Who should take this course?

Anyone who is either

- a computer science/engineering student
- interested in getting to know the most fundamental area of Computer Science

or

- forced to take it because it’s required and/or all other courses were filled up

The catches are

- data structures (CSE250 or equivalent)
- some formal calculus/analysis course
- and a course which requires formal proofs (Discrete Math.)

... not crucial if you’re motivated enough, though.
Who should teach this course?

- Hmm ...

Teaching Staff

Instructor:
- **Hung Q. Ngo**

Teaching Assistants:
- **Thanh-Nhan Nguyen**: recitation section [temporarily assigned]
  - **R1** (R1 Mon 8:00-8:50 214 Norton)
- **Yang Wang**: recitation section
  - **R2** (Wed 9:00-9:50 103 Talbert)
When/Where to Talk to Me?

Algorithm (Your First Algorithm)

1: Course blog http://ubcse531.wordpress.com/
2: email (hungngo@cse.buffalo.edu)
3: office hours - 238 Bell Hall, 9:30-10:30 Tue & Thu
4: sneak in whenever the door is opened
5: goto 1

What is the course about?

- Have fun learning!
- Grasp a few essential ideas of algorithm analysis and design
  - asymptotic notations and analysis
  - fundamental algorithm design methods: divide and conquer, greedy, dynamic programming, linear programming, network flow
  - the notions of NP-Completeness, approximation algorithms, and possibly randomized algorithms
- Gain substantial problem solving skills in designing algorithms and in solving discrete mathematics problems
Course Materials

Required textbook

Online Materials
http://www.cse.buffalo.edu/hungngo/classes/2007/Fall-531

Recommended references
Knuth’s Classic three volume *The Art of Computer Programming*.

Work Load

- Heavy! So, start early!
- Approx. 30 pages of **dense** reading per week
- 6 written homework assignments (to be done individually)
- 1 midterm exam (in class, closed book/notes)
- 1 final exam (in class, closed book/notes)
Grading Policy

- 6 assignments: 5% each
- 1 midterm exam: 30%
- 1 final exam: 40%

Note:
- Assignments are due at the beginning of the lecture on the due date
  - 1 day late (24 hours): 20% (of max score) reduction
  - each extra date: 40% more
- Incomplete grade and make-up exams: not given, except in provably extraordinary circumstances

Academic Honesty

Absolutely no tolerance on plagiarism

- Please do the assignments individually on your own. Do not discuss with classmates.
- 0 on the particular assignment/exam for first attempt
- Fail the course on the second plus report to department and school
- Consult the University Code of Conduct for details
- In summary, I will take plagiarism very seriously
### About partial credits

**Algorithm (Your second algorithm)**

*Input:* your write-up of a solution to a homework/exam problem

1. **if** the write-up contains non-sense (e.g., returned by Google!) **then**
2. You’ll get zero points
3. **else if** you admit you do not know how to solve the problem **then**
4. You’ll get $1/4$ of the total credit
5. **else**
6. Proceed with normal grading
7. **end if**

The point is ... **to reward intellectual honesty!**

---

### No Lame Excuses, Please!

- I have to go home early, please let me take the final on Dec 01.
- I had a fight with my girlfriend
- I’ve studied hard, I understood the material very well, ... but I got a C. Please consider giving me A-
- I think I deserve a better score, please give me some work to do next semester to improve the score
How to do well in this course?

- Ask questions in class
  The only stupid question is the question you don’t ask
- Suggestions are always welcome
- Attend lectures
- Do homework/reading assignments early!
- At least, skim through reading assignments before lectures
- Print out lecture notes before attending lectures

We, the TAs and I, are here to help. Don’t hesitate to ask.

A few motivating examples

Example (Fibonacci numbers)
Write an algorithm to calculate the $n$th Fibonacci number, given $n$

\[
F_0 = 0 \\
F_1 = 1 \\
F_n = F_{n-1} + F_{n-2}, \quad n \geq 2
\]

Example (Primality testing)
Given a natural number $n$, return
- YES if it is a prime number
- NO otherwise

(Agrawal, Kayal, Saxena – 2002)
A few motivating examples

Example (Shortest Path)
Devise an algorithm to find a shortest path from a source (e.g. your computer) to a destination (e.g. www.nfl.com) in the Internet

Example (Steiner Tree)
Given a set of cities, find an algorithm to assist in building a highway system connecting all these cities, so that the total length of highways is minimized.

Aha - Algorithms!

Algorithm (FibA)

Input: non-negative integer $n$.

1: if $n \leq 1$ then
2: return $n$
3: else
4: return $(FibA(n - 1) + FibA(n - 2))$
5: end if
Algorithm (FibB)

Input: non-negative integer \( n \).
1: if \( n \leq 1 \) then
2: return \( n \);
3: else
4: \( a \leftarrow 0; b \leftarrow 1; \)
5: for \( i \) from 1 to \( n - 1 \) do
6: \( \text{temp} \leftarrow a; a \leftarrow b; \)
7: \( b \leftarrow \text{temp} + a; \)
8: end for
9: return \( b \);
10: end if

Question

What are the pros and cons of FibA and FibB?

Analyzing Algorithms

- mean of “roughly predicting” the resources required
- Resources:
  - How fast: time complexity
  - Memory requirement: space complexity
  - Others: communication bandwidth, hardware costs, ...

Need a specific machine model: Turing machine, RAM, parallel computers, quantum computers, DNA computers, ...

- We’re mostly concerned with time complexity: a rough estimate of running time wrt the input size
- We will be very informal until NP-completeness is discussed
Approaches for Designing Algorithms

- Ask someone
- Hack around 'til it works
- Brute force
- Incremental
- Divide and conquer
- Greedy
- Dynamic programming
- Formulate the problem as something we already known how to solve (e.g, network flow, linear/non-linear programming, etc.)
- A stroke of genius
- Give up

Note: “programming” is not programming

Lastly

- Hope to learn as much from you as you’d learn from me
- Enjoy the ride!