Lecture 1
Analysis of Algorithms
- Course information
- What are algorithms?
- Why study them?
Course information

1. Staff
2. Prerequisites
3. Lectures
4. Handouts
5. Textbook
6. Course website
7. Homework
8. Grading policy
9. Collaboration policy
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
• answering web search queries

(Generally: precisely defined set of inputs and, for each input, acceptable outputs)
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
Course Objectives

• classical algorithms and data structures
• analysis of algorithms
• standard design techniques
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
Performance isn’t everything

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

• Even performance has many facets:
  – type of memory access
  – cache usage
  – network usage
  – parallelism

• This course: simple models, general skills
The problem of sorting

**Input:** sequence \( a_1, a_2, \ldots, a_n \) of numbers.

**Output:** permutation \( a'_1, a'_2, \ldots, a'_n \) such that \( a'_1 \leq a'_2 \leq \cdots \leq a'_n \).

**Example:**

**Input:** 8 2 4 9 3 6

**Output:** 2 3 4 6 8 9
Insertion Sort

```
INSERTION-SORT (A, n) ▷ A[1 . . n]
for j ← 2 to n
    do key ← A[j]
        i ← j – 1
    while i > 0 and A[i] > key
    do A[i+1] ← A[i]
        i ← i – 1
    A[i+1] = key
```

"pseudocode"
Insertion Sort

```
INSERTION-SORT (A, n)  ▷ A[1 . . n]
for  j ← 2 to n
    do  key ← A[j]
        i ← j − 1
    while i > 0 and A[i] > key
        do A[i+1] ← A[i]
        i ← i − 1
    A[i+1] = key
```

A:  

```
1  i   j  n
```

S. Raskhdonikova and A. Smith. Based on slides by E. Demaine and C. Leiserson