Date: 2025-10-16
Version: 2025-10-16 20:41

Exercise Sheet 1 zur Vorlesung Effiziente Algorithmen (Winter 2025/26)

Abgabe: *Bis 2025-10-24 18:00, on ILIAS.*

Exercise Policy

- The exercise problems take up the material from lecture. Working on them deepens understanding and increases proficiency, and I consider them an integral part of the course.
- You will require at least 50% of the total points to take the exam.
- You may collaborate on the problems and hand in your solutions as a group of 2–4 students.
- If you make use of external sources, make sure to clearly mark the used parts as such, including unambiguous references to the original sources. In case of websites, at least add the access date and use permanent links whenever possible (e. g., for Wikipedia). You may cite any publicly accessible sources, but take their credibility into account: you are responsible to verify what you cite.

  This applies even more so to generative AI answers. The tasks are designed to be solved without AI tools; don't deprive yourself of the learning experience.

  Collaboration between groups on conceptual solutions to the exercises is fine, but each group must work out a detailed submission individually. In these cases, add a quick statement with which groups you joined forces to work on a given problem.

  Plagiarism is not tolerated in academia, be it external sources, other groups' submissions, or genAI answers; severe cases can lead to expulsion from university.

1. Aufgabe

10 + 20 Punkte

Prove the following statements using mathematical induction.

a) For all $n \in \mathbb{N}_0$, we have $\sum_{k=0}^{n} q^k = \frac{q^{n+1} - 1}{q - 1}$.

b) For all $n \in \mathbb{N}_0$, we can solve the *Towers of Hanoi* puzzle with n disks in $2^n - 1$ steps.

A good description of the Tower of Hanoi puzzle can be found on Wikipedia
`https://en.wikipedia.org/wiki/Tower_of_Hanoi`

Tipp: Consider a recursive algorithm for solving the puzzle which moves a tower of a given height h from a given start pole to a given target pole.

Given a sequence of numbers $T(n)$ defined recursively by

$$T(n) = \begin{cases} 3, & \text{for } n = 0; \\ T(n-1) + 4, & \text{for } n \geq 1. \end{cases} \tag{1}$$

a) Compute the first 6 elements of $T(n)$, i.e., $T(0)$, $T(1)$, $T(2)$, $T(3)$, $T(4)$, and $T(5)$.

b) Make an educated guess about the general pattern that this sequence follows. Write this guess as a *closed form* for $T(n)$, i.e., a formula for $T(n)$ without recursive reference to T.

c) Now formally prove the correctness of your guess using mathematical induction.

2. Aufgabe

30 Punkte

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 the *modulo* operation, written "$r = n \bmod k$".

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

Apply the *decreasing potential method* to prove that the following function *Mod*(n, k)

[1] pun intended

always terminates when called with parameters $n \in \mathbb{N}$ and $k \in \mathbb{N}$, where $\mathbb{N} = \{1, 2, 3, \ldots\}$.

```
procedure $Mod(n, k)$
    // Input: positive integers $n, k$.
    // Output: value of $n$ mod $k$.
    $t := n$
    while $t \geq k$
        $t := (t - k)$
    end while
return $t$
```

3. Aufgabe

10 + 40 Punkte

The following algorithm calculates the power x^n for $n \in \mathbb{N}_0$. Here, `div` denotes integer division and `mod` the remainder of the division.

```
Algorithmus 1 : Power
Input : $x, n \in \mathbb{N}$
$p := 1$;
if $x \neq 0$ then
    while $n \neq 0$ do
        if $n \bmod 2 == 1$ then
            $p := p \cdot x$;
        $n := n \operatorname{div} 2$;
        $x := x \cdot x$;
else
    $p := 0$;
return $p$;
```

a) Run the algorithm for the values $x = 2$, $n = 7$. Give the values of x, n, p after each iteration of the loop.

b) Verify the correctness of the algorithm using a loop invariant. For each line of code, provide appropriate preconditions and postconditions, and state the loop invariant.