesort, Radix Sort, LSD/MSD Radix Sort, etc.

Each use different types of Data Structure and Procedures. Each has pros & cons. Context dictate preferred solution.
## Contemporary Examples

### Connectivity Problem

Given a set of nodes (e.g., computers) and links (edges) between the nodes (e.g., network connections), determine whether there exists a path between two given nodes.

### Path Optimization Problem

Given a set of nodes (e.g., computers) and links (edges) between the nodes (e.g., network connections), determine the **shortest** path between two given nodes.
Introduction to Computer Science II

Ed Goetze

CS171: Introduction & Preliminaries

Algorithms
Definition
Examples