

Final
Computer Science 2150
Introduction to Algorithms
Spring 2008

Instructions:

1. The test is closed book, closed notes.
2. If you are taking the final for preliminary exam credit, then write your preliminary number given to you by Keena on the exam answer sheet, but do not write your name. If you are not taking the final for preliminary exam credit, write your name on the exam answer sheet.
3. If you are taking the course for some grading option other than the standard letter grading option, please specify that on your exam answer sheet.
4. 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 a 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.
5. You must fully justify all assertions unless specifically instructed not to do so. All arguments should be from first principles. You should not use results from class, the book, etc. as a black box.
6. 25 % partial credit is given for the answer “I don’t know.” A blank answer will be interpreted as “I don’t know.” False or completely unsubstantiated assertions will receive lesser credit.
7. Solve up to 10 of the 15 questions. Clearly write the number of the problem that you are solving. If you answer more than 10, an arbitrary 10 answers will be graded.
8. **If you are uncertain about anything, ask a question!**

1. Give a general solution to the recurrence relation $T(n) = aT(n/a) + n \lg^k n$ for constants $a > 1$ and $k \geq 0$.
2. Consider the problem where the input is a collection of intervals, and the output is a maximum cardinality collection of disjoint intervals. Give a greedy algorithm for this problem and prove that it is correct using an exchange argument.
3. Two of the three following definitions of big Oh area logically equivalent. Identify the two that are logically equivalent, and prove that they are equivalent. Take care in your exposition. Prove that the third definition is not logically equivalent to the other two. You may assume that we are only considering functions that are defined on the positive integers and are positive and non-decreasing.
 - (a) $f(n) = O(g(n))$ iff $\exists c \exists N \forall n > N f(n) \leq cg(n)$
 - (b) $f(n) = \hat{O}(g(n))$ iff $\exists c \forall n f(n) \leq cg(n)$
 - (c) $f(n) = \tilde{O}(g(n))$ iff $\exists N \forall n > N \exists c f(n) \leq cg(n)$
4. Assume that problem P has worst-case time complexity $o(n^3)$. Assume that problem P has worst-case time complexity $\Omega(n^2)$. Assume that A is an algorithm for problem P . For each of the following statements, state whether the statement is logically implied by the above information, and state whether the statement is logically consistent with the above information. So I am looking for 8 yes/no answers. You need not justify your answers. But this question will be graded on an all or nothing basis, so take that into account.
 - (a) Algorithm A has worst-cast time complexity $O(n^2)$.
 - (b) Algorithm A has worst-cast time complexity $\Omega(n^3)$.
 - (c) Algorithm A has worst-cast time complexity $\omega(n)$.
 - (d) Algorithm A has worst-cast time complexity $o(n)$.
5. Consider the following problem. The input is an n by n array E . Entry $E[i, j]$ specifies the amount of currency j will be paid in exchange for one unit of currency i . The problem is to determine whether you have any arbitrage opportunities. That is, you want to determine whether there is any sequence of exchanges that would allow the amount of money in some particular currency to grow in an unbounded way. You can assume that you start with one unit of each currency. Give a polynomial time algorithm for this problem. You must justify the correctness of your algorithm.
6. Assume that you have a server that fails on a particular day of n days with probability 1 percent independent of whether it fails on other days. Let X_i be a random variable denoting

the maximum number of days that the server fails starting from day i . So if $X_i = k$ then the server fails on days $i, i + 1, \dots, i + k - 1$, but either the server doesn't fail on day $i + k$ or $i + k = n + 1$. Compute the expected value of $\max(X_i, \dots, X_n)$.

