Algorithms: Assignment 3

Due date: September 21 (Thursday)

Problem 1 (5 points)
Determine asymptotic upper and lower bounds for each of the following recurrences. Make your bounds as tight as possible.

(a) $T(n) = T(n/6) + T(n/3) + T(n/2) + n$.

(b) $T(n) = T(n-1) + n$.

(c) $T(n) = T(n-1) + 1/n$.

(d) $T(n) = T(\sqrt{n}) + 1$.

(e) $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$.

Problem 2 (5 points)
The standard complexity analysis of MERGE-SORT(A, p, q) is based on the assumption that, when making a recursive call, we pass the array $A[1..n]$ by a pointer, which means that parameter passing takes constant time. If a programming language does not allow passing an array by a pointer, we may have two other options:

(a) Pass the array $A[1..n]$ by copying all its elements, which takes $\Theta(n)$ time.

(b) Pass the array by copying only the elements used in the called function; then, the procedure has to copy the $A[p..q]$ segment of the array, which takes $\Theta(q-p+1)$ time.

For each of these two options, write a recurrence for the running time of MERGE-SORT and give an asymptotic upper and lower bound for the recurrence.

Problem 3 (bonus)
This problem is optional; if you solve it, you will get 2 bonus points toward your final grade for the course. You cannot submit this bonus problem after the deadline.

Suppose that $A[1..n]$ and $B[1..m]$ are sorted arrays, and the size of A is no greater than the size of B, that is, $n \leq m$. Write an algorithm that finds the smallest common element of these arrays; for example, if $A = \langle 1, 4, 5, 7 \rangle$ and $B = \langle 2, 3, 4, 7, 8 \rangle$, then the smallest common element is 4. If the arrays have no common elements, the algorithm should return 0.

Your solution should be efficient both when A is almost as large as B and when A is much smaller than B. In particular, if A is much smaller, the complexity should be better than $\Theta(m)$.