.

Course Outline

In this course, you will learn about design and analysis of algorithms with applications in many areas of computer science, including sorting and searching, string processing and editing, cryptography, bio-informatics, scheduling, artificial intelligence, and networks. More importantly, you will learn to how to design your own algorithms for new problems. The course is organized roughly as follows.

1. **Motivation; Introduction and Mathematical Foundations** (1.5 weeks): motivating examples of algorithm design and analysis, notation for measuring and comparing growth rates of functions, summations, examples of recurrence relations and their solution.

2. **Fundamental Algorithms** (4.5 weeks): We will consider deterministic and randomized algorithms from the following areas. Throughout, we will focus on learning design techniques, establishing correctness of algorithms and applying the analysis tools covered in the initial

part of the course. We will look at several applications to real-world problems. Although the underlying computer model will usually be the standard (sequential) computer, we will also consider some parallel algorithms.

- sorting and searching (1 week): sorting (quicksort) and selection, lower bounds
- graph theory (1 week): graph traversal algorithms and applications, minimum spanning tree, shortest paths
- strings (1 week): pattern matching, text compression
- algebra (.5 week): matrix and polynomial multiplication
- number theory (1 week): greatest common divisor, primality testing, cryptographic protocols

3. **NP-completeness** (1 week): a study of problems for which there are no known efficient algorithms.

Homeworks, and Exams

Graded course work will include five homeworks, two in-class midterms and a final exam. The grading scheme is:

- Homeworks — 20 %
- Friday May 23rd Midterm — 20 %
- Wednesday June 11th Midterm — 20 %
- Final — 40%

The instructor reserves the right to modify this grading scheme. All assignments are due at 6PM on the specified day. TA's will take 20% off for a day late and will not accept homeworks after this. **Please do not turn in homeworks in my mailbox or office or via email.**

The best way to learn the material is to work on lots of problems. There will be several opportunities to do this in class, in the tutorials, and in the homeworks. In mastering the material, you will learn how to think logically, how

to explain your ideas convincingly and how to hone your problem-solving skills. You may be surprised at how often the things you learn in this class will help you in your other courses too. Partial answers to the problems will be given partial grades, so always submit your best attempt at solving a problem.

Solving the homework problems will be not unlike solving puzzles – something that I think most people in CS have enjoyed since childhood. It can be very rewarding when you succeed, but tremendously frustrating when you are stuck. Start working on problems early. Sometimes when you are stuck, sleeping on it is the best strategy, and that's not an option if you wait till the last evening to get started. Use all the resources available to you when you reach the point of frustration – the instructor, TA's, class mailing list, and most importantly, your classmates. Group discussion of the homeworks is strongly encouraged, but **all write-ups should be done on completely on your own**.

Academic Conduct

I expect no problems with cheating but if it occurs it will be severely dealt with (see university web page for academic misconduct policies).

Class Participation

Please **ask questions** in class! You may find the mathematical tools used in the course intimidating at first. I will make every effort to help students understand (and even appreciate) the math. If you are confused, probably many others are too. The best way to clear up confusion is to ask questions. Others will appreciate it if you take the plunge.

Logistical Information

- **Instructor**

  Nando de Freitas, Room 183 CICSR, office hours: Monday 3:00-4:00. email: nando@cs.ubc.ca.

- **Tutorial Leaders**

  – James Cook. office hours: TBA. email: jcook@cs.ubc.ca.

  – Jihong Ren. email: jihong@cs.ubc.ca.

  – Jonathan Backer. email: backer@cs.ubc.ca.

- **TAs**

  To be announced.

- **Web page**

  The class web page location is:

  `http://www.cs.ubc.ca/~nando/320-2003/`

  The page will contain brief lecture notes, handouts, copies of all homework sets, solutions, and other important information.

- **Newsgroup**

  `ubc.courses.cpsc.320`

- **Recommended Texts**

  Since the on-line lecture notes will be fairly self-contained, it is not absolutely necessary to buy the text. However, the following text is recommended:

  Introduction to Algorithms by T. H. Cormen, C. E. Leiserson, and R. L. Rivest, McGraw Hill.

- **Prerequisites**

  The prerequisites for the course are: 1 of CPSC 216, CPSC 252 and 1 of CPSC 220, EECE 320 and 1 of STAT 241, STAT 200. You should have already be familiar with O-notation, recursion, basic data structures (linked lists, heaps, hashed structures, trees and graphs), basic sorting algorithms and graph algorithms (depth first search). All of this material is also covered in the Cormen et al. text so you can review it there.

Facility with basic discrete mathematics (recursion, manipulation of probabilities, summations and series, sets, relations, and functions) is also desirable for the course.