CPSC-320 Intermediate Algorithm Design and Analysis Winter 2000

Practice Homework # 2

These practice problems will be discussed at the tutorials during the week of September 25. To get the most out of the tutorial, start to work on the problems beforehand, and come with lots of questions!

1. Following is a list of statements about non-negative functions $f(n)$ and $g(n)$ over the positive integers. For each, state whether it is true or false and explain your answer. If a statement is true, you should justify it in a manner similar to that used to prove the claims in lectures 2 and 3. If a statement is false, you need to provide examples of two specific functions $f(n)$ and $g(n)$ for which the statement is false. You can assume that all logarithms are to the base 2.

   (a) $f(n) = cg(n) + o(g(n))$ for some constant $c \Rightarrow f(n) = O(g(n))$.
   (b) $\log f(n) = O(\log g(n)) \Rightarrow f(n) = O(g(n))$.
   (c) $2^{\log n} = O(n^2)$.

2. Give examples of nonnegative functions $f(n)$ and $g(n)$ satisfying the following relationships (you should use a different example for each relationship).

   (a) $f(n) \neq g(n)$ but $f(n) = g(n) + o(g(n))$.
   (b) $f(n) = O(g(n))$ but $f(n) \neq \Theta(g(n))$.
   (c) $f(n) = O(g(n))$ but $2^{f(n)} \neq O(2^{g(n)})$.

3. A *run of 0's* in a binary string s is a substring s' consisting only of 0's, such that each end of the substring is either adjacent to a 1 or is also the end of the whole string s. For example, the string 101100 has two runs of 0's, one of length 1 and one of length 2 and the string 000 has one run of 0's of length 3.

   Let S be the set of all binary strings of length n where $n \geq 1$ (note that there are 2^n strings in S). As a function of n, what is the total number of runs of 0's, summed over all strings in S?

4. Let $x = x_n x_{n-1} \ldots x_1$ and $y = y_n y_{n-1} \ldots y_1$. Here is a recursive algorithm for multiplying two n-bit numbers, due to Karatsuba. Suppose that n is even, $n > 1$ (when $n = 1$, there is a simple base case to do the multiplication).

   Let $x_L = x_{n/2} x_{n/2-1} \ldots x_1$ and let $x_H = x_n x_{n-1} \ldots x_{n/2+1}$. For example, if $x = 1101$ then $X_L = 01$ and $x_H = 11$. Define y_L and y_H similarly. Note that $x = x_L + x_H 2^{n/2}$ and $y = y_L + y_H 2^{n/2}$ Also, the product of x and y can be written as

$$\begin{aligned} xy &= x_L y_L + (x_H y_L + x_L y_H) 2^{n/2} + x_H y_H 2^n \\ &= z_0 + z_1 2^{n/2} + z_2 2^n, \qquad (*) \end{aligned}$$

where $z_0 = x_L y_L, z_1 = x_H y_L + x_L y_H$, and $z_2 = x_H y_H$.

The multiplication algorithm first uses addition to compute $x_L + x_H$ and $y_L + y_H$. Then it calls itself recursively three times on pairs of inputs of size at most $n/2 + 1$ to compute the three products $t = (x_L + x_H)(y_L + y_H), z_0 = x_L y_L$, and $z_2 = x_H y_H$.

When the recursive calls have completed, it computes $z_1 = t - z_0 - z_2$. Finally, using two more additions and two shifts, the product xy can be computed from z_0, z_1, and z_2 using (*). This completes description of the algorithm.

(a) Let $M(n)$ be the running time of this algorithm on two n-bit numbers. If additions and shifts on n-bit numbers can be done in $O(n)$ time, show that when n is even, $M(n)$ satisfies the following:

$$M(n) \leq 3M(n/2 + 1) + O(n).$$

(b) Consider the following almost-identical, but simpler recurrence:

$$M(n) = 3M(n/2) + O(n), \ n \text{ a power of 2}; \quad M(1) = 0.$$

Solve this recurrence using the iteration method, to get a bound on $M(n)$. Note: in using the iteration method, you will get a summation that is called a "geometric series;" you can refer to homework 1 to get a closed form expression for your summation.

You should find that this multiplication algorithm is $o(n^2)$.