CS 5633 Analysis of Algorithms – Spring 10

1/21/10

2. Homework

Due **1/28/10** before class

Always justify your answers. All algorithms should be as efficient as possible and all runtimes should be as tight as possible.

1. **Guessing and Induction (8 points)**

   For each of the following recurrences use the recursion tree method to find a good guess of what it could solve to (make your guess as tight as possible). Then prove that $T(n)$ is in big-Oh of your guess by induction (inductive step and base case). *(Hint: Appendix A in the book has a list of solved summations that might be helpful. For simplicity you may want to use* $\log_3 n$ *instead of* $\log_2 n$ *in some cases.)*

   Every recursion below is stated for $n \geq 2$, and the base case is $T(1) = 1$.

   (a) **(4 points)**
   $T(n) = 2T(\frac{n}{3}) + n^2$

   (b) **(4 points)**
   $T(n) = 4T(\frac{n}{2}) + 3n$

2. **Master theorem (6 points)**
   Use the master theorem to find tight asymptotic bounds for the following recurrences. Justify your answers (by showing which case of the master theorem applies, and why.) Assume that $T(1) = 1$.

   (a) **(2 points)**
   $T(n) = T(\frac{n}{3}) + \sqrt{n}$

   (b) **(2 points)**
   $T(n) = 8T(\frac{n}{2}) + n^3 \log^2 n$

   (c) **(2 points)**
   $T(n) = 9T(\frac{n}{3}) + n \log n$

FLIP OVER TO BACK PAGE $\Longrightarrow$

3. **Multiplying polynomials (10 points)**
A polynomial of degree n is a function

$$p(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

where a_i are constants and $a_n \neq 0$. For simplicity you may assume that n is a power of 2.

(a) (1 point) What is the runtime of the straight-forward algorithm for multiplying two polynomials of degree n?

(b) (5 points) We can rewrite the polynomial $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ as
$x^{n/2}(a_n x^{n/2} + a_{n-1} x^{n/2-1} + \cdots + a_{n/2+1} + a_{n/2}) + (a_{n/2-1} x^{n/2-1} + \cdots + a_1 x + a_0)$.
Use this as a starting point to design a divide-and-conquer algorithm for multiplying two degree-n polynomials (recurse on polynomials of degree $n/2$). Give a runtime analysis of your algorithm by setting up and solving a recurrence. The runtime of your algorithm should be the same as the runtime of part a).

(c) (1 point) Show how to multiply two degree-1 polynomials $ax + b$ and $cx + d$ using only three multiplications. *Hint: One of the multiplications is* $(a + b) \cdot (c + d)$.

(d) (3 points) Design a divide-and-conquer algorithm for multiplying two polynomials of degree n in time $\Theta(n^{\log_2 3})$. *Hint: Reuse part* **b)** *and speed it up with the knowledge of part* **c)**