7. We consider the Quicksort algorithm for sorting n numbers. Let $X_{i,j}$ be an indicator random variable denoting whether the i th smallest number is compared to the j th smallest number. So $X_{i,j}$ is 1 if the i th smallest number is compared to the j th smallest number, and 0 otherwise. What is $E[X_{i,j}]$ as a function of i, j and n ? Use this to compute the expected number of comparisons used by Quicksort.
8. Consider the problem where the input is n keys $K_1 < \dots < K_n$, with access probabilities p_1, \dots, p_n . The output should be a binary search tree T on these keys that minimizes the expected access time $\sum_{i=1}^n p_i \text{Depth}(T, k_i)$. Give an polynomial time dynamic programming algorithm for this problem. Make sure to explain how information is stored in your array.
9. Recall that the k -CNF-SAT problem takes as input a Boolean formula F in conjunctive normal form with exactly k literals per clause, and determines whether or not F is satisfiable. Assume that you know that 3-CNF-SAT is NP-complete. Use this fact to establish that 4-CNF-SAT is NP-hard.
10. We consider the setting of a two player zero-sum game described by a two dimension table specifying how much the column player pays the row player for each possible situation. Assume the row player goes first, and specifies a probability distribution over his possible moves. The column player then replies with the move that minimizes her expected payout to the row player. We consider the problem of determining the probability distribution for the row player that will maximize his expected payout. Explain how to construct a linear programming formulation of this problem. It is sufficient to construct the linear program for the following instance.

	X	Y	Z
A	-4	3	5
B	1	2	-6
C	7	-8	9

Then give the dual linear program for your linear program for the above instance.

11. Consider the red and blue jug problem. In this problem you have n red jugs and n blue jugs. You know that for each jug that there is a unique jug of the other color with the same volume. The only way that you can learn about the volumes of the jugs is to compare two jugs of different colors. This operation tells you whether the red jug has greater volume,

or whether the blue jug has greater volume, or whether they have the same volume. Show using an adversarial argument that any deterministic algorithm that solves this problem requires $\Omega(n \log n)$ operations.

12. Consider again the same red and blue jug problem. In this problem you have n red jugs and n blue jugs. You know that for each jug that there is a unique jug of the other color with the same volume. The only way that you can learn about the volumes of the jugs is to compare two jugs of different colors. This operation tells you whether the red jug has greater volume, or whether the blue jug has greater volume, or whether they have the same volume. Assume that you know that every linear time deterministic algorithm A is wrong on at least half of the $(n!)^2$ possible inputs in the collection \mathcal{I} of possible inputs. Explain why for every randomized algorithm \mathcal{A} it must be the case that there exists an input I on which \mathcal{A} is wrong with probability at least half. Your argument must be from first principles.
13. We consider the standard network flow problem. Let V be the number of vertices in the network, E be the number of edges in the network and F be the maximum edge capacity in the network. Assume that you have network flow algorithms A , B and C with worst-case running times $A(V, E, F) = \Theta(V^2 F)$, $B(V, E, F) = \Theta(V^2 \lg^3 F)$, and $C(V, E, F) = \Theta(V^2 E^2)$. Which of these algorithms are polynomial time algorithms? Start with a definition of polynomial time algorithm, and justify your answer.
14. Assume that you have n items with weights w_1, \dots, w_n that you wish to pack into k trucks so as to minimize the weight of the most heavily loaded truck. Consider the greedy algorithm that considers that items in an arbitrary order and packs that item onto the truck that currently has the least load. Prove that the greedy algorithm has worst-case ratio at most two for this problem. State which two lower bounds to optimal that you use.
15. Consider the online list update problem. The online algorithm maintains a list. The input in each round is an item I in the list. If the item I is in position k in the list, then the algorithm pays a cost of k . Further, at no cost, the item I may be to any position in the list that is not further from the front of the list than I 's current position. Consider the algorithm MTF that always moves the accessed item closer to the front. Show that MTF has cost at most twice the optimal cost for every input sequence using the potential function $\Phi =$ the number of transpositions between MTF's list and the optimal list. A pair (I, J) of items is a transposition if I and J occur in different order in MTF's and the optimal list.