CS 3343 Analysis of Algorithms – Spring 09

2/3/09

3. Homework

Due **2/10/09** before class

1. **Master theorem (8 points)**
   Use the master theorem to solve the following recurrences. Justify your results. Assume that $T(1) = 1$.

   - $T(n) = 8T(\frac{n}{2}) + n^3 \log^3 n$
   - $T(n) = 81T(\frac{n}{3}) + n^2 \log^2 n$
   - $T(n) = T(\frac{n}{2}) + \log n$
   - $T(n) = 16T(\frac{n}{4}) + n^2\sqrt{n}$

2. **Divide and Conquer (7 points)**

   Let $A[1..n]$ be an array of n distinct numbers. An *inversion* is a pair $(A[i], A[j])$ where $i < j$ and $A[i] > A[j]$, i.e., the numbers are out of order. In this problem we want to count the number of inversions in A. For example if $A[1..8] = 3, 7, 4, 8, 2, 9, 1, 6$ then there are 14 inversions: $(3, 2)$, $(3, 1)$, $(7, 4)$, $(7, 2)$, $(7, 1)$, $(7, 6)$, $(4, 2)$, $(4, 1)$, $(8, 2)$, $(8, 1)$, $(8, 6)$, $(2, 1)$, $(9, 1)$, $(9, 6)$.

   (a) (1 point) In the best case, how many inversions can an array of n numbers have? Justify your answer.

   (b) (1 point) In the worst case, how many inversions can an array of n numbers have? Justify your answer.

   (c) (5 points) Write a divide-and-conquer algorithm that counts the number of inversions in an array.

   - Follow the standard divide and conquer approach by dividing in half and recursing on both halves. How can you use the sub-solutions to obtain a solution for the whole array?
   - Try to develop an algorithm that is as efficient as possible. It is relatively easy to get a $O(n^2)$ algorithm, $O(n \log^2 n)$ is a bit harder, and even $O(n \log n)$ is possible. (For the latter you may need a preprocessing step that makes a copy of the input array and sorts it.) You will get more points for a more efficient algorithm.
   - Analyze your runtime by setting up a runtime recurrence and using the master theorem to solve it.

Flip over to back page ⟹

3. **Guessing and Induction (12 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 big-Oh-induction (= substitution method; including base case and inductive case).

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

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

(b) $T(n) = 2T(\frac{n}{3}) + 5n$ *(Hint: You may want to use* $\log_3 n$ *instead of* $\log_2 n$*.)*