CS 173: Discrete Structures, Fall 2009
Homework 8

This homework contains 3 problems worth a total of 30 regular points. It is due on Friday, October 30 at noon. Put your homework in the appropriate dropbox in the Siebel basement.

1. **More on recurrences [10 points]**

   In the discussion of Karatsuba's algorithm for multiplying integers (end of lecture 25), I left out some details of the analysis. Let's fill in some of them:

   (a) I claimed that if T has the following recurrence (where c and d are constants)

   $$T(1) = c$$
   $$T(n) = 4T(n/2) + dn$$

   then T is $O(n^2)$. Show that this is true by unrolling the recurrence, assuming that n is a power of two.

   (b) I claimed that $n(\frac{3}{2})^{log_2 n}$ is $O(n^{log_2 3})$. Show that this is correct. Hint: use algebra and standard properties of logs and exponentials.

2. **Algorithm analysis [10 points]**

   Consider the following mystery function:

   1. foo ($a_1, a_2, \ldots a_n$: real numbers)
      2. D = $|a_1 - a_2|$
      3. for $x = 1$ to n
         4. for $y = 1$ to n
            5. Q = $|a_x - a_y|$
            6. if ($x \neq y$ and $Q < D$) D = Q

      return D

   (a) Give a brief English description of what the function foo computes.

   (b) How many times are lines 5 and 6 executed, as a function of n?

   (c) What is the big-O running time of this algorithm? Briefly justify your answer.

   (d) This is a really bad algorithm for this task. Write pseudocode for a function with a better big-O running time. Hint: your new function can call any of the standard algorithms we've seen in lecture e.g. binary search.

   (e) What is the big-O running time of your new function from part d? Hint: if your new function called a standard library function, you need to include its cost in your big-O analysis. For example, a call to binary search requires $O(\log n)$ time, where n is the size of the array you're feeding to binary search.

3. **Son of algorithm analysis [10 points]**

A particularly irritating student in the beginning programming class submitted the following function. (The max function returns the largest of the numbers given to it.)

```
bar(a_1, a_2, ... a_n: real numbers)
    if (n = 1) then return 0
    else
        L = bar(a_2,a_3,...,a_n)
        R = bar(a_1,a_2,...,a_{n-1})
        Q = |a_1 - a_n|
        return max(L,R,Q)
```

(a) Give a succinct English description of what bar computes.

(b) Suppose that $T(n)$ is the running time for bar on the input n. Write a recurrence (with initial condition!) for $T(n)$.

(c) Draw a recursion tree for $T(n)$.

(d) Use the tree to derive a big-O solution for $T(n)$.

(e) Briefly explain why this isn't a good design and how you would write a more efficient algorithm for this task.