Homework No. 1 (Updated)

CSCI 4470/6470 Algorithms, CS@UGA, Spring 2017

Due Tuesday January 24, 2015

Seven questions; each question is worth 20 points. There are 140 points for graduate students and 110 points for undergraduates.

1. Assume algorithm A solves problem Π. Also assume that U(n) and L(n) are a time upper bound and a time lower bound for problem Π, respectively. Let T_A(n) and S_A(n) be the worst case time upper bound and the worst case time lower bound for algorithm A, respectively. What would you say about the following relations? Explain for each relation.
   (a) U(n) vs T_A(n).
   (b) U(n) vs S_A(n).
   (c) L(n) vs T_A(n).
   (d) L(n) vs S_A(n).

2. Using the definition of Big-O to prove
   (a) If T(n) = O(f(n)) and S(n) = O(f(n)), T(n) + S(n) = O(f(n)).
   (b) If T(n) = O(f(n)), it is not necessary that 2^{T(n)} = O(2^{f(n)}).

3. Consider the following summation of the harmonic sequence up to the n-th term, for any n ≥ 1,
   \[ h(n) = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n} \]

   (1) (Graduate students only, bonus for undergraduates) Prove that we have the following recurrence for h(n). (No math induction is needed in your proof.)
   \[ h(n) \leq h([n/2]) + 1 \]
(2) Use the substitution method to prove that, there are constant \( c > 0, k > 0 \) such that
\[
h(n) \leq c \log_2 n
\]
for all \( n \geq k \).

4. Find an upper bound (the big-\( O \) notation) for the following recurrence:
\[
T(n) = 3T(\lfloor \frac{n}{4} \rfloor) + n, \quad T(1) = 2
\]
(1) Use the substitution method (i.e., guess and check) to prove.
(2) Use the recursive tree method to prove the upper bound.

5. (Graduate students only, bonus for undergraduates) Suppose the recurrence in Question 4 is changed to
\[
T(n) = 3T(\lfloor \frac{n}{4} \rfloor + 2) + n, \quad T(1) = 2
\]
Prove that the upper bound you have derived for Question 4 still holds here.

6. The Insertion Sort algorithm discussed in the class (also in the textbook) can be written as recursive algorithms, by changing either or both of the loops in Insertion Sort to recursive calls. This exercise asks you to rewrite Insertion Sort to a recursive algorithm, namely Rec-Insertion Sort, by replacing only the outer \textbf{for} loop with a recursive call.

You would need to use the following recursion scheme for the recursive insertion sort. Let \((A, n)\) be an array with \( n \) elements to be sorted. The algorithm first sorts the prefix \( n - 1 \) items and then inserts the last item \( A[n] \) into the sorted prefix.

(1) Give such a recursive algorithm for insertion sort.

(2) Analyze your recursive algorithm: first give a recurrence for the time function of your algorithm and then prove that the time function is bounded by a quadratic function in \( n \).

(3) Someone has suggested that the steps for the inner \textbf{while} loop could be done more efficiently since the prefix sublist has already been sorted. For example, one could use a “binary search” to quickly identify where the new item can be inserted, which could be done in \( O(\log_2 n) \) time for each new item. So the total time would be \( O(n \log_2 n) \) instead of \( O(n^2) \). Prove or disprove this suggestion.
7. The **HeapSort** algorithm first builds the maximum heap with subroutine **Max-Heapify**, which is called ⌈n/2⌉ times, from the non-leaf node with the highest index to the one with the lowest index (i.e., the root). Would the algorithm still work if the order of these calls to **Max-Heapify** is reversed, i.e., from the root to the non-leaf node with the highest index? Justify your answer.

The answers must be word-processed or typed. You may substitute formulae and figures with hand-writings. Your submitted algorithms should be in the pseudo-code, not in any specific programming language. Answers deviating from these requirements will be returned without grading.

The answers must be the student’s own work. Idea sharing and referencing to others’ work (including those online) are not allowed. Plagiarism and other forms of academic dishonesty will be handled within the guidelines of the Student Handbook and reported to the University.