Assignement 2, COSC 3P03, Algorithms, Winter, 2016

Due: Feb. 10, Wed., 5:00 PM.

1. (10) The running time of an algorithm A is described by the recurrence $t(n) = 7t(n/2) + n^2$. A competing algorithm A' has a running time of $T(n) = aT(n/4) + n^2$. What is the largest integer value for a such that A' is asymptotically faster than A?

2. (40) For Fibonacci numbers defined as follows:

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n) &= f(n-1) + f(n-2),\ n \geq 2 \end{aligned}$$

   (a) Prove by induction that for any $n \geq 0$,

$$\begin{pmatrix} f(n) \\ f(n+1) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

   (b) Using the recursive definition of $f(n)$, design and implement a recursive algorithm that computes $f(n)$; Then using the just proved formula to design and implement a recusive algorithm that computes $f(n)$ in $O(\log n)$ time. Compare the running times (real or asymptotic) of these two algorithms.

3. (10) Given n arbitrary numbers, how can you use our linear-time selection algorithm to find the k smallest numbers in $O(n)$ time? For example, if we have 4, 3, 3, 9, 10, 2, 3, then the two smallest numbers are 2 and 3 (or 3 and 2. Your answer does not have to be sorted). Similarly, the four smallest numbers are 3, 3, 2, 3.

4. (10) Will we still have a linear selection algorithm if elements are grouped into groups of 7? Prove your answer. What about 3?

5. (10) Quicksort has a worst case running time of $O(n^2)$ and a best and average case running time $O(n \log n)$. How can you modify the quicksort so that its worst case running time is $O(n \log n)$?

6. (20) (a) What is the largest k such that if you can multiply 3×3 matrices using k multiplications, then you can multiply $n \times n$ matrices in time $o(n^{\log 7})$? What would the running time of this algorithm be? You can assume that n is a power of 3.

   (b) V. Pan has discovered a way of multiplying 68×68 matrices using 132,464 multiplications, a way of multiplying 70×70 matrices using 143,640 multiplications, and a way of multiplying 72×72 matrices using 155,424 multiplications. Which method yields the best asymptotic running time when used in a divide-and-conquer matrix-multiplication algorithm? How does it compare to Strassen's algorithm?