CS140: Algorithms

Spring 2001

Review Problems

This is an optional assignment. Any points you earn on this assignment will be used (at a 50% discount) to augment your grades for earlier assignments. To receive credit for this assignment you must submit it by Tuesday, Feb. 6. Your solutions must be typeset. You may NOT work with your classmates to complete this assignment.

1. Order the following functions by inreasing rate of growth. Indicate any equivalence classes you find:

$\log_3 n^3$	$n^2 + \log n$	$n^2 \lg n$	$\lg^2 n$	$2 \uparrow\uparrow (n)$
$2 \uparrow\uparrow (2n)$	$3^{\log_5 n}$	n^2	$(\lg n)^{\lg n}$	$n/\lg n$

2. For each of the following pair $f(n)$ and $g(n)$ state whether or not (i) $f(n) = O(g(n))$, (ii) $f(n) = o(g(n))$, (iii) $f(n) = \Omega(g(n))$, (iv) $f(n) = \omega(g(n))$, (v) $f(n) = \theta(g(n))$.

   (a) $f(n) = n^2 + n$ and $g(n) = n^3 - n^2$

   (b) $f(n) = n^2$ and $g(n) = f(2n)$

   (c) $f(n) = n!$ and $g(n) = f(2\lceil n/2 \rceil)$

3. State whether the following are true or false. You should assume that f and g are monotonically-increasing positive-valued functions on the natural numbers.

   (a) If for every positive integer n, algorithm $\mathcal{A}$ performs $n^3 - n$ steps on some input of size n then $\mathcal{A}$ is $O(n^3)$.

   (b) If $f(n)/g(n) = 2$ for even values of n then $f(n) \neq \omega(g(n))$.

   (c) If $f(n) = o(g(n))$ then $f(n) = O(g(n))$.

   (d) If $f(n) = O(g(n))$ then $f(n) = o(g(n))$.

   (e) If $\log(f(n)) = O(\log(g(n))$ then $f(n) = O(g(n))$.

   (f) If $2^{f(n)} = O(2^{g(n)})$ then $f(n) = O(g(n))$.

4. Algorithm $\mathcal{A}$ works on inputs of size $n = 2 \uparrow\uparrow (m)$, $m = 0, 1, ...$; i.e. towers of 2. On an input of $n = 1$ the algorithm performs a constant number of steps. On an input of size $n = 2 \uparrow\uparrow (m)$, $m > 0$, $\mathcal{A}$ makes $n/lg(n)$ recursive calls, each on an instance of size $\lg(n)$. It then performs an additional cn steps to produce its output.

(a) Write the recurrence relation of the running time.

(b) Consider the work tree for $\mathcal{A}$ on input of size $n = 2 \uparrow\uparrow (m)$.

i. What is the input size at level i, where the root is at level 0?

ii. How many nodes are at level i?

iii. What is the total work done by the nodes at level i?

iv. How many levels are in the work tree?

v. Write a summation that represents the running time.

(c) What is the running time of the algorithm?

(Note: Recall the notation for the ith iterated log of n is $lg^{(i)}n$; i.e. $lg^{(0)}n = n$ and for $i > 0$, $lg^{(i)}n = lg(lg^{(i-1)}n)$. This may be useful.)

5. Compute the running time of the following algorithm. (Note: As its name suggests, the boolean function odd(i) returns true if i is odd and false if i is even.)

```
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
```

6. Give asymptotically tight bounds for the following recurrences.

(a) $f(n) = 3f(n/2) + \sqrt{n}$

(b) $f(n) = f(9n/10) + n$

(c) $f(n) = f(\sqrt{n}) + n$