CS1004: Intro to CS in Java, Spring 2005

Lecture #19: Algorithms, cont’d.

Janak J Parekh
janak@cs.columbia.edu

Administrivia

- HW#4 due on Tuesday
- I’m behind on my email/homework fixes… I should be caught up by Tuesday

Board examples

- Palindrome checker (see book for code)
- Print out the first # Fibonacci numbers
- Search for a number (or an item in general) in a list
- Find the largest number in a list
- Sort numbers
- Other examples in the books
Strategy

- General process:
  - Identify the main classes/data involved
  - Try to do one or two steps by hand
  - Generalize and write out the algorithm
- Let’s begin, on the board
- We’ll then talk about how to characterize the resulting algorithms we get

Algorithm correctness & efficiency

- Define desirable characteristics in an algorithm:
  - Correctness
    - Does the algorithm solve the problem it is designed for?
    - Does the algorithm solve the problem correctly?
  - Ease of understanding
    - How easy is it to understand or alter an algorithm?
    - Important for program maintenance

Attributes of Algorithms (continued)

- Elegance
  - How clever or sophisticated is an algorithm?
  - Sometimes elegance and ease of understanding work at cross-purposes
- Efficiency
  - How much time and/or space does an algorithm require when executed?
  - Perhaps the most important desirable attribute
Measuring Efficiency

- Analysis of algorithms
- Study of the efficiency of various algorithms
- Efficiency measured as function relating size of input to time or space used
- For one input size, best case, worst case, and average case behavior must be considered
- The $\Theta$ notation captures the order of magnitude of the efficiency function
  - $\Theta$ ("big-Theta") vs. $O$ ("big-Oh") notation

Order of Magnitude: Order $n$

- As $n$ grows large, order of magnitude dominates running time, minimizing effect of coefficients and lower-order terms
- All functions that have a linear shape are considered equivalent
- Order of magnitude $n$
  - Written $\Theta(n)$
  - Functions vary as a constant times $n$

Sequential Search, analyzed

- Comparison of the NAME being searched for against a name in the list
  - Central unit of work
- For lists with $n$ entries:
  - Best case
    - NAME is the first name in the list, 1 comparison
    - $O(1)$
  - Worst case
    - NAME is the last name in the list, or not in list
    - $n$ comparisons, or $\Theta(n)$
  - Average case
    - Roughly $n/2$ comparisons, or $\Theta(n)$
Sequential Search (continued)

- Space efficiency
  - Uses essentially no more memory storage than original input requires
  - Very space-efficient
- But… is there a faster way to search through a list?

Binary Search

- Given ordered data,
  - Search for NAME by comparing to middle element
  - If not a match, restrict search to either lower or upper half only
  - Each pass eliminates half the data
- Efficiency
  - Best case
    - 1 comparison: Θ(1)

A Comparison of n and lg n (S/G, pg. 109)
Sorting

- What if we want to sort the numbers in a list?
- There are number of algorithms; book describes selection sort, but we'll also go over bubble sort very quickly.
- Let's begin!

Next time

- Continue working with algorithms