Course Web Page

- Most information about the course (including the syllabus) will be posted on the course wiki:
  

- Check your e-mail frequently for course announcements.

Algorithms

- Algorithm: a computational procedure that takes input and produces output.
- Algorithmic problem statement must specify the complete set of possible input instances and the desired output from each input instance.
- The distinction between a problem and an instance of a problem is fundamental.

Problem and Instances

The sorting problem is defined as follows:

**INPUT:** An array $A[1...n]$ of $n$ comparable items $\{a_1, a_2, ..., a_n\}$

**OUTPUT:** A permutation of the input array $\{a_1', a_2', ..., a_n'\}$ such that $a_1' \leq a_2' \leq ... \leq a_n'$.

An instance of the sorting problem might be an array of names, such as {Mike, Sally, Herbert, Tony, Jill}, or a list of numbers, {154, 245, 100, 987, 444}, etc.

Algorithm Correctness

- A correct algorithm is one in which every valid input instance produces the correct output.
- Proving correctness cannot be done by implementing and testing the algorithm. The correctness must be proved mathematically.

Algorithm Complexity

- Algorithm complexity is a measure of the resources an algorithm uses. The 2 resources we care about are: ??? and ???
- We express each of these quantities in terms of the input size.

Data Structures

- Covered previously:
  - Arrays
  - 2-D Arrays
  - Stacks
  - Queues
  - Lists

- Presented in 241:
  - Rooted trees
  - Adjacency lists (lists of lists)
  - Hash tables
  - Binary search trees
  - Graphs, directed and undirected
Practical Applications

- Most real-world problems require some combination of solutions to smaller problems.

Many of the algorithms we will study must be used in combination to solve more complex problems.

Eg, sorting is a fundamental subproblem for applications ranging from search engines to finding the smallest enclosing polygon of a set of points. But it is not the whole solution to most.

Hard Problems

- Problems are separated into categories according to whether or not an efficient solution can be found.

- Most of the problems presented in the book have efficient solutions; those that do not are called ???.

We will spend the last part of the semester studying NP-complete problems.

Lecture topics

- Introduction and Mathematical Fundamentals - Chapters 1, 2, 3, and 4
- Probabilistic Analysis and Randomized Algorithms – Chapter 5
- Sorting - Chapters 6, 7, and 8
- Selection - Chapter 9
- Hash Tables - Chapter 11
- Basic Graph Algorithms - Chapters 22, 23, and 24
- Data Structure for Disjoint Sets - Chapter 21
- Dynamic Programming, Greedy Algorithms - Chapters 15 & 16
- Aggregate Analysis - Chapter 17
- Geometric Algorithms - Chapter 33
- NP-Completeness - Chapters 34, 35

Book Pseudocode Conventions

In ed. 2 of our textbook, assignment is indicated by a left-pointing arrow and comparison is a single =

In ed. 3 assignment is = and comparison is ==

Arrays are usually assumed to be numbered starting at 1, not 0. A[j] indicates element j of array A and A[1...j] indicates a subarray of elements 1 through j in A

Indentation is the only indication of block structure.

Book Pseudocode Conventions (cont)

In ed. 2, loops end with a “do” and if statements end with “then” on the following line.

In ed. 3, there are no “do”s or “then”s.

In both books:
- “and”s and “or”s are “short-circuiting”
- “return” statements immediately exit the current procedure and may return more than one value
- variables are never declared unless passed into a procedure

See rest of pseudocode conventions in Section 2.1.

End of Lecture 1