# CS140: Algorithms

## Spring 2001

### Solutions to Review Problems

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

$$\begin{array}{lllll} \log_3 n^3 & n^2+\log n & n^2 \lg n & \lg^2 n & 2 \uparrow\uparrow (n) \\ 2 \uparrow\uparrow (2n) & 3^{\log_5 n} & n^2 & (\lg n)^{\lg n} & n/\lg n \end{array}$$

   Ans.

$$\begin{array}{l} \log_3 n^3 \\ \lg^2 n \\ n/\lg n \\ 3^{\log_5 n} \\ n^2+\log n, \quad n^2 \\ n^2 \lg n \\ (\lg n)^{\lg n} \\ 2 \uparrow\uparrow (n) \\ 2 \uparrow\uparrow (2n) \end{array}$$

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 $g(n) = n^3 - n^2$
   Ans. (i) and (ii) hold, (iii), (iv), and (v) do not hold
   (b) $f(n) = n^2$ and $g(n) = f(2n)$
   Ans. Note that $g(n) = f(2n) = 4n^2$ so (i), (iii) and (v) hold, (ii) and (iv) do not hold.
   (c) $f(n) = n!$ and $g(n) = f(2\lceil n/2 \rceil)$
   Ans. Note that $2\lceil n/2 \rceil = n$ when $n$ is even and it equals $n+1$ when $n$ is odd. Thus $g(n) = n!$ when $n$ is even and $(n+1)!$ when $n$ is odd. And therefore the ratio $f(n)/g(n)$ is 1 when $n$ is even and $1/n$ when $n$ is odd. So (i) holds and (ii), (iii), (iv), and (v) do not hold.

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 $O(n^3)$.
   Ans. False

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

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

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

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

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

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.
   Ans. $T(n) = \frac{n}{\lg(n)} T(\lg(n)) + cn$

   (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?
      Ans. $2 \uparrow\uparrow (m - i)$

      ii. How many nodes are at level $i$?
      Ans. $\frac{2\uparrow\uparrow m}{2\uparrow\uparrow(m-i)}$

      iii. What is the total work done by the nodes at level $i$?
      Ans. $cn = c2 \uparrow\uparrow m$

      iv. How many levels are in the work tree?
      Ans. $m + 1$

      v. Write a summation that represents the running time.
      Ans. $\sum_{i=0}^{m} cn$

   (c) What is the running time of the algorithm?
   (Note: Recall the notation for the $i$th 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.)
   Ans. Note that $m = \lg^* n$ so the running time is $n \lg^2 n$.

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
```

Ans. $T(n) = O(n^2)$

6. Give asymptotically tight bounds for the following recurrences.

   (a) $f(n) = 3f(n/2) + \sqrt{n}$
   Ans. $f(n) = \theta(n^{\lg 3})$

   (b) $f(n) = f(9n/10) + n$
   Ans. $f(n) = \theta(n)$

   (c) $f(n) = f(\sqrt{n}) + n$
   Ans. $f(n) = \sum_{i=0}^{\lg \lg n} 2^{2^i} = \theta(n)$