Algorithms – CS-27200/37000 Homework – January 24, 2005

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

ADVICE. Take advantage of the TA sessions.

CHANGE IN SCHEDULE: TA sessions are held in Ryerson-255, Tuesday and Thursday 5–6pm, Saturday 11am–noon, and (this is new) **Wednesday after class** 12:30–1:20 or 1:30–2:20 depending on demand. Indicate your interest in the Wednesday session to the instructor immediately after class. (The Wednesday evening sessions are discontinued.)

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 appropriate.

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.

READING. Graduate students: study Depth-First Search and its applications.

8.1 (U,G) (5 points) Given an undirected graph G by an array of adjacency lists, determine the degree of each vertex and sort the vertices by degree in linear time. Write a very simple pseudocode.

8.2 (U,G) (6 points) Recall that a digraph is *strongly connected* if every vertex is accessible from every vertex. Given a digraph $G = (V, E)$ by an array of adjacency lists, decide in linear time whether or not G is strongly connected. Your solution should be *very simple*, only 3 essential lines based on facts discussed in class or previously assigned as homework.

8.3 (U,G) (Due Friday, January 28) Let A and B be sets of integers. We define the set $A + B$ as $A + B = \{a + b : a \in A, b \in B\}$. For instance, if $A = \{2, 3, 5, 7\}$ and $B = \{1, 3, 4\}$ then $A + B = \{3, 4, 6, 7, 8, 9, 10, 11\}$. (Note that $6 \in A + B$ for three reasons: $6 = 1 + 5 = 3 + 3 = 4 + 2$.)

The *incidence vector* of a set $A \subseteq \{1, 2, \ldots, n\}$ is the $(0, 1)$-vector $v_A = (x_1, \ldots, x_n)$ where $x_i = 1$ if $i \in A$ and $x_i = 0$ if $i \notin A$.

Let the sets $A, B \subseteq \{1, 2, \ldots, n\}$ be given by their incidence vectors (treated as arrays). Compute the incidence vector of the set $A + B \subseteq \{1, 2, \ldots, 2n\}$. (This problem arose in machine vision.)

(a) (3 points) Write a simple pseudocode to do the computation in $O(n^2)$ steps. (Arithmetic with and copying numbers between 1 and $2n$ counts as a step.)

(b) (G only, 8 points) Solve the problem in $O(n^\alpha)$ steps where $\alpha = \log 3 \approx 1.58$. Describe your solution in English. Your solution should not be more than a short paragraph (with reference to a result proved in class).

(c) (G only, 0 points, do not hand in) Modify the solution to (b) so it will work in $O(n(\log n)^2)$ using Fast Fourier Transform (FFT). (Read about FFT.)

(d) (Open problem - possible research project) Solve the problem in $O(n)$. (To the instructor's knowledge, this is not known.)

8.4 (U,G) (Due Monday, February 7) Let $G = (V, E)$ be an undirected graph. Assume every vertex of G has degree ≤ 45. (The *degree* of a vertex is the number of its neighbors.) We wish to color the vertices red and blue (each vertex gets exactly one color) such that each vertex will have at most 22 neighbors of its own color. (Note that this is not a *legal* coloring in the sense of the definition of the chromatic number.) Show that this is always possible, using the following algorithm (given here in pseudocode).

procedure *Lovász-toggle*

1 Initialize by coloring each vertex arbitrarily
2 Call a vertex "bad" if it has more than 22 neighbors of its own color
3 BAD := set of bad vertices
4 **while** BAD $\neq \emptyset$
5 pick a bad vertex
6 recolor it
7 update BAD
8 **end**(**while**)

(a) (8 points) Prove that this algorithm will terminate in a finite number of steps. (Give a very simple and convincing argument, no more than 5 or 6 lines.) Give an upper bound on the number of cycles of the **while** loop in terms of the basic parameters $|V|, |E|$. *Hint.* Call the graph with a coloring a "configuration." With each configuration, associate an integer (the "potential") in such a way that each round of the Lovász-toggle reduces the potential. This will give a bound on the number of rounds. Note that "the number of bad vertices" is NOT an appropriate potential function: it can increase.

(b) Show that statement (a) becomes false if 45 is increased to 46 (but the number 22 remains unchanged). Construct graphs where each vertex has degree ≤ 46 and where

(i) (2 points) the algorithm never terminates, regardless of the initial coloring and the choice of bad vertex made in line 5;

(ii) (3 points) for some initial colorings and some choices of the bad vertex the algorithm will terminate, for others it will not.

(c) (Grad only; 5 points) Modify the above algorithm to achieve the following objective: each red vertex must have at most 25 red neighbors and each blue vertex must have at most 19 blue neighbors. Prove statements (a) and (b) above for the modified algorithm.

8.5 (Due Monday, February 14) CAR RACE PROBLEM. *The solution should be short,* **elegant***, and convincing.*

Let R be a subset of the $(n+1)^2$ points in the plane with integer coordinates between 0 and n. We call R the "race track." One of the points of R is designated as the start (S), another as the goal (G).

The points are represented as vectors (i, j). Cars are particles sitting on a point at any time. In one unit of time, a car can move from a point of R to another point of R, say from (i_1, j_1) to (i_2, j_2). The *speed vector* of the car during this time unit is defined as the vector $(i_2 - i_1, j_2 - j_1)$.

The *acceleration/deceleration* of the car is limited by the following constraint: from any one time unit to the next one, each coordinate of the speed vector can change by at most one.

For instance, if during time unit 6 the car was moving from point $(10, 13)$ to point $(16, 12)$ then its speed vector was $(6, -1)$ during this move; during the next time unit, the following are its possible speed vectors and corresponding destinations:

speed during time unit 7	destination at the end of time unit 7
$(7, 0)$	$(23, 12)$
$(7, -1)$	$(23, 11)$
$(7, -2)$	$(23, 10)$
$(6, 0)$	$(22, 12)$
$(6, -1)$	$(22, 11)$
$(6, -2)$	$(22, 10)$
$(5, 0)$	$(21, 12)$
$(5, -1)$	$(21, 11)$
$(5, -2)$	$(21, 10)$

Of course only those locations are legal which belong to R (the car cannot leave the race track).

During time unit 0, the car rests at Start with speed $(0, 0)$. The objective is to decide whether or not the Goal is reachable at all and if so, to reach it using the minimum number of time units.

(a) (15 points) Solve this problem in $O(|R| \cdot n^2)$ time. Describe your solution in clear English statements. Pseudocode not required. Algorithms discussed and analysed in class can be used as subroutines. Prove that your algorithm runs within the time claimed. *Hint.* Use BFS. The difficulty is in constructing the right graph to which to apply BFS. Do not overlook the fact that an optimal route of the car may visit the same location several times (at different speeds). (Construct an example where the optimal route visits the same point 100 times. Do not hand in the answer to this parenthetical, though enlightening, question.)

(b) (G only, 10 points) Solve the problem in $O(|R| \cdot n)$ time and space. (Note that you are not permitted to use an array with more than $O(|R| \cdot n)$ cells because of the space constraint.) (Hint: it is likely that you need only a minor modification of the algorithm you gave for (a) together with a more clever analysis.)