Applied Algorithmics COMP526 – tutorial 1

L.A. Gasieniec and D. Cartwright

1 Questions

1.1 Order of magnitude

Order the following functions with respect to their asymptotical order of magnitude.

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

1.2 Invariant method

There are two integral parts of integer division: *the result* and *the remainder*. I.e., if we take two integer numbers $n, k > 0$ the result of integer division n div k is defined as the largest non-negative integer x, s.t., $x \cdot k \leq n$. And the remainder of the division is defined as $r = n - x$. Note that $0 \leq r < k$. The value r is also known as the result of *modulo* operation, i.e., $r = n \bmod k$.

Example: 10 div 3 = 3 and 10 mod 3 = 1, also 13 div 5 = 2 and 13 mod 5 = 3.

Apply the *invariant method*, see lecture notes from week 1, to prove the correctness of a function $Mod(n, k)$ expected to compute the value of n mod k, where n and k are two positive integer input parameters of the function.

function $Mod(n, k$: *integer*): *integer*;
Input: positive integers n, k.
Output: value of n mod k.
    $temp \leftarrow n$;
    while $temp \geq k$ **do**
        $temp \leftarrow (temp - k)$;
    end-while;
    return *temp*;
end-function.

2 Solutions

2.1 Order of magnitude

The functions in our example can be ordered as follows:

$$2/n, 37, \sqrt{n}, n, n\log\log n, n\log n, n\log(n^2), n\log^2 n, n^{1.5}, n^2, n^2\log n, n^3, 2^{n/2}, 2^n,$$

Lets check a couple of pairs of functions.

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

Consider also $2^{n/2}$ and 2^n. Is $2^{n/2} = O(2^n)$? Lets take $n_0 = 1$ and $c = 1$, i.e., we have to check whether $2^{n/2} < 1 \cdot 2^n$, for all $n = 1, 2, 3, ...$ How do we do this? Since both functions are growing functions (their values get larger when n gets larger) we can take logarithm (log) from both sides of inequality, where $\log(2^{n/2}) = n/2 \cdot \log 2 = n/2$. And $\log(2^n) = n \cdot \log 2 = n$. And clearly $n/2 < n$ thus $2^{n/2} = O(2^n)$. Now does the opposite relation holds, i.e., $2^n = O(2^{n/2})$?. The answer is NO!. (Proof by contradiction) Assume that it is possible to find positive constants n_0 and c, s.t., $2^n < c \cdot 2^{n/2}$, for all $n > n_0$. Apply logarithm on both sides obtaining

$$\log(2^n) < \log(c \cdot 2^{n/2})$$

which is equivalent with

$$n < \log c + n/2$$

since $\log(a \cdot b) = \log a + \log b$. But note that $\log c$ is a constant and for large n is clearly larger then $n/2$ plus some constant, i.e., we get contradiction. Thus constants matter if they are in the exponent!!!

2.2 Invariant method

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

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

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

Stop condition refers to negation of the loop condition. Thus in our case the stop condition $F(temp) \equiv$ **not** $(temp \geq k) \equiv temp < k$.

Loop invariant must be true just before each iteration (execution of the body of the loop). More over the invariant plus the stop condition should enforce the expected solution. Note that choosing the right invariant is very often a matter of experience. But your ability to choose such an invariant shows that you really understand what the loop's task is. Anyway, we decide to chose the invariant $I(temp) \equiv ((temp \bmod k) = (n \bmod k) = r)$.

The actual proof is done as in the mathematical induction. We first check that the invariant is true before the first iteration (basis step) and later we show that any consecutive iteration does not violate the invariant.

Basis step Indeed, before the first iteration $temp = n$ thus the invariant is satisfied.

Inductive step Assume that at the beginning of iteration i the invariant is satisfied. Also note that the only instruction executed within the loop is $temp \leftarrow (temp - k)$. And if before the iteration $temp = y_1 \cdot k + r$, after the iteration $temp = y_2 \cdot k + r$, where $y_1 = y2 + 1$. Thus due to observation (1) we conclude that also on the conclusion of iteration i (i.e., just before iteration $i + 1$) the invariant is satisfied too.

Finally, from the stop condition $F(temp) \equiv temp < k$ and the invariant $I(temp) \equiv ((temp \bmod k) = (n \bmod k) = r)$ we conclude that indeed function $Mod(n, k)$ computes the value of $n \bmod k$.