Computer Science 238 November 8, 1995
Erindale Campus University of Toronto

Homework Assignment #4
Due: November 22, 1995 by 2:10pm

On the cover page of your assignment, you must write **and sign** *the following statement: "I have read and understood the policy on collaboration in homework stated in the Course Information handout." Without such a signed statement your homework will not be marked.*

---

1. (10 points) Recall that if $f, g : \mathbf{N} \to \mathbf{R}^+$ (where $\mathbf{R}^+$ is the set of nonnegative reals), then we say:

f is $O(g)$ if for some $c \in \mathbf{R}^+$ and some $n_0 \in \mathbf{N}$, $f(n) \leq cg(n)$ for all $n \geq n_0$;
f is $\Omega(g)$ if for some positive $c \in \mathbf{R}^+$ and some $n_0 \in \mathbf{N}$, $f(n) \geq cg(n)$ for all $n \geq n_0$;
f is $\Theta(g)$ if f is $O(g)$ and f is $\Omega(g)$.

i. Prove that for all such f and g, f is $\Omega(g) \Leftrightarrow g$ is $O(f)$.

ii. Prove that for all such f and g, $\mathrm{MAX}(f, g)$ is $\Theta(f + g)$. (Note that $\mathrm{MAX}(f, g)$ is that function which on input n, has value $\max(f(n), g(n))$. $f + g$ is that function which on input n has value $f(n) + g(n)$.)

iii. Prove or disprove: For all $f, g, h : \mathbf{N} \to \mathbf{R}^+$, if h is $O(f + g)$, then either h is $O(f)$ or h is $O(g)$.

2. (10 points) Consider the following recurrence, which is defined for n a power of 3:

$$T(n) \leq \begin{cases} 10 & \text{if } n = 1 \\ 25T(n/3) + n^2 \log_2 n & \text{if } n = 3^k \text{ for some } k > 0 \end{cases}$$

Prove that $T(n)$ is in $O(n^3)$, without using the theorem that gives a general solution to recurrences of this form (described in the text and a handout).
Hint: Prove that, for appropriately chosen constants $c, d > 0$, $T(n) \leq cn^{\log_3 25} - dn^2 \log_2 n$.

3. (10 points) Consider the following recurrence:

$$T(n) \leq \begin{cases} c_1 & \text{if } 3 \geq n \geq 1 \\ T(\lceil \frac{3}{4}n \rceil) + T(\lceil \frac{1}{5}n \rceil) + c_2 n & \text{if } n > 3 \end{cases}$$

Prove that $T(n)$ is $O(n)$.

Motivation: This recurrence actually comes up in the following situation. Say we wish to find the median of n distinct elements, using as few comparisons as possible. More generally, say we wish to find the k-th smallest of n elements. We could sort the n numbers but this takes about $n \log n$ comparisons. Instead we do the following:

Divide the elements up into groups of 5 (don't worry now about what to do if n isn't divisible by 5); find the median of each group of 5, using a *linear* in n number of comparisons in total (exactly how you find the median of 5 points isn't important). This gives us about $n/5$ median points. *Recursively* find the median of these median points using about $T(n/5)$ comparisons; call this element b. Using a *linear* in n number of comparisons, compare every element to b; let S_1 be the set of elements $\leq b$, and let S_2 be the set of elements $> b$. Because of the way b was chosen, each of S_1 and S_2 must have at most $\frac{3}{4}n$ elements. If $k \leq |S_1|$, then *recursively* find the k-th smallest element of S_1, otherwise *recursively* find the $(k - |S_1|)$-th smallest element of S_2; this takes at most $T(\frac{3}{4}n)$ comparisons.

4. (10 points) The point of this question is to develop an algorithm which, given an array of $n \geq 3$ distinct numbers, finds the three smallest numbers (in order). The obvious algorithm finds the smallest element by making $n-1$ comparisons, then finds the smallest of the remaining elements by making $n-2$ comparisons, then finds the smallest of the rest by making $n-3$ comparisons; this algorithm makes $3n-6$ comparisons in total, and we would like to do better. For example, it is possible to find the three smallest of four numbers by making at most 5 comparisons, as follows: say the numbers are a, b, c, d; compare a with b and compare c with d; assume (by symmetry) that $a < b$ and $c < d$; next, compare a with c, and assume (by symmetry) that $a < c$; next, compare b with c; if $b < c$ then the three smallest elements (in order) are a, b, c; if $c < b$, then compare b with d; if $b < d$, then the three smallest elements (in order) are a, c, b, otherwise they are a, c, d.

i. Define the function h on the set $\{3, 4, 5\}$ by $h(3) = 3$, $h(4) = 5$, $h(5) = 7$. We have shown above that it is possible to find the three smallest (in order) of 4 elements using at most $h(4)$ comparisons. Show that it is possible to find the three smallest (in order) of 3 elements using at most $h(3)$ comparisons. Show that it is possible to find the three smallest (in order) of 5 elements using at most $h(5)$ comparisons. (Partial credit will be given for doing this last part in 8 comparisons.)

ii. Give a divide and conquer algorithm which, given an array of n distinct elements ($n \geq 3$), finds the three smallest elements (in order). If $C(n)$ is the maximum possible number of comparisons that your algorithm makes (when working on n input elements), then C should satisfy:

$$C(i) \leq h(i) \text{ for } i \in \{3, 4, 5\};$$
$$C(n) \leq C(\lfloor n/2 \rfloor) + C(\lceil n/2 \rceil) + 3 \text{ for all } n \geq 6.$$

iii. Let C be a function which satisfies the inequalities in part (ii) above. Prove that for all $n \geq 3$, $C(n) \leq 2n - 3$.