Algorithms

-Design and Analysis of Algorithms-

컴퓨터 과학과 :김 윤호

E-mail: yhkim@smu.ac.kr

Office: G515

02-2287-5312

Course Description

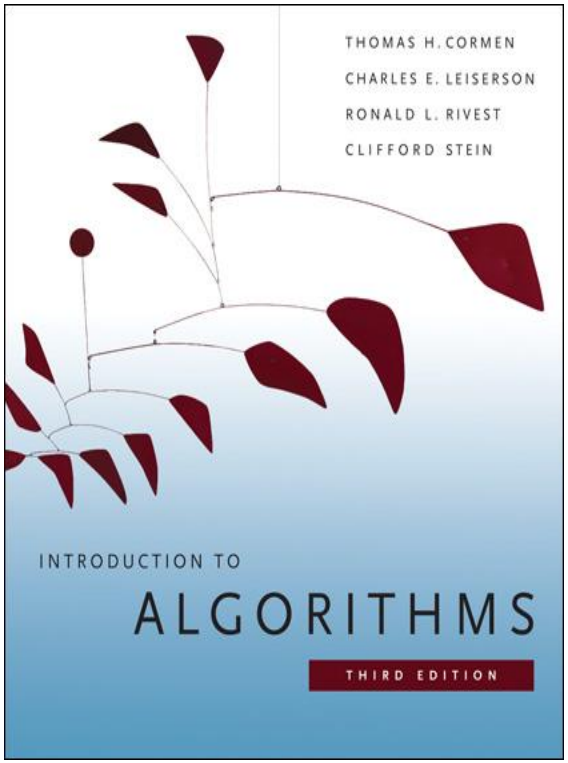
- Design and Analysis of Algorithms is the most important basic course in computer science and engineering curriculum.
- Study of Algorithm
  - Designing algorithms
  - Expressing algorithms
  - Algorithm Validation
  - Algorithm Analysis
  - Alternative techniques
- The course is not limited to any programming language.

Course Objectives

- The objective of this course is to build a solid foundation of the most important fundamental subject in computer science.
- Algorithms are extensively used in *Databases, networks, artificial intelligence, bioinformatics, pervasive and mobile computing, robotics, security, architecture, all engineering and science disciplines, finance, management, music, biology and indeed in everyday life.*
- Creative thinking is essential to algorithm design.
- Algorithm analysis and verification demands sound mathematical acumen and programming skills.

Textbook

- Introduction to Algorithms, Third Edition
  by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein , (CLRS) MIT Press



Grading Policy

- Prerequisite:
  - Discrete Math.
  - Data Structure
  - Programming
- Assignment
  - Handwriting assignments & Programming assignments
- Grade components & relative weights:
  - Handwriting & Programming assignments: 20%
  - mid-term exam: 35%
  - final exam: 35%
  - Attendance: 10%

What are algorithms?

- An algorithm is a precise and unambiguous specification of a sequence of steps that can be carried out to solve a given problem or to achieve a given condition.
- An algorithm is a computational procedure to solve a well defined computational problem.
- An algorithm accepts some value or set of values as input and produces a value or set of values as output.
- An algorithm transforms the input to the output.
- Algorithms are closely related to the nature of the data structure of the input and output values

Algorithm

- An algorithm is designed to solve a given problem
- An algorithm does not take into account the intricacies and limitations of any programming language.
- An algorithm should be unambiguous
  - it should have precise steps
- An algorithm has three main components:
  - Input, the algorithm itself , and output.
- An algorithm will be implemented using a programming language
- An algorithm designer is like an architect while programmers are like masons, carpenters, plumbers etc.

Algorithm Example

- An algorithm to sort a sequence of natural numbers into nondecreasing order
- An algorithm to find a shortest path from one node to another in a graph
- An algorithm to find the best scheduling of events to resources
- An algorithm to recognize a substring in a string of letters
- …..

Where do we use Algorithms?

- Everyday Life
- Computer Science
- Biology
- Economics
- Marketing
- Running a Business
- Music
- Games
- Others …

Algorithm Vs. Program Code

- An algorithm
  - is an abstraction of an actual program
  - is a computational procedure that terminates
  - is composed of a finite set of steps
- Program code
  - A program is an expression of an algorithm in a programming language.
  - A program code conforms to the dictates of policies and limitations of programming language

Presenting Algorithm

- Description : The algorithm will be described in English, with the help of one or more examples
- Specification : The algorithm will be presented as pseudo code (We don't use any programming language)
- Validation : The algorithm will be proved to be correct for all problem cases
- Analysis: The running time or space complexity of the algorithm will be evaluated

The algorithms we design should be

- Simple
  - Unambiguous
- Feasible
  - Should be implementable using a programming language and executable on a computer.
- Cost effective
  - CPU time
  - Memory used
  - Communication
  - Energy

Design and Analysis of Algorithms

- How to design algorithms?
  - Brute force (Naïve approach)
  - Divide and conquer
  - Dynamic programming
  - Greedy Algorithm
  - …
- How to analyze algorithm efficiency
  - Time
  - Space
  - Energy

Algorithm Analysis

- To analyze an algorithm means:
  - developing a formula for predicting *how fast* an algorithm is, based on the size of the input (time complexity), and/or
  - developing a formula for predicting *how much memory* an algorithm requires, based on the size of the input (space complexity)
- Usually time analysis is our biggest concern
  - Most algorithms require a fixed amount of space
  - But space analysis can be important for embedded system.

Algorithm Efficiency

- Consider two sort algorithms
  - **Insertion sort**
    - takes $c_1 n^2$ to sort n items
    - where c_1 is a constant that does not depends on n
    - it takes time roughly proportional to n^2
  - **Merge Sort**
    - takes $c_2 n \log n$ to sort n items
    - where c_2 is also a constant that does not depends on n
    - it takes time roughly proportional to $n \log n$
  - Insertion sort usually has a smaller constant factor than merge sort so that, $c_1 < c_2$

Algorithm Efficiency

- Consider now:
  - A **faster** computer **A** running **insertion sort** against
  - A **slower** computer **B** running **merge sort**
  - Both must sort an array of **one million** numbers
- Suppose
  - Computer **A** execute **one billion** (10^9) instructions per sec
  - Computer **B** execute **ten million** (10^7) instructions per sec
  - So computer **A** is **100** times faster than computer **B**
- Assume that
  - $c_1 = 2$ and $c_2 = 50$

Algorithm Efficiency

- To sort one million numbers
  - Computer A takes

$$\frac{2 \cdot (10^6)^2 \text{ instructions}}{10^9 \text{ instructions/second}} = 2000 \text{ seconds}$$

  - Computer B takes

$$\frac{50 \cdot 10^6 \cdot \log(10^6) \text{ instructions}}{10^7 \text{ instructions/second}} \approx 100 \text{ seconds}$$

- Computer B runs 20 times faster than Computer A
- **For ten million numbers**
  - **Insertion sort takes ≈ 2.3 days**
  - **Merge sort takes ≈ 20 minutes**

Algorithm Efficiency

n	**Insertion sort *O(n^2)***	**Merge sort *O(nlogn)***
4	16	8
8	64	24
16	256	64
32	1024	160
64	4096	384
128	16,394	896
256	65,536	2048
512	262,144	4608
1024	1,048,576	10,240
1,048,576	~1,000,000,000,000	20,971,5201024

Algorithm Efficiency

- What does this mean for computer science?
- It means that using efficient algorithms can be even more important than building faster computers: more efficient thinking beats more efficient hardware!
- And this means that algorithms are definitely worth studying.