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

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

---

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. 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 write your solutions to graduate problems on **separate sheets.**

7.1 (U,G) (8 points) *($k$-way merging.)* Give an $O(n \log k)$-time algorithm to merge $k$ sorted lists into one sorted list, where $n$ is the total number of elements in all the input lists. *Hint.* Use a heap for $k$-way merging.

7.2 (U,G) (a) (2 points) Let $\{a_n\}$, $\{b_n\}$, $\{c_n\}$ be sequences of real numbers. Prove: (a1) if $a_n = O(b_n)$ then $2a_n = O(b_n)$; (a2) if $a_n = O(b_n)$ and $|b_n| \le |c_n|$ then $a_n = O(c_n)$.

(b) (3 points) What is wrong with the following inductive proof of the obviously false statement $2^n = O(n)$?

"Proof by induction." *Base case:* $n = 1$: $2^1 = O(1)$ because $2^1 \le 2 \cdot 1$ $(c = 2)$. *Inductive hypothesis:* $2^n = O(n)$. *Desired conclusion:* $2^{n+1} = O(n+1)$. *Inductive step* (proof that the Desired Conclusion follows from the Inductive Hypothesis): $2^{n+1} = 2 \cdot 2^n$; by the inductive hypothesis this is $2 \cdot O(n)$; by (a1), this quantity is $O(n)$ (a constant factor is swallowed by the big-oh notation); finally any sequence that is $O(n)$ is also $O(n+1)$ (by (a2)). Combining these steps we obtain $2^{n+1} = O(n+1)$.

7.3 (U,G) (8 points) Consider the following subcase of the Knapsack problem: each of the $2n + 1$ input variables is a positive integer; each input variable is $\le W$ (the weight limit); and the integer $W$ has $O(\log n)$ digits. Prove that the dynamic programming algorithm given in class runs in polynomial time in this case.

Make sure you clearly state what you know about the bit-length of the input, highlight the pertinent information, and clearly state what quatity is polynomially bounded in terms of what quantity.