

Homework 7 – Due Friday, October 17, 2008

Please refer to the general information handout for the full homework policy and options.

Reminders

- Your solutions are due before the lecture. Late homework will not be accepted.
- Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to a member of the course staff if asked. You must also identify your collaborators. *Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.*
- To facilitate grading, please write down your solution to each problem on a separate sheet of paper. Make sure to include all identifying information and your collaborators on each sheet. Your solutions to different problems will be graded separately, possibly by different people, and returned to you independently of each other.
- For problems that require you to provide an algorithm, you must give a precise description of the algorithm, **together with a proof of correctness** and an analysis of its running time. You may use algorithms from class as subroutines. You may also use any facts that we proved in class or from the book.

Exercises These should not be handed in, but the material they cover may appear on exams:

1. Chapter 6 exercises.
2. Given two strings x, y of length m and n (respectively), you would like to compute a shortest possible string of which both x and y are substrings. Give an algorithm which outputs such a sequence in time $O(mn)$.

Problems to be handed in

<p>Page limits: The answer to each problem should fit in 2 pages (or one double-sided sheet) of paper. Longer answers will be penalized.</p>
--

1. (**Segmented LS with exactly k pieces**) Consider the following variant of the Segmented Least-Squares Problem (KT Chapter 6.3): given n points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ with distinct x coordinates, we want to compute a piece-wise-linear approximation to the point set *with exactly k pieces* that minimizes the sum squared error (that is, rather than minimizing $E + C \cdot L$ as in class, we want to minimize E subject to $L = k$).

Your algorithm should run in time $O(kn^2)$. You may assume that the numbers e_{ij} used in the algorithm in class have already been computed.

2. (**Longest well-colored path**) Suppose you are given an undirected graph G , along with a coloring function $c : V \rightarrow \{1, 2, \dots, k\}$ that assigns one of k possible colors to each vertex. We say a path in G is well-colored if every vertex along the path (including the start and end points) has a different color. Of course, well-colored paths can have length at most $k - 1$.

Give a $O(2^k \cdot \text{poly}(n))$ -time algorithm that outputs the longest well-colored path in G . (Here, $\text{poly}(n)$ can be any fixed polynomial in n that doesn't depend on k .)