CSCE629 Analysis of Algorithms

Homework 2

Texas A&M U, Fall 2019 — *09/06/19*

Lecturer: Fang Song — *Due: 10am, 09/13/19*

Instructions.

- Typeset your submission by LaTeX, and submit in PDF format. Your solutions will be graded on *correctness* and *clarity*. You should only submit work that you believe to be correct, and you will get significantly more partial credit if you clearly identify the gap(s) in your solution. You may opt for the "I'll take 15%" option (details in Syllabus).
- You may collaborate with others on this problem set. However, you must **write up your own solutions** and **list your collaborators and any external sources** for each problem. Be ready to explain your solutions orally to a course staff if asked.
- For problems that require you to provide an algorithm, you must give a precise description of the algorithm, together with a proof of correctness and an analysis of its running time. You may use algorithms from class as subroutines. You may also use any facts that we proved in class or from the book.

This assignment contains 5 questions, 5 pages for the total of 70 points and 15 bonus points. A random subset of the problems will be graded.

1. (More asymptotics)
   (a) (5 points) Prove that for any positive numbers $a, b > 0$, we have $n^b = \omega(\log^a(n))$. There are several ways to approach this. One way is to use L'Hospital's rule for evaluating limits together with the following fact: $f(n) = \omega(g(n))$ if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$.
   (b) (5 points) Prove or disprove: If $h(n) = \lceil n \log(n) \rceil$, then $n = \Theta(h(n) / \log h(n))$.

2. (Recurrence) Solve the following recurrences.

   (a) (5 points) $A(n) = 2A(n/4) + \sqrt{n}$

   (b) (5 points) $B(n) = 2B(n/4) + n$

   (c) (5 points) $C(n) = 3C(n/3) + n^2$

   (d) (5 points (bonus)) $D(n) = \sqrt{n}D(\sqrt{n}) + n$

3. (10 points) (Domain transformation) We analyzed the running time of Mergesort by the recurrence $T(n) = 2T(n/2) + O(n)$. The actual Mergesort recurrence is somewhat messier:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + O(n)\,.$$

We'll justify in this problem that ignoring the ceilings and floors in a recurrence is okay afterall using a technique called *domain transformation*.

First, because we are deriving an upper bound, we can safely overestimate $T(n)$, once by pretending that the two subproblem sizes are equal, and again to eliminate the ceiling:

$$T(n) \leq 2T(\lceil n/2 \rceil) + n \leq 2T(n/2 + 1) + n\,.$$

Second, we define a new function $S(n) = T(n + \alpha)$ for some α. **You are to complete the second step.** Show that you can find a nice α so that $S(n) \leq 2S(n/2) + O(n)$ does hold, and conclude from there that $T(n) = O(n \log n)$.

[Exercise (Do not turn in). Show how to remove floors by similar arguments.]

4. (Quicksort) We were not precise about the running time of Quicksort in class (for a good reason). We will give some case studies in this problem (and appreciate the subtlety).

   (a) (10 points) Given an input array of n elements, suppose we are unlucky and the partitioning routine produces one subproblem with $n-1$ elements and one with 0 element. Write down the recurrence and solve it. Describe an input array that costs this amount of running time to get sorted by Quicksort.

   (b) (5 points) Now suppose that the partitioning always produces a 9-to-1 proportional split. Write down the recurrence for $T(n)$ and solve it.

   (c) (5 points) What is the running time of Quicksort when all elements of the input array have the same value?

5. (Sumerians' multiplication algorithm) The clay tablets discovered in Sumer led some scholars to conjecture that ancient Sumerians performed multiplication by reduction to *squaring*, using an identity like

$$x \cdot y = (x^2 + y^2 - (x - y)^2)/2\,.$$

In this problem, we will investigate how to actually square large numbers.

(a) (7 points) Describe a variant of Karatsuba's algorithm that squares any n-digit number in $O(n^{\log 3})$ time, by reducing to squaring three $\lceil n/2 \rceil$-digit numbers. (Karatsuba actually did this in 1960.)

(b) (8 points) Describe a recursive algorithm that squares any n-digit number in $O(n^{\log_3 6})$ time, by reducing to squaring six $\lceil n/3 \rceil$-digit numbers.

(c) (10 points (bonus)) Describe a recursive algorithm that squares any n-digit number in $O(n^{log_3 5})$ time, by reducing to squaring only five $(n/3 + O(1))$-digit numbers. [Hint: What is $(a + b + c)^2 + (a - b + c)^2$?]