LECTURE 1
Analysis of Algorithms
• Course information
• What are algorithms?
• Why study them?

Sofya Raskhodnikova and Adam Smith
Course information

1. Staff
2. Prerequisites
3. Lectures
4. Handouts
5. Textbook
6. Course website
7. Homework
8. Grading policy
9. Collaboration policy
Course Objectives

• classical algorithms
• analysis of algorithms
• standard design techniques
Etymology of “Algorithm”

Abu Abdullah Muhammad ibn Musa al-Khwarizmi (c. 780 -- 850 AD)

- Persian astronomer and mathematician
- lived in Baghdad, father of algebra
- “On calculating with hindu numerals”
  a treatise in Arabic, 825
- “Agoritmi de numero Indorum”
  translation into Latin, 12th century
- author’s name, mistaken for a plural noun, came to mean “calculation methods”
Algorithm Design and Analysis

*Theoretical study of how to solve computational problems*

- sorting a list of numbers
- finding a shortest route on a map
- scheduling when to work on homework
Algorithms

• Definition: Finite set of unambiguous instructions for solving a problem.
  – An algorithm is **correct** if on all legitimate inputs, it outputs the right answer in a finite amount of time

• Can be expressed as
  – pseudocode
  – flow charts
  – text in a natural language (e.g. English)
  – computer code
Data Structures

- **Data structures** are ways to store information for which there are **algorithms** for performing particular operations (retrieving/manipulating information), e.g.
  - linked lists
  - hash tables
  - arrays
  - trees
  - heaps
Why study algorithms?

• a **language** for talking about program behavior
• standard set of algorithms and design techniques
• feasibility (what can and cannot be done)
  – halting problem, NP-completeness
• analyzing correctness and resource usage
• successful companies (Google, Mapquest, Akamai)
• computation is fundamental to understanding the world
  – cells, brains, social networks, physical systems all can be viewed as computational devices
• **IT IS FUN!!!**
Performance isn’t everything

• Typical goal: Find most space- and time-efficient algorithm for given problem.

• What else is important?

  – modularity
  – correctness
  – maintainability
  – functionality
  – robustness
  – user-friendliness
  – programmer time
  – simplicity
  – extensibility
  – reliability