CSOR W4231 – Spring, 2018

hw1

Out: Monday, Jan 22, 2018
Due: 8pm, Monday, Feb 5, 2018

Please keep your answers clear and concise. For all algorithms **you** *suggest, you must prove correctness, give the best upper bound that you can for the running time. You should always describe your algorithm clearly in English* **and** *give pseudocode.*

You should write up the solutions entirely on your own. Collaboration with your classmates is limited to discussion of ideas only. You should list your collaborators on your write-up. If you do not type your solutions, make sure that your hand-writing is legible, your scan is high-quality and your name is clearly written on your homework.

1. (25 points) In the table below, indicate the relationship between functions f and g for each pair (f, g) by writing "yes" or "no" in each box. For example, if $f = O(g)$ then write "yes" in the first box. Here $\log^b x = (\log_2 x)^b$.

f	g	O	o	Ω	ω	Θ
$\log^2 n$	$6 \log n$					
$\sqrt{\log n}$	$(\log \log n)^3$					
$4n \log n$	$n \log 4n$					
$n^{3/5}$	$\sqrt{n} \log n$					
$5\sqrt{n} + \log n$	$2\sqrt{n}$					
$n^5 4^n$	5^n					
$\sqrt{n} 2^n$	$2^{n/2+\log n}$					
$n \log 2n$	$\frac{n^2}{\log n}$					
$n!$	2^n					
$\log n!$	$\log n^n$					

2. (20 points) Suppose we want to evaluate the polynomial $p(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ at point x.

   (a) Show that the following simple routine, known as *Horner's rule*, solves this problem and stores the solution in z.

   $z = a_n$
   for $i = n - 1$ down to 0 **do**
   $\quad z = zx + a_i$
   end for

   (b) How many additions and multiplications does this routine use, as a function of n? Can you find a polynomial for which an alternative method is substantially better?

3. (20 points) The Hadamard matrices $H_0, H_1, H_2, \ldots$ are defined as follows:

   - H_0 is the 1×1 matrix $\begin{bmatrix} 1 \end{bmatrix}$
   - For $k > 0$, H_k is the $2^k \times 2^k$ matrix

   $$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix}$$

   Let ν be a column vector of length $n = 2^k$. Can you compute the matrix-vector product $H_k\nu$ faster than the straightforward algorithm that requires $O(n^2)$ time? (Assume that all the numbers involved are small enough that basic arithmetic operations like addition and multiplication take unit time.)

4. (30 points) An array A with n entries is said to have a *majority* element if more than half of its entries are the same. Given A, we want to find such a majority element, if one exists. A question of the form *"Is $A[i] = A[j]$?"* can be answered in constant time; however questions of the form *"Is $A[i] > A[j]$?"* are **not** permitted (for example, the elements of the array could be images so there is no order among them).

   (a) (10 points) Show how to solve this problem in $O(n \log n)$ time.

   (b) (20 points) Now give a deterministic linear-time algorithm for this problem.

5. (35 points) The Fibonacci numbers are defined by the recurrence

   $$F_n = \begin{cases} 0, & \text{if } n = 0 \\ 1, & \text{if } n = 1 \\ F_{n-1} + F_{n-2}, & \text{if } n \geq 2 \end{cases}$$

   (a) (4 points) Show that $F_n \geq 2^{n/2}$, $n \geq 6$.

(b) Assume that the cost of adding, subtracting, or multiplying two integers is $O(1)$, independent of the size of the integers.

- (4 points) Write pseudocode for an algorithm that computes F_n based on the recursive definition above. Develop a recurrence for the running time of your algorithm and give an asymptotic lower bound for it.
- (4 points) Write pseudocode for a non-recursive algorithm that asymptotically performs fewer additions than the recursive algorithm. Discuss the running time of the new algorithm.
- (12 points) Show how to compute F_n in $O(\log n)$ time using only integer additions and multiplications.

  (Hint: Express F_n in matrix notation and consider the matrix

  $$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

  and its powers.)

(c) (11 points) Now assume that adding two m-bit integers requires $\Theta(m)$ time and that multiplying two m-bit integers requires $\Theta(m^2)$ time. What is the running time of the three algorithms under this more reasonable cost measure for the elementary arithmetic operations?

RECOMMENDED EXERCISES (do NOT return, they will not be graded)

1. Give tight asymptotic bounds for the following recurrences.

   - $T(n) = 4T(n/2) + n^3 - 1$.
   - $T(n) = 8T(n/2) + n^2$.
   - $T(n) = 6T(n/3) + n$.
   - $T(n) = T(\sqrt{n}) + 1$.

2. Show that, if λ is a positive real number, then $f(n) = 1 + \lambda + \lambda^2 + \ldots + \lambda^n$ is

   (a) $\Theta(1)$ if $\lambda < 1$.

   (b) $\Theta(n)$ if $\lambda = 1$.

   (c) $\Theta(\lambda^n)$ if $\lambda > 1$.

   Therefore, in big-Θ notation, the sum of a geometric series is simply the first term if the series is strictly decreasing, the last term if the series is strictly increasing, or the number of terms if the series is unchanging.

3. A positive integer N is a *power* if it is of the form q^k, where q, k are positive integers and $k > 1$.

   (a) Design and analyze an efficient algorithm that takes as input an integer N and determines if it is a *square*, that is, if it can be written as q^2 for some positive integer q.

   (b) Suppose that $N = q^k$, where N, k, q are positive integers. Show that either $N = 1$ or $k \le \log N$.

   (c) Design and analyze an efficient algorithm for determining whether a positive integer N is a *power*.