Algorithms – CS-27200/37000 Homework – January 14, 2003

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

ADVICE. Take advantage of the TA's office hours on Tuesday and Thursday, 5–6pm in the Theory lounge, Ry-162

READING (due next class) Handouts: dynamic programming, knapsack problem. Review PSEUDOCODE conventions used in the handouts. Review all previous handouts.

HOMEWORK. Please **print your name on each sheet.** Print "U" next to your name if you seek 27000 credit and "G" if you seek 37000 credit, regardless of your grad/undergrad status. Please try to make your solutions readable. Unless expressly stated otherwise, all solutions are due at the **beginning of the next class.**

Homework is collected in three separate piles (U, G, "G only"). Please put your solutions to "G only" problems on that pile, and your solutions to other problems on the "U" or "G" pile according to the credit you seek.

4.1 (U, G) (12 points) *(All-ones square problem.)* Given an $n \times n$ array A of zeros and ones, find the maximum size of a contiguous square of all ones. (You do not need to locate such a largest all-ones square, just determine its size.) Solve this problem in *linear time.* "Linear time" means the number of steps must be O(size of the input). In the present problem, the size of the input is $O(n^2)$. Manipulaing integers between 0 and n counts as one step; such manipulation includes copying, incrementing, addition and subtraction, looking up an entry in an $n \times n$ array.

Describe your solution in **pseudocode.** The solution should be *very simple,* no more than a few lines. **Elegance counts.** *Hint:* dynamic programming. Example:

1	0	1	1	0	1
1	1	0	1	1	1
1	0	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	0
1	1	1	0	1	1

(OVER)

In this example, the answer is 3. There are three contiguous 3×3 square subarrays of all-ones. One is indicated below by underlines, another is shown in a box, the third one is indicated by *Italics.*

1	0	1	1	0	1
1	1	0	*1*	*1*	*1*
1	0	1	*1*	*1*	*1*
1	1	1	*1*	*1*	*1*
1	1	1	1	1	0
1	1	1	0	1	1

4.2 (U, G) (a) (2 points) Let $\{x_n\}$ be a sequence of real numbers. Define, using ϵ and a threshold value n_0, the statement that $\lim_{n\to\infty} x_n = 1$. Your answer should be a well-quantified, properly formed formula involving no English words. Warning: the order of the quantifiers is essential. (b) (G only) (4 points) Let a_n, b_n, c_n, d_n be sequences of *positive* real numbers. Prove, using the definition from (a): if $a_n \sim c_n$ and $b_n \sim d_n$ then $a_n + b_n \sim c_n + d_n$.

4.3 (G only) (2 points) Prove: $\binom{n}{5} \sim cn^d$ for some constants c, d. Determine c, d.

4.4 (G only) (a) (3 points) Prove: if p is a prime number and $x^2 \equiv 1 \pmod{p}$ then $x \equiv \pm 1 \pmod{p}$. (b) (8 points) Prove: if $N = pq$ where p and q are distinct odd primes then $(\exists x)(x^2 \equiv 1 \pmod{N}$ AND $x \not\equiv \pm 1 \pmod{N})$. (The universe of the variable x is $\mathbb{Z}$.)

4.5 (G only) Consider the Knapsack problem described in the last handout. The input parameters in that problem are positive reals called *weights* and *values*; and a "weight limit" W is given. It is shown in the handout how one can solve this problem in $O(nW)$ steps (arithmetic, comparison, bookkeeping) if all *weights* (including W) are *integers.*

(a) (15 points) Assume now that all *values* are *integers* (but the weights are real). Let V denote the sum of the values. Solve the knapsack problem in $O(nV)$ steps under this assumption. Your solution should be a simple **pseudocode**.

(b) (15 points, due Friday, January 20) *(Efficient approximation algorithm.)* Assume both the weights and the values are real and assume an error parameter ϵ is given ($0 < \epsilon < 1/2$). Find the maximum knapsack value within $\pm\epsilon V$ error. (Your solution must satisfy the weight constraint exactly, not approximately; the error is in comparison with the (unknown) optimum solution.) *Complexity:* Your algorithm should use $O(n^2/\epsilon)$ steps. The steps include bookkeeping, arithmetic and comparison of $O(\log(n/\epsilon))$-digit integers, arithmetic of reals, and rounding reals to the nearest integer.