CSE 331: Introduction to Algorithm Analysis and Design Fall 2012

HOMEWORK 3

Due Friday, September 28, 2012 by 1:15pm in class

IMPORTANT: Please submit each problem separately, i.e. each problem should begin on a new page and only the pages for one problem should be stapled together. Failure to do so might result in some problem(s) not being graded.

For general homework policies and our suggestions, please see the policy document.

For this homework, you can assume that addition, subtraction, multiplication and division of two numbers can be done in $O(1)$ time. (Side question: Is this assumption justified?)

---

1. (25+15 = 40 points) We will consider sorting in this problem. Recall that given n integers $a_1, \ldots, a_n$, its (ascending) sorted order are the numbers from the smallest to the largest. So for example, given

   $$3, 17, 1, -9, 100, 51, 1$$

   the sorted order is

   $$-9, 1, 1, 3, 17, 51, 100.$$

   (Note that the numbers need not be distinct.) Consider the following algorithm for sorting n numbers that are assumed be given as an array $A[1..n]$:

   (a) For $1 \leq i \leq n$ do:

   i. For $j = n$ down to $i+1$

   A. If $A[j] < A[j-1]$ then

   B. Swap the values in $A[j-1]$ and $A[j]$.

   Convince yourself that the algorithm above does compute the sorted order of the numbers $A[1], \ldots, A[n]$. (You do *not* have to include the proof of correctness in your submission.) Consider the following questions related to the run time of the algorithm:

   - Figure out a $f(n)$ such that the running time of the algorithm above is $O(f(n))$. Prove your bound on the running time.
   - Figure out a $g(n)$ such that the running time of the algorithm above is $\Omega(g(n))$. Prove your bound on the running time.

   (*Hint:* You can pick $g(n)$ and $f(n)$ such that $g(n)$ is $\Theta(f(n))$.)

2. (30 + 15 = 45 points) Polynomials are formal objects that even though pretty abstract have a lot of applications in practice. For example, the "error correcting code" that allows for CDs and DVDs to play even with scratches at its mathematical core is based on properties of polynomials. In this problem we will consider a fundamental operation on polynomials called *polynomial evaluation*.

   Recall that a polynomial of degree $n-1$ is defined by n coefficients $a_0, \ldots, a_{n-1}$ and is defined as the formal sum in some variable X as follows: $\sum_{i=0}^{n-1} a_i \cdot X^i$. For example, the polynomial corresponding to $a_0 = 1, a_1 = 0.2, a_2 = 100, a_3 = .5$ is $.5X^3 + 100X^2 + .2X + 1$.

For concreteness, given n real numbers $a_0, \ldots, a_{n-1}$, let us denote the polynomial $\sum_{i=0}^{n-1} a_i \cdot X^i$ by $P_{a_0,\ldots,a_{n-1}}(X)$. Now the evaluation of the polynomial $P_{a_0,\ldots,a_{n-1}}$ at a real number y is obtained from the formal sum by replacing X with y, i.e. $P_{a_0,\ldots,a_{n-1}}(y) = \sum_{i=0}^{n-1} a_i \cdot y^i$. For example, for the polynomial considered above when evaluated at 2 gives

$$P_{1,.2,100,.5}(2) = .5 \times 2^3 + 100 \times 2^2 + .2 \times 2 + 1 = 405.4$$

Consider the following algorithm for polynomial evaluation:

/* Input: Real numbers $a_0, \ldots, a_{n-1}$ and y */

(a) `eval` $\leftarrow 0$.

(b) For $i = 0 \cdots n-1$

  i. `temp` $\leftarrow 1$.

  ii. For $j = 1 \cdots i$

    - `temp` $\leftarrow$ `temp` $\cdot\, y$

  iii. `eval` $\leftarrow$ `eval` $+ a_i \cdot$ `temp`

(c) Return `eval`

Convince yourself that the algorithm above does indeed output $P_{a_0,\ldots,a_{n-1}}(y)$. Now consider the following questions:

- Prove that the algorithm above has a running time of $\Theta(n^2)$.
- The polynomial fairy tells you that there is a faster algorithm for the polynomial evaluation problem than the one above.[1] Design an algorithm with a running time of $O(f(n))$ such that $\lim_{n\to\infty} \frac{f(n)}{n^2} = 0$. You will receive more credit the (asymptotically) smaller your function $f(n)$ is.

3. $(7 + 8 = 15$ points) Exercise 8 in Chapter 2.

[1] The polynomial fairy never lies.