Homework 7 – Due Friday, March 2, 2012

Reminders 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.

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

- The segmented least squares algorithm we saw in class required values \( e(i, j) \) be precomputed \((e(i, j) \text{ was the total squared error of the best-fit single line for points } i, ..., j)\). The straightforward algorithm for computing all the \( e(i, j) \) values runs in time \( \Theta(n^3) \), since each individual term can be computed in time \( \Theta(|j - i|) \) using the formulas stated in class.

  Give an algorithm that computes a table with all the \( E(i, j) \) values and runs in time \( O(n^2) \).

- Erickson’s dynamic programming exercises 1–9.  

- Erickson’s greedy algorithms exercises 1–8.  

Problems to be handed in. Please submit each problem on a separate sheet of paper.

1. (Segmented Least Squares, Revisited) Give an algorithm that takes a sequence of points in the plane \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\) and an integer \( k \) as input and returns the best piecewise linear function \( f \) consisting of at most \( k \) pieces that minimizes the sum squared error \( \sum_i (y_i - f(x_i))^2 \).

  That is, instead of adding a penalty term \( cL \) for the number of lines \( L \) (as we did in class), here we put a hard limit on the number of lines in our function.

  Follow the guidelines from the previous problem set in presenting your answer. Be sure to the running time and space usage of your algorithm.

  For full credit, your solution should use \( O(n^2 k) \) time and \( O(n k) \) space.  
  (Hint: Build up a table \( M \) of \( n \times k \) entries. Each entry \( M[i, L] \) in the table corresponds to the optimal fit of \( L \) lines to a subset of the data.)

Extra credit Code up your algorithm from the previous problem in Python. Submit your Python code as well as some test results on data that you think show your algorithm’s correctness. (For each test data set, plot the data as well as the piecewise-linear function output by your algorithm.)

When deciding which test results to submit, ask yourself what test results you would want to see if you were evaluating someone else’s code. For simplicity, submit at most 5 tests.
2. (Greedy Algorithms: Interval Sub-Covers) Given a family of intervals $[a_i, b_i], \ i = 1, \ldots, n$, a sub-cover is subset of the intervals that covers the same area of the real line as the union of all the intervals. For example, in the following picture, the gray intervals form a sub-cover consisting of 5 intervals:

![Diagram of intervals]

(Warm-up exercise: what is the smallest subcover for this family?)

You want to design an efficient algorithm that finds a sub-cover with as few intervals as possible. In the following, you may assume that (i) all the endpoints $(a_i, b_i)$ are distinct, and (ii) the original set of intervals covers a contiguous segment of the real line.

(a) Your friend suggests the following greedy approach: at each stage, add the interval with the most length not covered by the intervals selected so far. Give an example showing that this approach will not find the smallest sub-cover.

(b) Give a polynomial-time algorithm to find a smallest sub-cover, given the $a_i$'s and $b_i$'s as input. Faster (correct) algorithms are worth slightly more points than slower ones.

(c) What is the running time of your algorithm?

(d) Explain carefully why your algorithm is correct. You may find it helpful to read the rest of the question first.

(e) Given a set of intervals, a spread is a set of points no two of which are contained by a single interval. Prove that if there exists a spread of size $k$, then every sub-cover must have size at least $k$.

(f) Show that for any collection of intervals, the size of the largest spread is exactly equal to the size of the smallest sub-cover. [Hint: can you find an algorithm that produces a sub-cover and a spread of the same size?]