Algorithms – CS-37000 Homework 4 – January 12, 2006

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

ADVICE. Take advantage of the TA's problem sessions.

READING KT, 2.5 (Priority Queues), Karacuba–Ofman multiplication handout

REVIEW from Discrete Math: Graphs and Digraphs, Asymptotic notation. Review all previous handouts and readings.

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

This homework is due on **Tuesday, January 17** at the **beginning of the class** except problem 4.6 which is due Thursday, January 19.

Unless expressly stated otherwise, **all algorithms must be described in pseudocode**. IMPORTANT: **Define and explain your variables** and put comments on the lines of your pseudocode. (Otherwise your code will be unintelligible. You lose credit if understanding your solution requires unreasonable amount of work.)

All problems below have VERY SIMPLE solutions (just a few lines of pseudocode). ELEGANCE COUNTS!

For each problem below, state your answer in the form of an elegant pseudocode. Add definitions of your variables and brief comments explaining what is happening. No explanation of the correctness and the cost of your algorithms is required (as long as they are indeed correct and run in linear time).

4.1 (4 points) Implement a complete binary tree with n nodes as an array $A[1, \ldots, n]$. For each address $i = 1, \ldots, n$ you need to define how to compute parent(i), leftchild(i), rightchild(i) as very simple arithmetic operations (or "NIL" if the corresponding child does not exist). Hint. Number the nodes of a complete binary tree row-wise starting at the root; examine how the number of a node is related to its children and parent.

4.2 (8 points) *(k-way merging.)* Give an $O(n \log k)$-time algorithm to merge k sorted lists into a single sorted list, where n is the total number of elements in all the input lists. *Hint.* Use a heap for k-way merging.

4.3 (3+3 points) (a) Write simple pseudocode for DECREASE_KEY(H, x, r) where x is a node in the heap H and r is a real number (the new key). You need to replace key(x) by r if r is smaller. (b) Do the same for INCREASE_KEY(H, x, r). You need to replace key(x) by r if r is greater.

4.4 (6 points) A recursive algorithm reduces the the computation on an input of size n to the computation on 5 inputs of size $n/3$ each. Calculate the complexity of the algorithm (complexity = cost as a function of n). Ignore the cost of the reduction (the cost of computing the 5 smaller inputs out of the one larger input). You may assume that n is a power of 3. Inputs of size 1 require 1 unit of time.

4.5 (8 points) We are given an array of real numbers $x[1], \ldots, x[n]$. The sum of the interval $[i, j]$ is the quantity $S[i, j] := \sum_{k=i}^{j} x[k]$. Find the maximum interval sum $S_{\max}$. Find this value in *linear* time (i. e., the number of operations should be $O(n)$).

(The solution should be a very simple pseudocode, no more than a few lines. **Elegance counts.** *Hint*: dynamic programming.)

Note: you are not required to output the interval with the maximum sum, just the value of the maximum sum. Observe the following convention:

Convention. If $j < i$, we say that the interval $[i, j]$ is *empty*; the sum of the empty interval is zero. Empty intervals are admitted in the problem. Therefore $S_{\max} \geq 0$ even if all the $x[i]$ are negative.

4.6 (Due Thursday, January 19) 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) (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.)