

IT-math F2003 : Selected Solution(s)

Episode 10, April 8, 2003

FP3. Prove that if $f = O(g)$ and $g = o(h)$, then $f = o(h)$.

Solution. Fix $c > 0$. Since $f = O(g)$, there exists a $D > 0$ such that $|f(n)| \leq D \cdot |g(n)|$ for almost all n . Since $g = o(h)$, one has $|g(n)| < \frac{c}{D} \cdot |h(n)|$ for almost all n . Therefore

$$|f(n)| < D \cdot |g(n)| \leq D \cdot \frac{c}{D} \cdot |h(n)| = c \cdot |h(n)|$$

for almost all n , showing $f = o(h)$.

SC1(a). Prove or refute: $n \cdot \log n = o(n^2)$.

Solution. We show $n \cdot \log_a n = o(n^2)$ (any $a > 1$). Let $c > 0$. Let $\delta = a^c - 1 > 0$ (as $a^c > 1$). We have $n = o\left(\binom{n-1}{2} \cdot \delta^2\right)$ since the right hand side is a polynomial in n of degree 2. Therefore $n < \binom{n-1}{2} \cdot \delta^2$ for almost all n . Hence for almost all n one has

$$n < \binom{n-1}{2} \cdot \delta^2 \leq \binom{\lfloor n \rfloor}{2} \cdot \delta^2 \leq \sum_{0 \leq k \leq \lfloor n \rfloor} \binom{\lfloor n \rfloor}{k} \cdot \delta^k \stackrel{(*)}{=} (1 + \delta)^{\lfloor n \rfloor} = (a^c)^{\lfloor n \rfloor} \leq a^{c \cdot n}$$

(where the equality (*) holds by the Binomial Theorem). Taking logarithm to base a we get $\log_a n < c \cdot n$ (almost all n). If $n > 0$ we can multiply both sides of the latter inequality by n : $n \cdot \log_a n < c \cdot n^2$ (almost all n). This shows $n \cdot \log_a n = o(n^2)$.

SC1(b). Prove or refute: $2 \log n + 4n + 3n \log n = o(n \cdot \log 2n)$.

Solution. We refute this by showing that $2 \log n + 4n + 3n \log n = \Omega(n \cdot \log 2n)$ (thus by Lemma 7(b) from the Supplementary Material $2 \log n + 4n + 3n \log n = o(n \cdot \log 2n)$ cannot be the case). First, observe that

$$n \cdot \log 2n = n \cdot (\log 2 + \log n) = n \cdot \log 2 + n \cdot \log n.$$

We have $n \cdot \log 2 = O(3n \log n)$ because $n \cdot \log 2 \leq 3n \cdot \log n$ for all $n \geq 2$, and $n \cdot \log n = O(3n \cdot \log n)$ because $n \cdot \log n \leq \frac{1}{3} \cdot 3n \cdot \log n$ for all $n > 0$. Therefore

$$n \cdot \log 2n = n \cdot \log 2 + n \cdot \log n = O(3n \cdot \log n),$$

or equivalently $3n \log n = \Omega(n \cdot \log 2n)$. For an arbitrary $c > 0$, if $n > \frac{2}{c}$ and $n > 1$, we have $2 \log n < c \cdot n \cdot \log 2n$. Also, if $n > \frac{a^{4/c}}{2}$ then $4n < c \cdot n \cdot \log_a 2n$ (where $a > 1$). These considerations show that $2 \log n = o(n \cdot \log 2n)$ and $4n = o(n \cdot \log 2n)$. Recalling Lemma 9(b) we get $2 \log n + 4n + 3n \log n = \Omega(n \cdot \log 2n)$ as required.

SC1(c). Prove or refute: $3^n = O(2^n)$.

Solution. We refute this by showing $2^n = o(3^n)$. Fix $c > 0$. Assume $n > \log_{3/2} \frac{1}{c}$. Then $\left(\frac{3}{2}\right)^n > \frac{1}{c}$ hence $2^n < c \cdot 3^n$. This shows $2^n = o(3^n)$.

SC1(d). Prove or refute: $n! = o(n^n)$.

Solution. We prove that this is indeed the case. Let $c > 0$. Suppose $n > \frac{1}{c}$ and $n \geq 1$. Then

$$n! = 1 \cdot 2 \cdot 3 \cdots n < (c \cdot n) \cdot \underbrace{2 \cdot 3 \cdots n}_{n-1 \text{ factors}} \leq c \cdot n \cdot \underbrace{n \cdots n}_{n-1 \text{ factors}} = c \cdot n^n.$$

Thus $n! = o(n^n)$.

