Algorithms – CS-37000 Homework 7 – Feb 9; updated Feb 11, 2006

Instructor: László Babai Ry-164 e-mail: laci@cs.uchicago.edu

ADVICE. Take advantage of the TA's problem sessions. This is the **principal venue to discuss past homework and test problems.** Note that such problems may prop up in future tests.

READING (updated on Feb 11, 3:30pm). Review from Discrete Math: Markov Chains. Finite Probability Spaces. - Review all previous handouts and readings (including convolution and Fast Fourier Transform). Text: closest pair of points in the plane (Chap. 5.4), hashing (Chap. 13.6). Recommended reading (if you aspire for an A or A-): use hashing to solve the closest pair problem in the plane in $O(n)$ expected time (Chap. 13.7).

HOMEWORK. Please **print your name on each sheet.** Put each solution on a separate sheet. Please try to make your solutions easily readable.

This homework is due on **Tuesday, February 14**. at the **beginning of the class**.

Problems updated on Feb 11, 3:30pm. The requirement of proof to 7.1(a) was highlighted and 7.1(b) was added. The requirement of pseudocode was added to 7.2.

7.1 (8+6 points)

(a) The weather on planet X follows the following pattern: most days are sunny and it rains for one day every once in a while (it never rains two days in a row). Suppose the average number of consecutive sunny days is D. Design a Markov Chain with two states, SUN and RAIN, that models the weather on planet X. **Prove that your model corresponds to the specification.** Warning: you will need to calculate the expected value of a random variable which takes infinitely many values. This requires a natural extension of the material in the Finite Probability Spaces handout.

(b) Construct a Markov Chain with $D + 2$ states which models the pattern that each string of sunny days between two rainy days lasts either $D - 1$ or D or $D + 1$ days and these three outcomes occur with equal frequency. Explain why this is a Hidden Markov Model.

7.2 (12 points) Let S be a set of n points in the plane. Design an algorithm for finding a pair of points a and b that are as far apart as possible. Your algorithm should run in $O(n \log n)$ time. **Describe your algorithm in pseudocode.** (Hint: use convex hulls.)