CS483 Lecture 01

Jyh-Ming Lien

Course Topics

- Jan 22 Introduction
- Jan 29 Algorithms with numbers
- Feb 05 Divide-n-conquer algorithms
- Feb 12 Graphs
- Feb 19 Paths in graphs
- Feb 26 Greedy algorithms
- Mar 26 Dynamic programming
- Apr 09 Linear programming
- Apr 23 NP-completeness

- **See details and updates on the course webpage**

Course Info

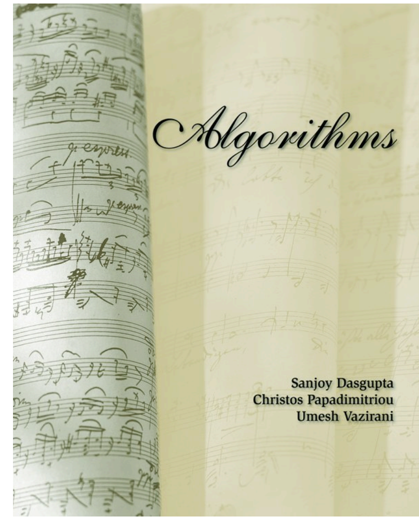
- course webpage:
  - from the syllabus on http://cs.gmu.edu/
  - http://cs.gmu.edu/~jmlien/teaching/09_spring_cs483/
- Information you will find
  - course syllabus
  - time table
  - problem sets
  - pdf copies of the lectures
  - office hours
    - (mine) monday 6-7pm, (TA) Tue. 4-6pm

Prerequisite

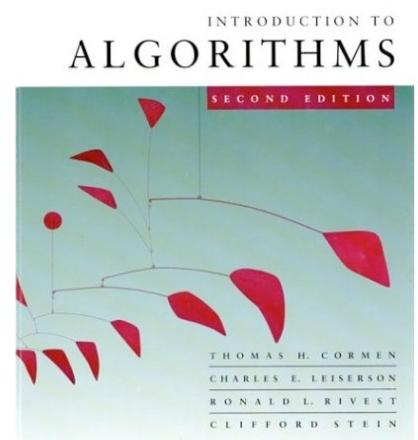
- Data structures and algorithms (CS 310)
- Formal methods and models (CS 330)
- Calculus (MATH 113, 114, 213)
- Discrete math (MATH 125)

Textbook

- **Algorithms**, by Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani, McGraw-Hill, 2006



- I also recommend you read the following book: **Introduction to Algorithms** by T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, The McGraw-Hill Companies, 2nd Edition (2001)



Grades

- Quizzes (every week) 30%
- Midterm Exam (March 18) 35%
- Final Exam (May 6) 35%
- Make-up tests will **NOT** be given for missed examinations

Other Important Info

- **Email**
  - make sure your gmu mail is activated
  - send only from your gmu account; mails might be filtered if you send from other accounts
  - when you send emails, put [483] in your subject header
- **Discussion Board**
  - it's a google group, add yourself

OK, lets start

Rabbits

- you are given a newly-born pair of rabbits, one male, one female
- Rabbits are able to mate at the age of one month so that at the end of its second month a female can produce another pair of rabbits
- rabbits never die
- the female always produces one new pair

Rabbits

- How many pairs will there be in one year?
  - End of month 1
  - End of month 2
  - End of month 3
  - End of month 4
  - End of month 12

Fibonacci Number

$$fib(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ fib(n-1) + fib(n-2) & \text{if } n > 1 \end{cases}$$

- Question: what is *fib*(200), what about *fib*(200000000000000)?

Our First Algorithm

procedure $\text{FIB}(n)$
if $n = 0$
  then return (0)
if $n = 1$
  then return (1)
return $(\text{FIB}(n-1) + \text{FIB}(n-2))$

- Questions we should ask
  - Is the algorithm correct?
  - What is the running time of our algorithm?
  - Can we do better

Analyze Algorithm

- Is it correct?
- How fast does it run?
  - we let the run time of fib(n) be T(n)
    - T(n)=
    - T(200)=
  - assume you have a Intel Pentium 4 CPU (3GHz)
    - It takes

Analyze Algorithm

- Can Moose's law, which predicts that CPU get 1.6 times faster each year,solve our problem?

Analyze Algorithm

- Can we do better?

Analyze Algorithm

- When we design an algorithm, we should ask ourselves
  - Is the algorithm correct?
  - How efficient is the algorithm?
    - Time efficiency
    - Space efficiency
  - Can we do better?
- How do we measure the efficiency?
  - empirical analysis
  - theoretical analysis

Summary

- General ideas of design of algorithms
- Analysis of algorithms: experimental and theoretical
- Asymptotic notations: O (upper bound), (lower bound), (tight bound)

Warning

- Make sure you have the mathematics or CS prerequisites
- You must make arrangements to come to GMU to take the exams on-site
- You need to spend a minimum of 9~12 hours a week outside of class reading the material and doing practice problem sets

Suggestions

- **Don’t fall behind** - maintain a steady effort
- **Take the homework (quizzes, practice problems) seriously** - these are the only ways to exercise for the exams
- **Make use of office hours** - we are here to help, but it will be more helpful if you can think about the questions in advance
- **Read the materials before the class** and ask during the class- this prepares you for the quizzes
- **Form study groups** - things become easier if there is joint force

Course Outcome

- An understanding of classical problems in Computer Science
- An understanding of classical algorithm design and analysis strategies
- An ability to analyze the computability of a problem
- Be able to design and analyze new algorithms to solve a computational problem

Practice Problem Set

- Go to course webpage