CSci 4041: Algorithms and Data Structures (Spring'15)

Homework 1, Due 02/05/15 (in class)

Answer all of the following questions, and always explain your answer. You are expected to do the work on your own.

Any proofs you do have to be rigorous, covering all cases, to get credit. Just giving an example does not constitute a correct proof, and will not receive credit. When you are disproving something, a counter example is sufficient.

The homework is due in class on Feb 05. We want a paper copy submission.

Good Luck!

1. (30 points) Bubblesort is a popular, but inefficient, sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order.

   BUBBLESORT(A)

   ```
   1  for i = 1 to A.length − 1
   2      for j = A.length downto i + 1
   3          if A[j] < A[j − 1]
   4              exchange A[j] with A[j − 1]
   ```

   Let A' denote the output of BUBBLESORT(A). To prove that BUBBLESORT is correct, we need to prove that

   $$A'[1] \leq A'[2] \leq \cdots \leq A'[n] \ , \tag{1}$$

   where $n = A.length$.

   (a) (15 points) State precisely a loop invariant for the for loop in lines 2-4, and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in Chapter 2.

   (b) (15 points) Using the termination condition of the loop invariant proved in part (a), state a loop invariant for the for loop in lines 1-4 that will allow you to prove inequality (1). Your proof should use the structure of the loop invariant proof presented in Chapter 2.

2. (25 points) Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove **any five** of the following conjectures:

   (a) $f(n) = O(g(n))$ implies $g(n) = O(f(n))$.

   (b) $f(n) = O(g(n))$ implies $\log(f(n)) = O(\log(g(n)))$, where $\log g(n) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n.

(c) $f(n) = O(g(n))$ implies $2^{f(n)} = O(2^{g(n)})$.

(d) $f(n) = O((f(n))^2)$.

(e) $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$.

(f) $f(n) = \Theta(f(n/2))$.

(g) $f(n) + o(f(n)) = \Theta(f(n))$.

3. (30 points) Give asymptotic upper and lower bounds for $T(n)$ in **any six** of the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

   (a) $T(n) = 4T(n/2) + n^2\sqrt{n}$.

   (b) $T(n) = T(7n/10) + n$.

   (c) $T(n) = T(n-2) + n^2$.

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

   (e) $T(n) = T(n-1) + \log n$.

   (f) $T(n) = T(n-1) + 1/\log n$.

   (g) $T(n) = 2T(n/2) + n/\log n$.

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

4. (15 points) We consider the maximum-subarray problem discussed in class. Use the following ideas to develop a nonrecursive, linear-time algorithm for the maximum-subarray problem:

   Start at the left end of the array, and progress toward the right, keeping track of the maximum subarray seen so far. Knowing a maximum subarray of $A[1 \ldots j]$, extend the answer to find a maximum subarray ending at index $j+1$ by using the following observation: a maximum subarray of $A[1 \ldots j+1]$ is either a maximum subarray of $A[1 \ldots j]$ or a subarray $A[i \ldots j+1]$, for some $1 \le i \le j+1$. Determine a maximum subarray of the form $A[i \ldots j+1]$ in constant time based on knowing a maximum subarray ending at index j.