CS140: Algorithms
Fall 1999
Assignment 2, 9/15/99
Due Wed, 9/22/99

This assignment is due at the start of class on Wed., 9/22/99. No late homework will be accepted. Your solutions will be graded for clarity, conciseness, and correctness.

1. (20 pts.) Compute the running times for the following algorithms:

   (a) `algorithm` veryodd(n)

   ```
           for i = 1 to n do
              if odd(i) then
                 begin
                    for j = i to n do x ← x + 1
                    for j = 1 to i do y ← y + 1
                 end
           end
   end algorithm
   ```

   (b) `algorithm` what(n)

   ```
           for i = 1 to ⌊√n⌋ do
              for j = ⌊√n⌋ downto 1 do
                 for k = 1 to j do
                    { O(1) time sequence of statements }
                 end
              end
           end
   end algorithm
   ```

2. (25 pts.) Algorithm Resquare is a recursive algorithm that works on inputs of size $n = 2^{2^m}, m \geq 0$. On input of size 2 the algorithm performs a constant amount of work. On input size $n = 2^{2^m}, m > 0$, the algorithm makes $\sqrt{n}$ recursive calls, each on an instance of size $\sqrt{n}$. The algorithm performs an additional cn steps to combine the results of the recursive calls. Draw the work tree for Resquare for $m = 0, m = 1$, and $m = 2$. Answer the following questions with respect to the work tree for Resquare for arbitrary m.

   (a) How many nodes are at level i, where the root is at level 1?

(b) What is the instance size associated with a node at level i?

(c) How much work is associated with a node at level i?

(d) How many levels are in the work tree?

(e) What is the running time of Resquare?

3. (25 pts.) Problem 4-2 on page 73 of Cormen, Leiserson, Rivest.

4. (30 pts.) In analyzing the running time of recursive algorithms, it is often convenient to focus on instances of particular sizes; e.g. b^m, where $b > 0$ is constant and $m \in \{0, 1, \ldots\}$. In this problem we'll explore conditions for which this restricted analysis is sufficent to characterize the running time of an algorithm on all input sizes. Let k be a positive integer. Let $f(n) = n^k$ and $g(n) = k^n$. Define $h(0) = 1$ and for $n > 0$, $h(n) = 2^{h(n-1)}$.

   (a) Which of the following statements are true:

      i. $f(n) = O(f(n-1))$

      ii. $f(b^m) = O(f(b^{m-1}))$ where b and m are positive integers.

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

   (b) Repeat question (a) replacing f by g.

   (c) Repeat question (a) replacing f by h.

   (d) For each of the following conditions, construct a monotonically increasing function $s : \mathcal{N} \to \mathcal{N}$ that satisfies the condition or state that such a function cannot exist.

      i. $s(n) = f(n)$ when n is a power of 2 but $s(n) \neq O(f(n))$.

      ii. $s(n) = g(n)$ when n is a power of 2 but $s(n) \neq O(g(n))$.

      iii. $s(n) = h(n)$ when n is a power of 2 but $s(n) \neq O(h(n))$.