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 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 reasonable confidence in.

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

6. If you are uncertain about anything, ask a question!
1. Prove the correctness of Dijkstra’s single source shortest path algorithm using an exchange argument. Start with a brief description of Dijkstra’s algorithm. Then explain how an “exchange argument” works.

2. A palindrome is a nonempty string that reads the same forward and backwards. Give an $O(n^3)$ time dynamic programming algorithm to find the longest palindrome that is a subsequence of a given input stream.

3. Give a polynomial-sized linear program for the problem of finding the minimum spanning tree of an undirected edge-weighted graph. Explain how to obtain in polynomial time a minimum spanning tree from the potentially fractional solution produced by a black-box linear programming solver.

4. Consider comparison based algorithms for some offline problem. State Yao’s theorem in this context for lower bounding the time required by Las Vegas algorithms. Include a definition of Las Vegas algorithms. Explain why Yao’s theorem in this context is a consequence of weak duality of linear programs. Include a definition of weak duality.

5. Consider the following online problem. You given a sequence of bits $b_1, \ldots, b_n$ that arrive over time. Each bit is in an envelope. You first see the envelope for $b_1$, then the envelope for $b_2$, etc. When you get the envelope $i$, you can either look inside to see the bit, or destroy the envelope (in which case you will never know what the bit is). You know a priori that at least $n/2 + 1$ of the bits are 1. The goal is to find an envelope containing a 1 bit. You want to open as few envelopes as possible. Show using Yao’s technique for Monte Carlo algorithms that every Monte Carlo algorithm must open $\Omega(\log n)$ envelopes if it is to be incorrect with probability $< 1/n$. Start by stating precisely the version of Yao’s technique/theorem that you will use. You do not need to prove Yao’s theorem, just state it correctly, and apply it correctly.

6. Assume that you have a collection of $n$ boxes $b_1, \ldots, b_n$ 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 $O(1)$.

7. The input for the bottleneck traveling salesman problem is a complete undirected graph with positive edge weights satisfying the triangle inequality. Feasible solutions are tours that visit each vertex exactly once. The objective is to minimize the length of the longest
edge in the tour. Give a polynomial time 3-approximation algorithm for this problem. You
must prove your algorithm produces a 3-approximation. Start with a definition of what it
means for an algorithm to be a 3-approximation algorithm in this setting.

8. Consider the online paging/caching problem. The input is a sequence of pages. If a page is
not currently in a cache, a cache miss occurs. The objective is to minimize the number of
cache misses. Show that ejecting the Least Recently Used page from cache of size \( k \) results
in at most twice as many cache misses as the minimum possible number of misses for a size
\( k/2 \) cache.

9. Prove that if there is a polynomial time approximation algorithm for the maximum clique
problem that has approximation ratio 1000 then there is a polynomial time approximation
algorithm with approximation ratio 1.000000001.

10. State the push-relabel algorithm for network flow. Explain why the algorithm terminates
in polynomial time with the correct answer when the network is a single path from the
source to the sink.

11. Assume that you have to solve the hiring problem at a large institution where effectively you
can’t fire anyone (note that this is a realistic assumption). Thus once you hire someone, the
game is over. Find a strategy that will hire the best person with probability \( \Omega(1) \) assuming
that each permutation is equally likely. Note that for the purposes of this problem, hiring
the second best person, hiring no one, and hiring the worst person are all equivalent, the
only thing that matters is hiring the best person. The person hiring knows \( n \), the number
of applicants, a priori. Justify that the algorithm hires the best person with probability
\( \Omega(1) \).

12. You have a sorted array \( A \) of containing \( n \) real numbers each selected independently and
uniformly at random from the interval \([0, 1]\). You have an real \( x \) in \([0, 1]\). The problem is
to find a subarray of size \( \sqrt{n} \) that contains \( x \). Give an algorithm with expected running
time \( O(1) \). You must prove this claimed running time. You may use without proof the fact
that the probability that a binomially distributed random variable \( x \) is more than \( 1 + \delta \)
times its mean \( \mu \) is at most \( e^{-\mu\delta^2/4} \), and the fact that the probability that a binomially
distributed random variable \( x \) is less than \( 1 + \delta \) times its mean \( \mu \) is at most \( e^{-\mu\delta^2/2} \).

13. A binary search tree is \( \alpha \)-balanced if for every node \( x \), the size of the left subtree of \( x \) is
at most \( \alpha \) times the size of the subtree rooted at \( x \), and the size of the right subtree of \( x \)
is at most \( \alpha \) times the size of the subtree rooted at \( x \). Consider a binary search tree with
standard insert and delete operations, with the following exception. Let \( x \) be the highest
node that is no longer \( 2/3 \) balanced. Then every node in the subtree rooted at \( x \) is made
balanced. You may assume without justification that the time required is linear in the size of the tree rooted at \( x \). Show using the following potential function that the amortized time for \( n \) insert and delete operations is \( O(\log n) \). Define \( \Delta(x) \) to be the absolute value of the difference in the size of the left subtree of node \( x \) and the right subtree of node \( x \). Then the potential function

\[
\Phi = c \cdot \sum_{x: \Delta(x) \geq 2} \Delta(x)
\]

Start by explaining how a potential function argument works in this setting, and what you need to prove in order to establish \( O(\log n) \) amortized time.

14. State and prove the master theorem for recurrences of the form

\[
T(n) = aT(n/b) + n^k
\]

for integer constants \( a \geq 1, b \geq 2 \) and \( k \geq 0 \).

15. You are given the exchanges rates between \( n \) different currencies. Give an \( O(n^3) \) time algorithm to determine whether you can make money via arbitrage (assume no costs for conversion between currencies). Prove the correctness of this algorithm.