BU CLA CS 530: Analysis of Algorithms
Spring 1999

Selected Solutions for Problem Set 9

Problem 1.

Part (a) is Exercise 28.2-2, page 641, in [CLR], and part (b) is a generalization. We prove both parts simultaneously.

The argument is like the proof of the 0-1 principle. Consider the input permutation $\langle a_1, a_2, \ldots, a_n \rangle$. For each $k = 1, 2, , \ldots, n$, define a monotonically increasing function f_k as:

$$f_k = \begin{cases} 0 & \text{, if } x \leqslant a_k \\ 1 & \text{, if } x > a_k \end{cases}$$

For each f_k except the one that is all 0, test the network on the 0-1 sequence $\langle f_k(a_1), f_k(a_2), \ldots, f_k(a_n) \rangle$. If the network fails to sort the input permutation $\langle a_1, a_2, \ldots, a_n \rangle$, it will fail to sort at least one of the $n-1$ test sequences, by the following argument.

If the sort fails, there are input elements a_i and a_j such that $a_i < a_j$, but the network places a_j before a_i in the output sequence. Because f_i is monotonic, the network places $f_i(a_j)$ before $f_i(a_i)$ in the output sequence when $\langle f_i(a_1), f_i(a_2), \ldots, f_i(a_n) \rangle$ is the input. But $f_i(a_j) = 1$ and $f_i(a_i) = 0$, which means the network fails to sort the 0-1 sequence $\langle f_i(a_1), f_i(a_2), \ldots, f_i(a_n) \rangle$ correctly.

(The preceding argument is similar to the proof of the 0-1 principle).

Hence, if the network sorts all of the $n-1$ test sequences, it sorts the input permutation.

Problem 2. (Exercise 28.3-2, p. 645, in [CLR])

A bitonic-sorter consists of a half-cleaner (which contains $n/2$ comparators) and 2 bitonic-sorters each working on $n/2$ inputs. This creates the following recurrence:

$$T(n) = \begin{cases} 1 & \text{if } n = 2, \\ 2 \cdot T(n/2) + (n/2) & \text{if } n = 2^k \text{ and } k \geqslant 2. \end{cases}$$

Using the Master Theorem method in Chapter 4 of [CLR], this matches case 2: If $f(n) = n/2 = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$. In this case $b = 2$ and $a = 2$. Hence, the solution is: $T(n) = \Theta(n \log n)$.

Problem 3. (Exercise 28.5-1, p. 650, in [CLR])

MERGER[n] is essentially a BITONIC-SORTER[n] (the difference being that the left-most HALF-CLEANER[n] is modified). From Exercise 28.3-2 (for which a solution was sketched in lecture), the size of MERGER[n] is therefore $\Theta(n \log n)$. For the size of SORTER[n] we need to solve the following recurrence:

$$T(n) = \begin{cases} 1 & \text{if } n = 2, \\ 2 \cdot T(n/2) + \Theta(n \log n) & \text{if } n = 2^k \text{ and } k \geqslant 2. \end{cases}$$

This recurrence can be solved by iteration to obtain:

$$T(n) = \Theta(n(\log n)^2).$$

Problem 4. (Exercise 28.5-2, p. 650, in [CLR])
The depth of $\text{S}\textsc{orter}[n]$ is given by:

$$\log 2 + \log 4 + \log 8 + ... + \log n = 1 + 2 + 3 + ... + \log n.$$

(see Figure 28.12 (c) for the case of n = 8, page 649 in [CLR]). This summation adds up to: $(\log n)(1 + \log n)/2$.

Assaf Kfoury
Created: 99.03.20