COMP 271 Design and Analysis of Algorithms
2005 Spring Semester
Written Assignment 1
Distributed: March 3, 2005
Due: March 17, 2005 *at start of class*

Note: Please follow the guidelines on doing your own work and avoiding plagiarism on the class home page.

Problem 1. Prove the following properties of the $O()$-notation:

1. If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$.
2. If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$, then $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$.
3. If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$, then $f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$.

Problem 2. The following recurrence often arises in divide-and-conquer algorithms, where the processes of dividing or combining involves sorting.

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T\left(\frac{n}{2}\right) + n \lg n \qquad \text{for } n > 1, \end{aligned}$$

where n is a power of 2. *Prove* that $T(n) = \Theta(n \log^2 n)$.

Problem 3. Let $T(n, i)$ denote the average number of comparisons of array elements done by the *Randomized-Select* algorithm for determining the ith smallest of n elements.

(a) Write the recurrence relations for $T(n, 1)$, $T(n, 2)$, and $T(n, 3)$.

(b) Give the exact values of $T(1, 1)$, $T(2, 2)$, and $T(3, 3)$. Show your calculations.

Problem 4. CLRS p.162 **7-5**.

Problem 5. CLRS p.192 **9.3-5**.

Problem 6. CLRS p.548 **22.3-4**.
(Follow the definition of edges classification for DFS on a directed graph on p.546.)

Problem 7. CLRS p.559 **22-3**.

A Note about Writing Algorithms: When writing pseudo-code do *not* write a complete program with variable and class declarations. Explain the algorithm intuitively in English, and then give a high-level pseudocode description. The level of detail should be just enough that a competent programmer could take your explanation and implement it. You may assume that low-level data structures have been provided for you. So, it is clearer to say "Append x to the end of list L", rather than giving code for pointer manipulation.

It is a good idea to provide an example to illustrate how your algorithm works on a concrete example. Reading pseudocode is hard, but a good example makes things much easier to understand. It also provides the grader with more understanding your intentions. This can also help you receive partial credit in case you make a coding error.

Throughout this course, every algorithm should be accompanied with (1) an explanation of its correctness and (2) and explanation of its running time.