Prof. Friedrich Eisenbrand
Nicolai Hähnle

Discussions from: February 22, 2011

# Computer Algebra

Spring 2011

Solutions 1

*Note:* These are just notes and not necessarily full solutions for each exercise. Please report any mistakes you may find.

**Exercise 1**
The following list shows the functions in "ascending order". That is, if function $f$ is shown to the left of function $g$, than $f = O(g)$. If they are separated by a semicolon, then $f = o(g)$. Functions $f, g$ that are separated only by a comma have the same asymptotic growth, i.e. $f = \Theta(g)$.

$$13;\ \log n^{1337},\ \log n;\ \log^2 n;\ \sqrt{n};\ 2^{3+\log n},\ 3n;\ e^{\log n};\ 2^{4\log n};\ n^6 - 5n^2;\ 2^{\log^2 n};\ 2^n;\ 4^n$$

Proofs are left to the reader.

**Exercise 2**
Let $f, g : \mathbb{N} \to \mathbb{R}_+$. To show: $f = O(g)$ if and only if $\limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty$.

Let $f = O(g)$. By definition, there is a $c > 0$ and an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ we have $f(n) \leq c g(n)$. This means that $\frac{f(n)}{g(n)} \leq c$ for all $n \geq n_0$, and so $\limsup_{n\to\infty} \frac{f(n)}{g(n)} \leq c$.

Now suppose $\limsup_{n\to\infty} \frac{f(n)}{g(n)} = c < \infty$. By definition, this means that for all $\varepsilon > 0$ there exists an $n_0 \in \mathbb{N}$ such that for all $n_1 \geq n_0$ one has $|\sup\left\{\frac{f(n)}{g(n)} \mid n \geq n_1\right\} - c| \leq \varepsilon$. In particular, $\sup\left\{\frac{f(n)}{g(n)} \mid n \geq n_0\right\} \leq c + \varepsilon$. Let us choose $\varepsilon = 1$. Then it follows that for all $n \geq n_0$ we have $\frac{f(n)}{g(n)} \leq c + 1$, i.e. $f(n) \leq (c+1)g(n)$ and thus $f = O(g)$.

**Exercise 4**
Let $A_1$ and $A_2$ be algorithms for the same problem which run for $T_1(n) = 5n^2$ and $T_2(n) = 1000 n \log n$ machine operations on an input of size $n$, respectively. Let $M_1$ be a machine that can execute $10^{10}$ machine operations per second, and $M_2$ a machine that can execute $10^6$ machine operations per second. For which values of $n$ is $A_1$ on $M_1$ faster than $A_2$ on $M_2$?

Let's call $s_1(n) = T_1(n) \cdot 10^{-10}$ and $s_2(n) = T_2(n) \cdot 10^{-6}$ the number of seconds to process an input of size $n$ in the first and second scenario, respectively. One easily sees that $s_1$ starts out smaller but grows faster than $s_2$. So where is the point where $s_1$ and $s_2$ reach parity?

$$s_1(n) = s_2(n) \implies 5 \cdot 10^{-10} n^2 = 1000 \cdot 10^{-6} n \log n \implies \frac{n}{\log n} = 2 \cdot 10^6$$

There is no elementary closed form solution to such an equation, so let's start up Sage and find a root of that equation numerically:

```
sage: crossover = find_root(x/log(x, 2) == 2*10^6, 0, 10^8)
sage: crossover
51220432.180321828
sage: 5*crossover^2*10^-10
1311766.336369474
sage: _ / 86400
15.18248074501706
```

So the first scenario is faster until a problem size of about $5.1 \cdot 10^7$, or 51 millions, at which point both settings take a little more than 15 days to run.

**Exercise 5**

Let $x \in \mathbb{R}$ and $n \in \mathbb{N}_{\geq 1}$. To show: $\lfloor \lfloor x \rfloor / n \rfloor = \lfloor x/n \rfloor$. (The other part follows by setting $x = a/b$.)

Using a variant of division with remainder, we can write $x = an + b$ with $a \in \mathbb{Z}$ and $0 \leq b < n$. Then we have

$$\left\lfloor \frac{\lfloor x \rfloor}{n} \right\rfloor = \left\lfloor \frac{an + \lfloor b \rfloor}{n} \right\rfloor = \left\lfloor a + \underbrace{\frac{\lfloor b \rfloor}{n}}_{\in [0,1)} \right\rfloor = a$$

Similarly,

$$\left\lfloor \frac{x}{n} \right\rfloor = \left\lfloor a + \underbrace{\frac{b}{n}}_{\in [0,1)} \right\rfloor = a$$

and so $\lfloor \lfloor x \rfloor / n \rfloor = a = \lfloor x/n \rfloor$.