SC1(e). Prove or refute: $n = \Theta(\lceil n \rceil)$.

Solution. We prove that this is the case. We have $n \leq \lceil n \rceil$ for all n . Therefore $n = O(\lceil n \rceil)$. For $n \geq 1$ one has $2n \geq \lceil n \rceil$. Hence $n = \Omega(\lceil n \rceil)$. Thus $n = \Theta(\lceil n \rceil)$.

SC3. Consider the following fragment of Java code:

```
int x = 0;
for (i = 1; i <= n; i++) {
    for (j = 1; j <= i; j++) {
        x++;
    }
}
```

If n is the value of an `int` variable `n` immediately before the execution of this fragment, $n > 0$, and $x(n)$ is the value of `x` immediately after the execution (considered as a function of n), show that $x(n) = \Theta(n^2)$.

[In this exercise, assume that Java does perfect integer arithmetic rather than arithmetic mod 2^{32} .]

Solution. Since the only statement in the above fragment that changes the value of `x` is the statement `x++`, we just have to estimate the number of times this statement is executed. The outer loop is executed n times, once with each value of `i` with `i` going from 1 to n . When the inner loop is executed, the statement `x++` gets executed i times, where i is the current value of `i`. Therefore the total number of times `x++` is executed is $x(n) = 1 + 2 + \cdots + n = \sum_{i=1}^n i = \frac{n(n+1)}{2}$. This being a polynomial of degree 2 in n , we have $x(n) = \Theta(n^2)$.

LH1. Give an example of three functions $f, g, h : \mathbb{N} \rightarrow \mathbb{R}$ such that $f = \Omega(g)$, $h = \Omega(g)$, yet $f + h \neq \Omega(g)$.

Solution. It is sufficient to take the functions defined by $f(n) = n$, $h(n) = -n$, and $g(n) = 1$ for all n . [Observe that $(f + h)(n) = 0$ for all n .]

LH2. Prove or refute: If $f(n) = \Theta(g(n))$ then $2^{f(n)} = \Theta(2^{g(n)})$.

Solution. We shall refute this. Let $f(n) = n$ and $g(n) = 2n$. Clearly $f(n) = \Theta(g(n))$. Further, $2^{f(n)} = 2^n$ and $2^{g(n)} = 2^{2n} = 4^n$. As in the solution of SC1(c), one shows $2^{f(n)} = o(2^{g(n)})$. Therefore by Lemma 7(b) one cannot have $2^{f(n)} = \Omega(2^{g(n)})$. Hence $2^{f(n)} \neq \Theta(2^{g(n)})$.

LH3. Assume the functions f and g take on positive values only. Show that $f(n)+g(n) = \Theta(h(n))$, where the function h is defined by $h(n) = \max\{f(n), g(n)\}$.

Solution. First we show $f + g = O(h)$. Since $f(n) \leq \max\{f(n), g(n)\}$ and $g(n) \leq \max\{f(n), g(n)\}$, we have $|f(n) + g(n)| = f(n) + g(n) \leq 2 \cdot \max\{f(n), g(n)\} = 2 \cdot h(n) = 2 \cdot |h(n)|$ for all n . This shows $f + h = O(h)$.

Further, as $f(n) \geq 0$ and $g(n) \geq 0$, we have $|f(n) + g(n)| = f(n) + g(n) \geq \max\{f(n), g(n)\} = h(n) = |h(n)|$ for all n . This shows $f + h = \Omega(h)$.

DS1. Prove or refute: For any functions $f, g : \mathbb{N} \rightarrow \mathbb{R}$, $f(n) = O(g(n))$ implies $f(n+1) = O(g(n))$.

Solution. We refute this by the following example: $f(n) = g(n) = n!$. Clearly, $f(n) = O(g(n))$. We show $g(n) = o(f(n+1))$, so that one cannot have $f(n+1) = O(g(n))$ (recall Lemma 7(b)).

Let $c > 0$. If $n > \frac{1}{c}$ then $n+1 > n > \frac{1}{c}$, so that $1 < c \cdot (n+1)$. Multiplying both sides by $n! > 0$ we get $n! < c \cdot (n+1) \cdot n! = c \cdot (n+1)!$, or in other words $|g(n)| < c \cdot |f(n)|$. Thus $n! = o((n+1)!)$ as required.