Algorithms – CS-27200/37000 Homework – January 14, 2005

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

IMPORTANT. If you have not done so yet, please send e-mail to the instructor with your name, major, year, type of credit sought (letter grade, P/F, etc.), list of proof-oriented math courses previously taken; include whether or not you took CMSC-27100 (Discrete Math). In the subject write 27200 info or 37000 info, as approrpiate.

HOMEWORK. Please **print your name on each sheet.** Print "U" next to your name if you seek 27200 credit and "G" if you seek 37000 credit. Undergraduates receive the stated number of points as *bonus points* for "G only" problems. – 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) (6 points) Let H be a heap and x the address of a node in H. Let us change the value $key(x)$ to $newkey(x)$ and suppose $newkey(x) < key(x)$. Write a PSEUDOCODE to restore the heap structure. Estimate the number of comparisons made in the process. Your estimate should be a function of n, the number of items in the heap (which is the number of nodes of the heap). Your estimate should be asymptotically best possible (with the correct constant).

4.2 (G only) (10 points) Suppose we are given n data (reals) arranged in a heap. Prove: sorting the data still requires asymptotically $n \log n$ comparisons. (The meaning is that "preprocessing" the data by arranging them in a heap does not significantly reduce the cost of sorting.) WARNING: this problem is NOT about the HEAPSORT algorithm. We are allowed to pick any two items, compare them, record the result, and choose our next step as a function of the result. The method does not need to have anything to do with the heap structure. Your lower bound must be valid for *all conceivable algorithms.*