ECE 406 – Algorithm Design & Analysis (Winter 2011)

Instructor: Mahesh V. Tripunitara, tripunit@uwaterloo.ca, x32864, EIT 3112
Prerequisite: ECE 250 (Data Structures) or equivalent (e.g., SE 240)
Textbook: "Algorithm Design," Kleinberg and Tardos
http://www.aw-bc.com/info/kleinberg/

An algorithm is a method that solves a problem. They are at the foundations of computing. It is important that every practitioner of computer engineering understands how algorithms are designed, and how to analyze them for correctness and efficiency. It is important also to be able to distinguish intractable problems from ones that are tractable so one does not naively seek efficient solutions when none may exist. For cases that are intractable, it is important to know how to design approximate solutions that satisfy bounds on correctness and efficiency. Industry has long recognized the critical importance of algorithms that are correct and efficient.

Topics: Models of computation, kinds of problems (e.g., decision, optimization) and relationships between them, complexity classes (e.g., L, P, NP, PSPACE), analysis techniques for P, hardness and completeness from the standpoint of complexity, correctness (soundness and completeness) of an algorithm, average-case analysis for search algorithms, average-case analysis for hash tables, trees (including B-trees and its variants) and corresponding analysis, graph algorithms, optimization problems and dynamic programming, other topics (e.g., string matching, number theoretic algorithms, computational geometry), NP-completeness, approximation algorithms.

Grading: There will 2 midterms and a final that comprise the final grade. Midterm 1: 20%, Midterm 2: 30%, Final: 50%. There will be weekly homework problems that are assigned. The homework assignments will not be graded. However, I will discuss them in class, and provide solution sketches for them. The midterms and exams will be based heavily on the lectures and homework assignments. The midterms and final will be open book, open notes, but with no electronic materials allowed.

Example Problems

1. (a) Write pseudo-code for an efficient algorithm that solves the following decision problem. Inputs: (i) a sorted array of numbers, A, (ii) a number, n, and, (iii) a number, r. Output: 'true,' if there exists a in A such that $a \in [n - r, n + r]$, and 'false,' otherwise.

(b) State a correctness property, and prove that your algorithm is correct.

(c) What are the best, average and worst-case time-complexities of your algorithm?

2. Show that $f(n)$ is bounded from above by a polynomial if and only if $\lg(f(n)) = O(\lg n)$.

3. In a hash table with m slots and n elements, in which collisions are resolved by chaining, show that a search takes $\Theta(1 + n/m)$ on average, assuming uniform hashing.

3. You have access to an oracle that works as follows. Given a set of integers A and an integer m, the oracle answers 'true' if A has a subset, the sum of whose entries is m, and 'false' otherwise. Assume that the oracle answers a query in $O(1)$ time. Give an algorithm that takes as input a set S of integers and an integer k, and outputs a subset of S, the sum of whose entries is k, if such a subset exists. Otherwise it outputs 'no such subset.' Your algorithm should run in $O(n)$ time, where $n = |S|$.

4. Prove that an AVL tree with n nodes has height $O(\lg n)$.

5. Suppose that the keys $\{1, 2, \ldots, n\}$ are inserted into an empty B-tree with minimum degree 2. How many nodes does the final B-tree have?

6. Let $G = (V, E)$ be an unweighted, directed graph. Let $\{u, v\} \subseteq V$. Give a linear-time algorithm to compute the number of shortest paths from u to v. Two paths are distinct if one contains a vertex that is not in the other.

7. Let $G = (V, E)$ be a connected, undirected graph with $|V| \geq 3$. A *separation edge* of G is an edge whose removal disconnects G. A *separation vertex* is a vertex whose removal disconnects G. Prove or disprove: If G has a separation vertex, then it has a separation edge.

8. Suppose you have one machine and n tasks, $a_1, \ldots, a_n$. Each task a_j has a processing time t_j, a profit p_j and a deadline d_j. The machine can process only one task at a time, and task a_j must run uninterruptedly for t_j consecutive time units. If you complete task a_j by its deadline d_j, you receive profit p_j, but if you complete it after its deadline, you receive no profit. Show that the problem of determining whether a profit of at least p can be achieved given n such jobs is **NP**-Complete.

9. Show that the problem from Question 8 is in **P** if we assume that every t_j is an integer between 1 and n.

10. Reduce the shortest path problem for graphs to the longest path problem. Give an approximation algorithm for the longest path problem.