CORE EXAMINATION
Department of Computer Science
New York University
September 23, 2005

This is the common examination for the M.S. program in CS. It covers core computer science topics: Languages and Compilers, Operating Systems, and Algorithms. The exam has two parts. The first part lasts three hours and covers the first two topics. The second part, given this afternoon, lasts one and one-half hours, and covers algorithms.

Use the proper booklet or answer sheet for each question. Each booklet is marked with the Area and Question number, in the form PL&C1, PL&C2, PLC&C3, OS1, OS2, ALGS1 (this question has an answer sheet and not a booklet), ALGS2, ALGS3. Use the appropriate booklet for each question. DO NOT put your name on the exam booklet. Instead, your exam number must be on every booklet.

You will be graded according to your exam number, shown on the envelope containing the booklets. Remember your exam number: when grades are given out, they will be published according to this number, not by name.

Make sure your name and signature are on the envelope. This is the only place where your name appears. Please include all the booklets inside the envelope. You can keep the exam.

Good luck!

Basic Algorithms

Question 1

Question 1 is written on a separate sheet. Please answer it directly on that sheet.

Question 2

Let $G = (V, E)$ be a directed graph, with $n = |V|$. Each edge $e \in E$ is assigned a nonnegative weight $w(e)$. Moreover, each vertex $v \in V$ is assigned a color, either *blue* or *red*. Finally, two distinguished vertices $s, t \in V$ given.

Design an $O(n^2)$-time algorithm that finds a shortest path from s to t that includes a blue vertex; that is, among all paths from s to t that include a blue vertex somewhere along the path (including possibly s or t), your algorithm should find a path of minimal weight (where the weight of a path is the sum of the weight function on the edges in the path).

For simplicity, you need only describe an algorithm that computes the weight of such a path (and not the path itself). You may use any "standard" algorithms as subroutines. Partial credit may be given for algorithms that are slower than $O(n^2)$.

Question 3

We have a joy ride in an amusement park, where riders wait in a queue. We want to assign riders to cars where each car has a weight limit of $M > 0$. The number of riders in a car is unimportant, as long as their total weight is $\leq M$ pounds. For instance, if $M = 400$ and the weights of the riders in the queue are $Q = (40, 190, 80, 220, 100, 90, 50, 160)$ then we can put them into three cars in the following way:

$$S_1 : (40, 190, 80; \quad 220, 100; \quad 90, 50, 160).$$

Solution S_1 says to put the first three riders into the first car, the next two into the next car, etc. It is the "greedy solution" in which you load as much weight as possible into the next available car. Here is another solutions different than the greedy one:

$$S_2 : (40, 190; \quad 80, 220; \quad 100, 90, 50, 160).$$

part 1 (2 points) Write an algorithm in pseudo-Java to implement the greedy algorithm. Describe any data structure that you use in enough detail so that we can easily turn your algorithm into actual code.

part 2 (4 points) Prove that the greedy algorithm is optimal, i.e., the greedy solution has the minimum number of cars.

part 3 (4 points) Now suppose the riders wait in two queues, Q and Q'. The "greedy solution" again require each car to be filled with the maximum weight before it is sent off. Although the riders in each queue must be taken in the queue order, there is no particular priority for two riders if they are in different queues. For instance, with M and Q as before, let $Q' = (160, 200, 110, 50)$ then we could fill the first car with $(40, 190, 80)$ (all from Q) or $(160, 200)$ (all from Q') or $(40, 190, 160)$ (from Q and Q'). The greedy solution would pick the last choice, as this has the maximum weight.

Explain clearly how you can find the maximum weight that can go into the first car. How much time would this procedure take?

part 2 (5 points)

In the following, you are presented with descriptions of recursive algorithms. For each, indicate the rate of growth of the running time by writing one of the letters a–g in the corresponding box to specify one of the following functions:

(a) $\log n$

(b) n

(c) $n \log n$

(d) n^2

(e) $n^2 \log n$

(f) n^3

(g) 2^n

(h) 3^n

☐ On inputs of size n, the algorithm spends time n, plus the time needed to recursively solve two subproblems, each of size $\lfloor n/2 \rfloor$.

☐ On inputs of size n, the algorithm spends time n^2, plus the time needed to recursively solve one subproblem of size $n - 1$.

☐ On inputs of size n, the algorithm spends time 1 plus the time needed to recursively solve one subproblem of size $\lfloor n/3 \rfloor$.

☐ On inputs of size n, the algorithm spends time n^2, plus the time needed to recursively solve three subproblems, each of size $\lfloor n/2 \rfloor$.

☐ On inputs of size n, the algorithm spends time 1 plus the time needed to recursively solve three subproblems, each of size $n - 1$.

ALGS1

part 1 (5 points)

For each of the following pairs of functions, write a letter a–d in the corresponding box, as follows:

(a) if $f = O(g)$ and $g \neq O(f)$,

(b) if $f = O(g)$ and $g = O(f)$,

(c) if $f \neq O(g)$ and $g = O(f)$,

(d) if $f \neq O(g)$ and $g \neq O(f)$.

☐ $f(n) = n \log_2 n$, $g(n) = n^2$

☐ $f(n) = n^{1.999}(\log_2 n)^{10}$, $g(n) = n^2$

☐ $f(n) = (2n + 10)^2$, $g(n) = (3n + 2)^2$

☐ $f(n) = n^2$, $g(n) = n^{\log_2 3}$

☐ $f(n) = \log_{10} n$, $g(n) = \log_2 n$

☐ $f(n) = 10^n$, $g(n) = 2^n$

☐ $f(n) = n^{10}$, $g(n) = n^2$

☐ $f(n) = n^{1/10}$, $g(n) = n^{1/2}$

☐ $f(n) = n!$, $g(n) = 2^n$

☐ $f(n) = \log_2(n!)$, $g(n) = n \log_2 n$