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 to prove 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 interpreted as “I don’t know.” An answer that displays a major conceptual error will likely receive a grade of zero. It is perfectly fine to give an incomplete answer, e.g. ”Here’s how one proves a problem NP-hard, but I don’t know how to prove this problem NP-hard.” I will make a judgement call on how much credit such answers should receive, but generally it will be around 50%.

4. I will assume that if you write something, that you are asserting that you have good confidence in the correctness of what you write. It is a bad strategy to give an answer that you do not have good confidence in.

5. Answer up to 5 of the problems. Clearly write the number of the problem that you are answering. If you answer more than 5, an arbitrary 5 answers will be graded.

6. If you are uncertain about anything, ask a question!
1. Consider the problem of finding a minimum weight spanning tree in a graph.

(a) Give a polynomial sized integer linear programming formulation $F$.

(b) Consider the relaxation $R$ of $F$ where the variables are allowed to be arbitrary rational numbers (not necessarily integers). Explain how to round a rational optimal solution to $R$ to obtain an integer optimal solution to $F$.

2. State Yao’s theorem within the context of Monte Carlo algorithms. Use Yao’s theorem to prove that no comparison based Monte Carlo algorithm can sort $n$ numbers using $3n$ comparisons.

3. Prove that the vertex cover problem (deciding whether a given graph has a vertex cover of a given size) is NP-hard using the fact that 3SAT (determining whether a Boolean formula in conjunctive normal form with exactly 3 literals per clause is satisfiable) is known to be NP-hard.

4. Assume that you have a collection of $n$ boxes arriving online over time that must be loaded onto $m$ trucks. When a box arrives, the online algorithm learns the weight of the box, and a list of trucks that that box can be loaded on. So not every box is allowed to be loaded on every truck. At the time that a box arrives, the online algorithm must pick a truck to load the box on. The objective is to minimize the weight of the most heavily loaded truck. Give an adversarial argument to show no deterministic online algorithm can achieve approximation ratio $O(1)$.

5. Consider the following online problem. There are two taxis on a line that initially start at the origin. At positive integer time $t$, a request point $h_t$ on the line arrives. In response, each taxi can move to a different location on the line, or stay put at the current point. The path traveled by at least one of the two taxis must cross $h_t$. The objective is to minimize the total movement of the taxis. Consider the following algorithm $A$. If both taxis are to the left of $h_t$, then the rightmost taxi moves to $h_t$. If both taxis are to the right of $h_t$, then the leftmost taxi moves to $h_t$. If $h_t$ is between the two taxis, then both taxis move toward $h_t$ at the same rate until one of the taxis reaches $h_t$, at which point both taxis stop moving. Show that this algorithm is 2-competitive using the following potential function: $\Phi = (\text{the distance between the leftmost taxi for } A \text{ and the leftmost taxi for optimal}) + (\text{the distance between the rightmost taxi for } A \text{ and the rightmost taxi for optimal}) + (\text{the distance between the leftmost and the rightmost taxis for } A)$.

6. Consider the following problem. The input is a collection $C = \{(x_1, y_1) \ldots (x_n, y_n)\}$ of points in two dimensions. The output is the minimum perimeter vertically-simple polygon with for which all the points in $C$ are on the perimeter. A polygon is vertically simple if
no vertical line intersects the polygon more than twice. Give a polynomial time algorithm for this problem.

7. Consider the following problem. The input is $n$ disjoint line segments contained in an $L$ by $L$ square $S$ in the Euclidean plane. The goal is to partition $S$ into convex polygons so that the interior of every polygon intersects at most one line segment. So it is ok for a line segment to be in multiple polygons, but each polygon can intersect at most one line segment.

Consider the following algorithm that maintains a partition $C$ of $S$ into polygons. Initially this collection is just the single polygon $S$, that is $C = \{S\}$. Let $\pi$ be a random permutation of the line segments. While there is a polygon $P \in C$ that contains more than one line segment: let $l$ be the first line segment in the $\pi$ order that intersects $P$, and cut $P$ into two polygons using the linear extension of $l$ (so you extend the line segment $l$ into a line and then use that to cut $P$). Show that the expected number of resulting polygons is $O(n \log n)$.

8. State the Bellman-Ford algorithm, and explain how it can be used to determine whether one can make a profit in currency arbitrage. In the currency arbitrage problem, you are given $n$ currencies $c_1, \ldots, c_n$, and for each ordered pair $(c_i, c_j)$ you are given an exchange rate $e_{i,j}$ which is the amount of currency $c_j$ you can obtain from one unit of currency $c_i$. The problem is to determine whether there is a currency $c_i$ and a sequence of exchanges, where you can end up with more of currency $c_i$ than you started with.

9. Consider the parallel prefix problem. The input is $n$ integers $x_1, \ldots, x_n$. The output is $n$ numbers representing the sums of all $n$ nonempty prefixes. So if the input was 1, 4, 3, 6, 5, 9, the output would be 1, 5, 8, 14, 19, 28. Give a parallel algorithm for this problem that would run on a PRAM where neither concurrent reading or concurrent writing is allowed that would run in time $O(\log n)$ if the number $p$ of processors was $n$.

10. Consider the following variant of the Traveling Salesman Problem (TSP). The input is $n$ points in the plane (or some other finite metric space). The output should be the shortest route to visit all points and return to the starting point. Give a polynomial time algorithm that has an approximation ratio of at most 2. Prove that the approximation ratio of your algorithm is at most 2.