**Final**
**Computer Science 2150**
**Introduction to Algorithms**
**Spring 2011**

Instructions:

1. The test is closed book, closed notes.

2. For most of the problems, I am interested in testing whether you understand the techniques
   and concepts more than I am interested in the solution to the particular problem. For
   example, if I ask you to prove that a problem is NP-hard, I am more interested in learning if
   you know how to prove that a problem is NP-hard, than I am in the specifics of the problem.
   If I ask you problem that a greedy algorithm is correct using an exchange argument, I am
   more interested in learning if you know how an exchange argument works, than I am in
   the specifics of the problem. I ask these questions in the context of specific problems to
   allow you to demonstrate your understanding in a concrete setting. Of course you have
   to take into account the specifics of the problem, but make sure to explain the general
   method/technique/concept that you are using as well.

3. 25 % partial credit is given for the answer "I don't know." A blank answer will be inter-
   preted as "I don't know." False or completely unsubstantiated assertions will receive lesser
   credit.

4. Solve up to 6 of the problems. Clearly write the number of the problem that you are
   solving. If you answer more than 6, an arbitrary 6 answers will be graded.

5. **If you are uncertain about anything, ask a question!**

1. Give a polynomial-time dynamic programming algorithm to compute the shortest bitonic tour of $n$ points in the Euclidean plane. Recall that a bitonic tour starts at the leftmost point, moves right until is hits the rightmost point, and then moves left until it returns to the leftmost point.

2. Give a greedy algorithm for the following problem, and prove using an exchange argument that this algorithm is correct. The input to this problem consists of $n$ jobs. For each job $j$ you are given a release time $r_j$ and a processing time $p_j$. The output is a feasible schedule that minimizes the average completion time of a job. A feasible preemptive schedule is a mapping from each time $t$ to a job, released before time $t$ that is being processed at time $t$. A job $j$ is completed once it has been processed for $p_j$ units of time.

3. Prove that the Subset Sum problem is NP-hard by reduction from Vertex Cover or by reduction from 3CNF-SAT. Recall that the Subset Sum problem takes as input positive integers $x_1, \ldots x_n, L$ and asks whether there is a subset of the $x_i$'s that sums to $L$. Recall that the input to the Vertex Cover problem is a graph $G$ and an integer $k$ and the question is to determine whether $G$ has a vertex cover of size $k$ (a vertex cover is a collection of vertices such that every edge is incident on at least one vertex in the collection). Recall that the input to the 3CNF-SAT problem is a Boolean formula in conjunctive normal form with exactly 3 literals per clause, and the question is to determine if the formula is satisfiable.

4. Give an adversarial argument to show that every comparison based algorithm to determine whether $n$ input numbers are distinct requires $\Omega(n \log n)$ comparisons.

5. Calculate the expected time to insert $n$ items into a hash table of size $n$ using uniform open hashing, which means that each probe into the hash table is equally likely to go to every table energy independent of other probes.

6. Show that if $n$ items are inserted into a closed addressed hash table of size $n$, then expected number of items in the table entry with most items is $O(\log n)$.

7. State/Explain how the push-relabel network flow algorithm works. You do not need to prove that it is correct, or analyze the running time, just state the algorithm.

8. Consider the problem of constructing a maximum cardinality bipartite matching. The input is a bipartite graph, where one bipartition are the girls, and one bipartition is the boys. There is an edge between a boy and a girl if they are willing to dance together. The problem is to matching the boys and girls for one dance so that as many couples are dancing as possible.

   (a) Construct an integer linear program for this problem

   (b) Consider the associated linear program where the integrality requirements are dropped. Explain how to find an integer optimal solution from any rational optimal solution.

   (c) Construct the dual program.

   (d) Give a natural English interpretation of the dual problem

   (e) Explain how one can always give a simple proof that a graph doesn't have a matching of a particular size using the dual using linear programming duality. You should be able to come up with a method that would convince someone who knows nothing about linear programming.

9. Consider an online or approximation problem where there are only finitely many possible algorithms and finitely many possible inputs. We consider generalizing Yao's technique to approximation ratios. Assume that the problem is a minimization problem. Assume that you have an input distribution $\mathcal{I}$, such that for all deterministic algorithms $A$ it is the case that $E[A(\mathcal{I})]/E[Opt(\mathcal{I})] \geq c$. Show that you can logically conclude that the expected approximation ratio for every randomized algorithm is at least $c$.

10. Assume that you want to load $n$ crates with weights $w_1, \ldots, w_n$ onto $k$ trucks so as to minimize the weight of the heaviest load on any one truck. Consider the algorithm that considers the crates in an arbitrary order, and puts the next crate on the truck with the lightest load to date. Show that the approximation ratio for this algorithm is at most 2. Explicitly state what lower bounds you are using for optimal.

11. Assume that we have a dynamic table to which we can insert and delete items. Let us say that each of insertion and deletion costs 1. When the table is 80% full, the table size is doubled. If the table is doubled from size $n$ to size $2n$ then let us say that this costs $2n$. When the table is 20% full, the table size is halved. If the table is halved from size $2n$ to size $n$ then let us say that this costs $2n$. Prove using a potential function argument that the amortized cost per insertion and deletion operation is $O(1)$.

12. Recall that the input to the Knapsack problem was $n$ coins with positive integer weights $w_1, \ldots, w_n$, positive integer values $v_1, \ldots, v_n$, and a weight limit $W$. The problem was to find a subset of the coins with maximum aggregate value subject to the constraint that the aggregate weight is at most $W$. Give an algorithm to solve this problem in $O(W)$ space and $O(nW)$ time. You must justify the correctness of the claimed time and space bounds. Note that the algorithm must produce the actual subset of coins, not just the weight and the value. An algorithm that solves this problem using more time or more space is not worth any partial credit.

13. Consider the paging problem where there are $k$ pages of fast memory and $n > k$ pages of slow memory. So an online algorithm sees over time a sequence of requests to pages in slow memory. If the page requested is not in fast memory, then a page fault occurs. In response to a page fault, the online algorithm has to evict a page of its choice from fast memory, and copy the requested page to fast memory. The objective is to minimize the number of accesses to slow memory. Show that the paging algorithm FIFO with a fast memory of size $k$ has at most twice as many page faults as the optimal paging algorithm with a fast memory of size $k/2$. Recall that FIFO evicts the page that entered fast memory at the earliest point in time.

14. State and prove the "Master Theorem" for divide and conquer recurrence relations. Recall that the Master Theorem gives the solution to recurrences of the form $T(n) = aT(n/b) + n^k$ for constants $a \geq 1$, $b > 1$ and $k \geq 0$.