UNIVERSITY OF LIVERPOOL

Department of Computer Science
Sebastian Wild

Date: 2020-02-05
Version: 2020-02-06 23:44

Tutorial 1 for COMP 526 – Applied Algorithmics, Winter 2020

—including solutions—

It is highly recommended that you first try to solve the problems on your own before consulting the sample solutions provided below.

Problem 1 (Orders of magnitude)

Order the following functions with respect to their asymptotic order of magnitude (i.e., their Θ-class).

$$n,\ \sqrt{n},\ n^{1.5},\ n^2,\ n\lg n,\ n\lg\lg n,\ n\lg^2 n,\ n\lg(n^2),\ \frac{2}{n},\ 2^n,\ 2^{n/2},\ 37,\ n^3,\ n^2\lg n.$$

Solutions for Problem 1 (Orders of magnitude)

The functions in our example can be ordered as follows:

$$\frac{2}{n},\ 37,\ \sqrt{n},\ n,\ n\lg\lg n,\ n\lg n,\ n\lg(n^2),\ n\lg^2 n,\ n^{1.5},\ n^2,\ n^2\lg n,\ n^3,\ 2^{n/2},\ 2^n.$$

Lets check a couple of pairs of functions.

- For example, $n\lg n$ and $n\lg(n^2)$. Note that $n\lg(n^2) = 2n\lg n$ (since in general $\lg a^b = b\lg a$). Thus now the relation $n\lg n \le 2n\lg n$ is more apparent. Note also that $n\lg n = O(2n\lg n)$ and vice versa (multiplicative constants do not matter in asymptotic consideration, right?), which means that in fact $n\lg n = \Theta(2n\lg n)$.

- Consider also $2^{n/2}$ and 2^n. Is $2^{n/2} = O(2^n)$?

  Let's compute the ratio $2^{n/2}/2^n = 2^{n/2-n} = 2^{-n/2} = (\frac{1}{2})^{n/2}$. This ratio tends to 0 when $n \to \infty$, i.e., formally

  $$\lim_{n\to\infty} \frac{2^{n/2}}{2^n} \;=\; 0.$$

  Since 0 is bounded, we find $2^{n/2} = O(2^n)$, and indeed $2^{n/2} = o(2^n)$.

  Now does the opposite relation also hold, i.e., $2^n = O(2^{n/2})$? The answer is No. Again, we consider the limit of the ratio, but this time the other way round: $2^n/2^{n/2} = 2^{n/2} \to \infty$ as $n \to \infty$. This is clearly *not* bounded, and so $2^n \neq O(2^{n/2})$. Indeed, we have $2^n = \omega(2^{n/2})$.

  $\rightsquigarrow$ Different constants in the exponent define different Θ-classes!

- How about proving $n \lg\lg n = \mathcal{O}(n \lg n)$.

  Again, consider the ratio

  $$\lim_{n\to\infty} \frac{n \lg(\lg(n))}{n \lg n} \;=\; \lim_{n\to\infty} \frac{\lg(\lg(n))}{\lg n}.$$

  Here, no direct simplification seems to apply, but it is an "$\frac{\infty}{\infty}$" instance of *L'Hôpital's Rule* ↗. So we try taking derivative of numerator and denominator: $\frac{d}{dn}\lg n = \frac{d}{dn}\frac{\ln n}{\ln 2} = \frac{1}{\ln 2}\cdot\frac{1}{n}$ and using the *chain rule of calculus* ↗:

  $$\frac{d}{dn}\lg(\lg n) \;=\; \frac{d}{dn}\frac{\ln(\frac{\ln n}{\ln 2})}{\ln 2} \;=\; \frac{1}{\ln 2}\cdot\frac{1}{\frac{\ln n}{\ln 2}}\cdot\frac{1}{n\ln 2} \;=\; \frac{1}{\ln 2}\cdot\frac{1}{n\ln(n)}.$$

  So the ratio of derivatives is

  $$\lim_{n\to\infty} \frac{\frac{1}{n\ln(n)}}{\frac{1}{n}} \;=\; \lim_{n\to\infty} \frac{1}{\ln(n)} \;=\; 0,$$

  which is clearly bounded, thereby proving $n \lg\lg n = \mathcal{O}(n \lg n)$.

Problem 2 (Loop invariants)

There are two integral[1] parts of integer division: *the quotient* and *the remainder*. For two integers n, $k > 0$ the quotient (or result) of the integer division "n div k" is defined as the largest integer m with $m \cdot k \leq n$. The remainder of the division is defined as $r = n - m \cdot k$. Note that $0 \leq r < k$. The value r is also known as the result of *modulo* operation, written "$r = n \bmod k$".

Example: $10 \operatorname{div} 3 = 3$ and $10 \bmod 3 = 1$,
$13 \operatorname{div} 5 = 2$ and $13 \bmod 5 = 3$.

[1] pun intended

Apply the *invariant method* to prove the correctness of the following function $Mod(n, k)$, which is supposed to compute $n \bmod k$, where n and k are two positive integer input parameters of the function.

```
1  procedure Mod(n, k)
2  // Input: positive integers n, k.
3  // Output: value of n mod k.
4  t := n
5  while t ≥ k
6      t := (t − k)
7  end while
8  return t
```

Solutions for Problem 2 (Loop invariants)

Note that from the definition of *modulo* operation one can conclude that for two positive integers $x_1 \neq x_2$ with $x_1 = y_1 \cdot k + r$ and $x_2 = y_2 \cdot k + r$ for two positive integers y_1 and y_2, we also have

$$x_1 \bmod k \;=\; x_2 \bmod k \;=\; r.$$

Note that this holds true also when, e.g., $y_1 = y_2 + 1$. (1)

Loop condition: The loop condition is $cond \equiv t \geq k$.

Loop invariant: This is a statement that must be true just before each iteration (execution of the body of the loop). Moreover, the invariant plus the stopping condition should enforce the expected postcondition. Note that choosing a sufficiently strong loop invariant can be tricky and is often a matter of experience. But your ability to choose such an invariant shows you that you really understand what the loop's goal is.

In our example, a sufficient invariant is $I \equiv ((t \bmod k) = (n \bmod k) = r \wedge t \geq 0)$.

The actual proof is done by mathematical induction. We first check that the invariant is true before the first iteration (induction basis) and later we show that any consecutive iteration re-establishes the invariant.

Induction basis: Before the first iteration, we have $t = n$, so the invariant is (trivially) satisfied.

Inductive step: Assume that at the beginning of the ith iteration the invariant is satisfied. Also note that the only instruction executed within the loop is $t := (t - k)$. If we had $t = y_1 \cdot k + r$ before the iteration, we will have $t = y_2 \cdot k + r$ after the iteration, where $y_2 = y_1 - 1$. Thus by observation (1) we conclude that also at the end of the ith iteration (i.e., just before the $(i + 1)$st iteration), we have $(t \bmod k) = (n \bmod k) = r$. Moreover, at the beginning of the iteration, we had $t \geq k$, so we now have $t \geq 0$. Hence, the invariant is also satisfied at the beginning of the next iteration.

Finally, from the negated loop condition (i.e., the stopping condition) $\neg cond \equiv t < k$ and the invariant $I \equiv ((t \bmod k) = (n \bmod k) = r \wedge t \geq 0)$ we conclude that indeed function $Mod(n, k)$ computes the value of $n \bmod k$ in t (and returns it).