58131 Data Structures

I exercise, week 40/2003, English translation

Exercise I.1: Show that $\mathcal{O}$-notation is *transitive*: if $f(n) = \mathcal{O}(g(n))$ and $g(n) = \mathcal{O}(h(n))$, then also $f(n) = \mathcal{O}(h(n))$.

How could this result be used?

Exercise I.2: Let $f, g\colon \mathbb{N} \to \mathbb{R}$ be nonnegative functions for which the condition

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

holds. Prove:

(a) Every function in $\mathcal{O}(f(n))$ is also in $\mathcal{O}(g(n))$.

(b) On the other hand, for example $g(n)$ itself is not in $\mathcal{O}(f(n))$.

That is, $\mathcal{O}(f(n)) \subsetneq \mathcal{O}(g(n))$.

How could this way of comparing functions $f(n)$ and $g(n)$ be uselful?

Exercise I.3: The table below contains pairs $f(n), g(n)$ of functions for which either $f(n) = \mathcal{O}(g(n))$ or $g(n) = \mathcal{O}(f(n))$ holds, but not both. Which one?

$f(n)$	$g(n)$
$\frac{n^2-n}{2}$	$6n$
$n + 2\sqrt{n}$	n^2
$n + n\log n$	$n\sqrt{n}$
$n^2 + 3n + 4$	n^3
$n \log n$	$\frac{n\sqrt{n}}{2}$
$n + \log n$	$\sqrt{n}$
$2(\log n)^2$	$\log n + 1$

Exercise I.4: *Bubblesort* (kuplalajittelu) sorts an array $A[1 \ldots N]$ given as input in the following way:

```
1: for all i := 1 up to N − 1 do
2:    for all j := N down to i + 1 do
3:       if A[j] < A[j − 1] then
4:          swap the contents of locations A[j] and A[j − 1] with each other
5:       end if
6:    end for
7: end for
```

(a) Choose for the inner loop in lines 2–6 an invariant which helps you in part (b). Prove also that your invariant really does hold.

(b) Choose for the whole loop in lines 1–7 an invariant which allows you to show that the algoritm works correctly. Prove also that your invariant really does hold.

(c) Count how many times line 3 is executed for a given input length N.

Why is this count particularly interesting?

(d) Would the algorithm still work, if its line 4 was changed to read "swap the contents of locations $A[j]$ and $A[\underline{i}]$ with each other" instead, where the change is underlined?

Justify your answer with the invariant in part (b).

Exercise I.5: Consider the following algorithmic problem:

The algorithm receives a subroutine named outo(p: $\mathbb{Z}$): $\mathbb{Z}$ as a parameter. We only know that it is strictly ascending; that is, outo$(m) <$ outo$(m+1)$ holds for all $m \in \mathbb{Z}$.

The algorithm must return the $q \in \mathbb{Z}$ for which outo$(q) = 0$, if such a q exists. Otherwise it must return "none".

(a) Develop an algorithm which finds the solution q using $\mathcal{O}(\log_2 |q|)$ calls to the subroutine outo, if q exists.

Prove that your algorithm really solves the problem and meets this extra condition. Use the methods in the book.

(b) How many times does your algorithm call the subroutine outo if q does not exist?

(c) If the subroutine outo runs in constant time, then what is the total maximum time used by your algorithm?

(d) If the subroutine outo satisfies only outo$(m) \leq$ outo$(m+1)$ for all $m \in \mathbb{Z}$, then does your algorithm still work? Justify your answer.

(Total number of exercises: 5 pcs.)