Homework Assignment No. 1

CSCI 4470/6470 Algorithms, CS@UGA, Spring 2018

Due Thursday January 25, 2018

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.

Six questions; each question is worth 20 points. There are 120 points for graduate students and 100 points for undergraduates.

1. Using the definition of Big-$O$ to prove
   (a) If $T(n) = 200n \log_2 n + 10n$, then $T(n) = O(n^2)$;
   (b) If $T(n) = \frac{1}{200}n^2$, then $T(n) \neq O(n \log_2 n)$

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

3. Analyze the time upper bound $T(n)$ for the following algorithm BubbleSort, where $n$ represents the number of items in the list $A$. 


1
Algorithm BubbleSort(A);
1. n = length(A);
2. swapped = true;
3. m = n-1;
4. While (swapped = true) do
5. swapped = false
6. for i = 1 to m do
7. if A[i-1] > A[i] then
8. swap( A[i-1], A[i] )
9. swapped = true
10. m = m - 1
11. return (A)

4. Binary search on a sorted list of $n$ numbers has the time complexity $T(n)$ that satisfies the recurrence

$$T(n) = T\left(\frac{n}{2}\right) + c$$

where $c \geq 1$ is a constant. Use the substitution method (i.e., induction) method to prove that $T(n) = O(\log_2 n)$.

5. Consider a variant of Merge Sort that splits an unsorted list of $n$ items into two sublists of lengths $\frac{2n}{3}$ and $\frac{n}{3}$, recursively sorts the two sublists, respectively, and merges the sublists into one after they are sorted.

Let $T(n)$ be the time upper bound of this algorithm. Give a recurrence that is satisfied by $T(n)$ and based on the recurrence to prove that $T(n) = O(n \log_2 n)$ using the recursive tree method.

6. (Graduate students only, bonus for undergraduates) We know that every loop can be replaced with a recursion solution. Rewrite the algorithm Insertion Sort as a recursive algorithm such that no loop statements (such as for and while) appear in the algorithm.