CSC373S Algorithm Design and Analysis Lecture 1 Instructor: A. Borodin
Text: "Algorithm Design " by Jon Kleinberg and Eva Tardos

- Overall goal of course: trying to make algorithm design and analysis into a coherent field of study. We will be concentrating on discrete computation considering problems from a number of different areas such as scheduling, logic, graph theoretic problems, geometric problems, algebraic and number theoretic problems.

- Different types of computational problems

  1. functions: $f : D \rightarrow D$;
     e.g. given an integer $x \geq 2$, output $y =$ the smallest prime factor of x. (If x is prime then $y = x$.)

     Note: in CS, we think more computationally and we often assume that the inputs/outputs have been encoded as strings over some finite alphabet; i.e. $D = \Sigma^*$. For the above problem, it is important to think of the complexity as a function of the length of the (say binary or decimal) encoding of x.

  2. Search problems: For a relation $R \subseteq D \times D$, given x find y (if it exists) such that R(x,y); if no such y exists then say so.
     e.g. $FACTOR(x, y)$ iff x and y are (encodings of) integers and y is a proper factor of x.

     Given a set of jobs x, and a schedule y, $R(x, y)$ iff schedule y achieves some criteria (eg no job is scheduled after its deadline, at least z "non-conflicting" jobs are scheduled, etc.)

     Given prop. formula F and truth assignment τ, $R(x, y)$ iff τ satisfies F.

  3. Optimization problems: a search problem with an additional objective function which is being optimized.
     e.g. Given set of jobs, find a schedule that minimizes the maximum lateness; find a schedule that maximizes the number/profit of non-conflicting jobs, etc.

     For optimization problems we are often willing to (or must) sacrifice optimality for efficiency and then we are interested in obtaining a solution which is "close" to optimal whenever this is possible.

- In this course, most problems we will study are optimization problems as this allows us to explore a wide variety of general algorithmic techniques. The entire course will concern "discrete" computation say in contrast to numerical optimization.
- Common algorithmic techniques

  Most courses and texts in this area organize the material in terms of common algorithmic techniques/paradigms/meta-algorithms. (Alternatively, one could chose some basic problems (or problem areas such as scheduling problems, graph problems, geomtric problems, logic problems) and then apply a number of techniques to solving each problem.)

  1. Greedy algorithms
  2. Divide and conquer
  3. Dynamic programming (DP)
  4. Flow based algorithms
  5. Local search
  6. LP and other "mathematical programming" approaches (eg SDP)
     (a) LP relaxations of an IP
     (b) primal dual algorithms

- Additional general methods we probably wont discuss

  1. algebraic algorithms
  2. backtracking and branch and bound; brute force
  3. genetic algorithms
  4. generalizations of local search (eg simulated annealing)
  5. Multiplicative weights update

- Additional algorithmic topics

  1. Randomized algorithms: Note: this is NOT a meta-algorithm but an additional computational idea that can be utilized in conjunction with any algorithmic technique.
  2. Transforming or reducing one problem to another. We often solve a given problem P_1 by using a subroutine for a problem P_2.

     In CSC 363/365, we use this idea to show that if P_1 is a "hard problem", then P_2 must also be hard. Here in CSC373 we use such problem reductions to show how to utilize a solution for P_2 to obtain a solution to P_1.
     Note: The first question in the problem set is essentially a reduction.
  3. Approximation algorithms. Note: this is not a meta-algorithm but rather we apply the above techniques to obtain (hopefully) good approximations to an optimal solution.

The grading scheme will be based on 3 problem sets (5% each), each of which will be immediately followed by a term test (15% each), and a final exam (40%). As soon as an assignment is due (on a Wednesday) and collected, we will discuss the solutions in class and a term test will follow (on Friday). Therefore, no late asssignments will be accepted. See the course web page (www.cs.toronto.edu/~bor/373s11) for the dates of all problem sets and tests.

Office hours (SF 2303B): To be announced. Tentatively: T,R 2-3. Beyond any posted office hours, students are always welcome to make appointments and/or drop by to see if I am available. In general, I prefer speaking to people in person than via email!