

Final Exam

December 15, 2004

READ THIS PAGE. DO NOT TURN TO THE NEXT PAGE UNTIL YOU ARE TOLD YOU CAN START.

- NAME: _____
- This is the CSE 441T exam. If you are a CSE 541T student please raise your hand so I can give you the CSE 541T exam.
- In the end I decided against having short answer questions. There are 5 questions for you to answer that are in the same style as those you've had in the past.
- If any question is not clear to you, come up to the front and I will answer any questions you have. There is no point cost (unlike with the hints) if you want to ask for clarification about the problem itself.
- Please write up all solutions clearly and legibly in the space provided. If needed you can attach an extra piece of paper. Be sure to write your name on it. If you need scratch paper please come up front.
- For any problem if you feel that you are stuck and need a hint, please come to the front and for an appropriate reduction in the maximum score you can get on that problem a hint will be provided. I have written the exam so that very few hints should be needed – most are already included with the problem.
- When you are asked to give a lower bound, spend at most 15 minutes determining the best bound you can prove and then write-it up. If you have extra time you can come back later and see if you have any ideas for improvements. Your write-up should have the following steps:
 - State the lower bound which you will prove.
 - Clearly describe your adversary strategy.
 - Prove the stated lower bound.
 - (OPTIONAL) Describe an alternate adversary strategy that you think is better but for which you can't prove any better bound. Also feel free to give a few sentences of thoughts about how you might prove a better bound with it.
- To give you a sense of how to pace yourself, this was designed as a 2 hour exam with the expectation that a p point problem should take you about p minutes. You have 3 hours available so you can spend up to $1.5p$ minutes on a p point problem and finish on time.
- Relax and begin working when instructed to start.

1. (10 points) Here we consider the problem of deciding where to put line breaks when formatting text where the goal is just to minimize the number of lines needed in the layout. The input is a sequence of n words w_1, w_2, \dots, w_n of lengths $\ell_1, \ell_2, \dots, \ell_n$, measured in characters. The maximum number of characters that can be placed on a line is M . (Assume $\ell_i \leq M$ for all i .)
Prove that the greedy algorithm of putting as many words as you can on each line will produce a layout with as few lines as possible. You need not analyze the time complexity of this algorithm.

2. (25 points) The Longest Simple Path (LSP) problem is defined as follows. You are given as input a weighted graph G where all edge weights are positive, and a source vertex s and destination vertex t . The problem to be solved is to find the weight of the longest simple path from s to t . A path is simple if it visits each vertex at most once. Under the assumption that $P \neq NP$ prove that there is no polynomial time algorithm that can solve this optimization problem.

You can make use of the fact that the Hamilton-Path problem is NP-complete. In the Hamilton Path problem you are given as input an unweighted undirected graph $G = (V, E)$ and two specified vertices $u, v \in V$. The question is whether or not there is a simple path from u to v in the graph such that this path visits every vertex exactly once.

3. (30 points) The Max Alternating Sum Subsequence (MASS) problem which takes as input a sequence $\langle x_1, x_2, \dots, x_n \rangle$ of non-negative integers. The goal is to find a subsequence $A = \langle x_{i_1}, x_{i_2}, \dots, x_{i_k} \rangle$ where $i_1 < i_2 < \dots < i_k$ that maximizes $x_{i_1} - x_{i_2} + x_{i_3} - x_{i_4} \dots$.

Here are two example inputs and their solutions. If $X = \langle 4, 9, 2, 4, 1, 3, 7 \rangle$ then an optimal solution is $A = \langle 9, 2, 4, 1, 7 \rangle$ which has the value of $9 - 2 + 4 - 1 + 7 = 17$. If $X = \langle 7, 6, 5, 4, 3, 2, 1 \rangle$ then the optimal solution is $A = \langle 7 \rangle$.

You are to use dynamic programming to design an algorithm to optimally solve this problem. I will get you started. There are two general subproblem forms that you will use and then combine. They are:

- Let $O[i]$ be the maximum odd (i.e. k is odd) length solution for the input $\langle x_1, \dots, x_i \rangle$.
- Let $E[i]$ be the maximum even length solution for the input $\langle x_1, \dots, x_i \rangle$.

- (a) (10 pts) Write a recursive definition for both $O[i]$ and $E[i]$.

- (b) (3 pts) Given a formula for the optimal value for the original problem (i.e. input $X = \langle x_1, \dots, x_n \rangle$).

- (c) (5 pts) Argue that your definition for $O[i]$ is correct. (A similar argument should apply for $E[i]$ but you don't need to include it here.)
- (d) (7 pts) Write pseudo-code (or code) for an iterative, non-recursive procedure to return the optimal value for input X . You may assume that X is stored in an array X where $X[i]$ holds the value of x_i . Similarly, you can assume that the arrays O and E are defined. Also, you can assume that a `max` function is defined.
- (e) (5 pts) Analyze the asymptotic time complexity of your algorithm.

4. (40 points) Give the best lower bound for two of the following three problems. (If you answer all three, clearly indicate which two we should grade.) For each problem you solve be sure to do each of the following steps (in the given order): (1) state the lower bound which you will prove, (2) clearly describe your adversary strategy, and (3) prove your stated lower bound.
- (a) A unimodal array A has the property that it is strictly increasing up until some index m and then is strictly decreasing. Thus $A[m]$ is the maximum. The model of computation here is that the algorithm only access A by asking if $A[i] < A[j]$ for any i, j . Prove the best lower bound you can on the number of comparisons that must be made for the problem of finding m in a unimodal array of n elements.
 - (b) Let A be an array of $2k + 1$ elements for k any integer ≥ 0 . A majority element of A is one that occurs at least $k + 1$ times. The model of computation here is that the algorithm can ask if $A[i] = A[j]$ for any i, j . Prove the best lower bound you can for the number of equality questions that must be posed for the problem of determining if there is a majority element.
 - (c) Let $B = (b_1, b_2, \dots, b_n)$ be an arbitrary bit string. The model of computation here is that the algorithm can only find out about b_i by posing a query asking for the value of b_i . Prove the best lower bound you can on the number of bits that must be queried for the problem of determining if there are 3 consecutive 1s in B .

Additional Space for Problem 4.

5. (15 points) Consider the following on-line problem (which is a variant of that from HW 5). You are given a sequence of prices $\langle p_1, p_2, p_3, \dots, p_m \rangle$ where $p_i \in \{1, 2, 3, 4\}$ (that is each price is either 1,2,3, or 4). When presented with p_i you can either buy the item for that price, or pass in which case you have committed to buying that item at price among p_{i+1}, \dots, p_m . Your goal is to buy the item for the cheapest price. Here is a randomized on-line algorithm A for this problem. Flip a fair coin. If the coin is a head then only buy an item if the price is 1. If the coin is a tail then you only buy an item if the price is 1 or 2. (In both cases if p_m is reached and the item has not yet been purchased then you must buy it at cost p_m).

What is the competitive ratio of this randomized on-line algorithm? Be sure to show your work with enough detail that we can see how you arrived at your answer.

Problem	Points Possible	Points Received
1	10	
2	25	
3	30	
4	40	
5	15	
total	120	