Assignment Written 1

Jonathan Buss, Mark Petrick, Masoomeh Rudafshani

Due Sept. 26, by 11:59pm, to Crowdmark.
All submitted work must be the student's own.

Question 1 (12 marks).

Let f and g be two functions defined on the set of positive reals. Assume that each is monotonically nondecreasing: i.e., for all $m < n$ it holds that $f(m) \le f(n)$ and $g(m) \le g(n)$. Also assume that $f \in O(g)$ but $g \notin O(f)$.

For each of the following pairs of functions F and G, determine whether $F \in O(G)$ and whether $G \in O(F)$. In each case, if the answer is always "yes" or is always "no", then give a proof. If the answer depends on what the actual f and g are, then give explicit functions for each.

(a) $F(n) = f(n+1)$; $G(n) = g(n)$.

(b) $F(n) = \log_2 f(n)$; $G(n) = \log_3 g(n)$.

(c) $F(n) = f(n)^2$; $G(n) = g(n)^2$.

(d) $F(n) = f(\sqrt{n})$; $G(n) = g(\sqrt{n})$.

Question 2 (12 marks).

Consider the function T defined over the positive integers by

$$T(n) = \begin{cases} \lceil\sqrt{n}\rceil \cdot T(\lceil\sqrt{n}\rceil) + dn & \text{if } n > 2 \\ dn & \text{if } n \le 2 \end{cases},$$

where d is a fixed constant. For purpose of simplifying the analysis, we also consider the function T' defined over the positive reals, that satisfies the recurrence

$$T'(n) = \begin{cases} \sqrt{n} \cdot T'(\sqrt{n}) + dn & \text{if } n > 2 \\ dn & \text{if } n \le 2 \end{cases}.$$

It should be clear that $T'(n) \le T(n)$ when n is an integer; the relationship in the opposite direction may not be so clear.

(a) For $n > 2$, determine the number of iterations of the recurrence needed to determine a value of $T'(n)$; that is, the number of terms in the sequence n, $\sqrt{n}$, $\sqrt{\sqrt{n}}, \ldots$ required to get a value at most 2.

Note that even when n is an arbitrary real $n \ge 2$, the length of the sequence is an integer. If you wish, you may start by considering the cases in which each element of the sequence is an integer.

(b) Using your answer to the previous part, give an explicit formula for a function f such that $T'(n) = f(n)$. Prove it is correct, using a recursion tree, substitution, or other suitable method.

(c) Prove that the solution to the recurrence for T (with the $\lceil\cdot\rceil$s) satisfies $T(n) \in O(f)$ (or, equivalently, $T(n) \in O(T'(n))$).

Question 3 (12 marks).

Consider the following problem. An input is an $n \times n$ array w`[][]` of numerical values. We consider a value $w[i][j]$ to be a "neighbour" of the values at the four locations $w[i][j \pm 1]$ and $w[i \pm 1][j]$ (if any of the four does not exist, there are simply fewer neighbours). A solution to the problem is any i and j such that $w[i][j]$ is less than or equal to each of its neighbours.

(*If you wish, you may consider this as an undirected graph on the* n^2 *vertices* $\{\langle i, j\rangle \mid 1 \leq i, j \leq n\}$, *each having an edge to each of its neighbours. Alternatively, you may simply use the input array, and ignore graph theory.*)

Using a divide-and-conquer approach, design an algorithm that finds a solution after examining at most $O(n)$ elements of the array. (That is, much fewer than the total number n^2.)

Carefully explain why your algorithm is correct – why it always finds a solution – and prove that it examines at most $O(n)$ locations of the array.

Question 4 (12 marks).

A manufacturer of computer chips has a "test jig" which allows two chips to test one another for proper behaviour. Each chip can be either "good" or "bad"; when two chips are put into the test jig, each reports on the status of the other.

- A chip which is actually good correctly reports the status of any other tested with it.
- A chip which is actually bad may make any report. In particular, it may conspire with the other bad chips to be as confusing as possible.

Consider that you have n chips, each either good or bad, and would like to know which is which. Since n is large, you would rather not test every possible pair.

(a) Suppose that half or more of the chips are bad. Show that one cannot guarantee to identify even a single good chip using the jig, even if one performs all pairwise tests. (Hint: describe how the bad chips can provide test results, so that one or more of the bad chips is indistinguishable from the good ones.)

(b) Now suppose that it is guaranteed that more than half of the n chips are good.

Give a divide-and-conquer algorithm to identify any one good chip using $O(n)$ tests. Justify that it works correctly, in any case in which the guarantee holds.

(Your algorithm should perform zero or more tests, and from their results produce one or more sub-problem(s) of smaller size. After solving each sub-problem recursively, further tests may be used to determine the final outcome.)

(c) Give a recurrence relation for the number of tests used by your algorithm, in the worst case. Prove (using any suitable method) that its solution is in $O(n)$. For full marks, identify the constant.