CSOR W4246 – Fall, 2015

HW1 – Theoretical part

Out: Thursday, September 17, 2015
Due: 8pm, Monday, September 28, 2015

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

You are encouraged to brainstorm with a small number of your classmates. However collaboration is limited to discussion of ideas only and you should write up the solutions entirely on your own. You should list your collaborators on your write-up. If you do not type your solutions, be sure that your hand-writing is legible, your scan is high-quality (if applicable) and your name is clearly written on your homework.

If you submit your assignment electronically, **please submit a .pdf file**. **Other file formats** *(such as .jpg, .doc, .c, or .py)* **will not be graded, and will automatically receive a score of 0.**

1. (10 points)
   (a) Find (with proof) a function f_1 such that $f_1(2n)$ is $O(f_1(n))$.
   (b) Find (with proof) a function f_2 such that $f_2(2n)$ is not $O(f_2(n))$.

2. (20 points) Use the Master theorem to give tight asymptotic bounds for the following recurrences.
   - $T(n) = 4T(n/2) + n^2$.
   - $T(n) = 8T(n/2) + n^3$.
   - $T(n) = 11T(n/4) + n^2$.
   - $T(n) = 7T(n/3) + n$.

3. (20 points) Consider the following recursive algorithm. On input a list of distinct numbers, the algorithm runs in three phases. In the first phase, the first 2/3 elements of the list are sorted (recursively). In the second phase, the last 2/3 elements are sorted (recursively). Finally, in the third phase, the first 2/3 elements are sorted (recursively) again. Derive a recurrence describing the running time of the algorithm and use the recurrence to bound its asymptotic running time. Would you use this algorithm in your next application?

4. (30 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.

f	g	O	o	Ω	ω	Θ
$10\log n$	$\log^3 n$					
$n\log(2n)$	$n\log n$					
$\sqrt{\log n}$	$\log\log n$					
$10n^2 + \log n$	$n^2 + 11\log^3 n$					
$\sqrt{n} + \log n$	$n^{2/3} + 10$					
$n^2 2^n$	3^n					
$n^{1/3}$	$(\log n)^2$					
$n\log n$	$\frac{n^2}{\log n}$					
$n!$	n^n					
$\log n!$	$\log n^n$					

5. (**OPTIONAL — EXTRA CREDIT**, 30 points)

The Fibonacci numbers are defined by the recurrence

$$F_0 = 0, F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}, n \geq 2$$

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

- 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, using the fact that $F_n \geq 2^{n/2}$, for $n \geq 6$.
- Write pseudocode for a non-recursive algorithm that asymptotically performs fewer additions than the recursive algorithm. Discuss the running time of the new algorithm.
- Show how to compute F_n in $O(\log n)$ time using only integer additions and multiplications. Provide an upper bound for the running time of this algorithm.
  (Hint: Express F_n in matrix notation as follows

  $$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix}.$$

  Consider how you may use powers of the matrix $\left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ in the equation above to compute F_n.)

(b) 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?