**COMP 200: Elements of Computer Science**
**Fall 2004**
**Lecture 2: August 23, 2004**

To begin class, take everyone outside and have them sort themselves into ascending alphabetical order by first name. Next, into ascending order by physical height starting with an empty set.

## On the Board

Reading: Chapter 1 in Algorithmics

Homework: first one assigned today, due Monday

## Administrative Details

1. Check the web site (http://www.owlnet.rice.edu/~comp200).
2. Laboratory Assistants are Nathan Froyd and Gabriel Marin; office hours and contact information to be posted on the web site

## Example — using a phone book

Last class we considered the problem of obtaining a phone number for Star Pizza (the official pizza of COMP 200). We outlined four methods:

1. Linear search through the book, starting with the first entry.
2. Binary search — repeatedly divide the problem in half, until you get to the point where you have a single entry.
3. Binary search just the entries that begin with “s”. (Linear search just the entries that begin with “s”.)
4. Convert the string “Star Pizza”) to a unique integer and associate the phone number with that integer. ($27^{10}$, or almost 206 trillion possibilities — 205,891,132,094,649)

All of these are algorithms. They each have different properties, such as expected running time. We would expect the number of steps to be

1. on average, about 1/2 the number of entries in the phonebook;
2. on average, the base 2 logarithm of the number of entries in the phonebook;

3. same complexities, smaller input set (just the entries for “s”); and
4. constant time — the time to compute the number from the string and the time to find the number associated with that string.

**Expressing Algorithms**

A critical issue in the study of “algorithmics” or “computation” is the expression of algorithms. How do we write down an algorithm? What properties are needed to specify an algorithm?

Classic answer to the question, “What is an algorithm?” is “a recipe.”

Let’s look at several recipes that I make to understand some of the properties of recipes that cause people to consider “algorithms” and “recipes” as similar objects. At the same time, maybe we can discover some of the features of algorithms that must be expressible.

These recipes all have finite (even small) inputs and a concise description. The descriptions include lists of actions to be performed in order, along with a mechanism for repeating some list of actions (and for deciding when to stop repeating those actions).

The chocolate sauce recipe is the most sophisticated. To melt chocolate, it specifies a repeated series of actions, along with a test to determine if another repetition is needed! (In computer programming, we might call this a “loop” because we “loop” through the sequence of actions. A loop must have a clear “termination condition.”)

**Back to Our Class-Sorting Exercise**

Describe how you put yourself into the right position in the set sorted by first name. Discuss.

Points to raise:

- How did the initial condition factor into your actions? Mass chaos versus insertion into a sorted set
- Termination — how did you know when you were in the right place
- Relationship between your actions and those of other students
- How many times did you compare your name with that of another student? Was it different when sorting by height? Names are hidden and require an explicit question, while height is both visible and apparent.
- Did the class size factor into your behavior?
- Is an individual’s view different than describing the whole process?

When we describe algorithms, we need tools to talk about these kinds of issues.

- Cannot know everyone's name ⇒ we need an abstraction to talk about individuals. ("I took the place after her." "I asked him his name.")
    - Akin to variables in algebra
- Abstract names for individuals are distinct from abstract names for values. ("My name was later in the alphabet than her name." "We had the same first name.")
    - Akin to values in algebra, except that we may need to give them descriptive names so that we can manipulate them
- Algorithms often involve making decisions based on the data presented for input. (e.g., comparing names, deciding when you were in the right place) The description of an algorithm must include abstractions for decision making and for repeating sets of operations.
    - Akin to the "repeat" structure in the chocolate sauce recipe
- Algorithm needs to work on arbitrary input sets, so it must be parameterized in a way that works on sets of (effectively) any size.

These observations underlie the design of *programming languages*, formal languages designed to express algorithms.

**Homework 1** (Due Monday, August 30, 2004)

Write a concise algorithm for sorting the class into ascending order by first name.

- What does your algorithm do with people who have identical names?
- What does your algorithm do with an empty input set?
- Does your algorithm halt and produce the correct answer?

You may assume that every individual has a name, that all the names are in the same alphabet (so comparisons make sense), and that the number of individuals is finite.