Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer
Johannes Lengler, Gleb Novikov, Chris Wendler, Ulysse Schaller

28. September 2020

Algorithms & Data Structures — Exercise sheet 2 — HS 20

Exercise Class (Room & TA): ____________________

Submitted by: ____________________

Peer Feedback by: ____________________

Points: _____

The solutions for this sheet are submitted at the beginning of the exercise class on October 5th.

Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

Exercise 2.1 *Bounding an inductive sequence* **(1 point)**.

Consider a sequence of natural numbers $(a(n))_{n\in\mathbb{N}_0}$ that is defined by

$$a(0) = 1, \qquad a(1) = 2, \qquad a(n) = 2a(n-1) + 3a(n-2) \quad \text{for } n \geq 2.$$

In this exercise, we will show that this sequence grows exponentially fast.

a) Use induction to show that for all $n \geq 4$, we have $a(n) \geq e^n$.

b) Show that for all $n \geq 0$, $a(n) \leq \mathcal{O}(3^n)$.

c) Suppose that we redefine the starting values of the sequence, i.e. that for some $b, b' \in \mathbb{N}$, $(a(n))_{n\in\mathbb{N}}$ is given by

$$a(0) = b, \qquad a(1) = b', \qquad a(n) = 2a(n-1) + 3a(n-2) \quad \text{for } n \geq 2.$$

Show that for any choices of $b, b' \in \mathbb{N}$ we still have $a(n) \leq \mathcal{O}(3^n)$ for all $n \geq 0$.

Exercise 2.2 *Iterative squaring.*

In this exercise you are going to develop an algorithm to compute powers a^n, with $a \in \mathbb{Z}$ and $n \in \mathbb{N}$, efficiently. For this exercise, we will treat multiplication of two integers as a single elementary operation, i.e., for $a, b \in \mathbb{Z}$ you can compute $a \cdot b$ using one operation.

a) Assume that n is even, and that you already know an algorithm $A_{n/2}(a)$ that efficiently computes $a^{n/2}$, i.e., $A_{n/2}(a) = a^{n/2}$. Given the algorithm $A_{n/2}$, design an efficient algorithm $A_n(a)$ that computes a^n.

b) Let $n = 2^k$, for $k \in \mathbb{N}_0$. Find an algorithm that computes a^n efficiently. Describe your algorithm using pseudo-code.

c) Determine the number of elementary operations (i.e., integer multiplications) required by your algorithm for part b) in $\mathcal{O}$-notation. You may assume that bookkeeping operations don't cost anything. This includes handling of counters, computing $n/2$ from n, etc.

d) Let $\text{Power}(a, n)$ denote your algorithm for the computation of a^n from part b). Prove the correctness of your algorithm via mathematical induction for all $n \in \mathbb{N}$ that are powers of two.

In other words: show that $\text{Power}(a, n) = a^n$ for all $n \in \mathbb{N}$ of the form $n = 2^k$ for some $k \in \mathbb{N}_0$.

*e) Design an algorithm that can compute a^n for a general $n \in \mathbb{N}$, i.e., n does not need to be a power of two.

Hint: *Generalize the idea from part a) to the case where n is odd, i.e., there exists a $k \in \mathbb{N}$ such that $n = 2k + 1$.*

*f) Prove correctness of your algorithm in e) and determine the number of elementary operations in $\mathcal{O}$-Notation. As before, you may assume that bookkeeping operations don't cost anything.

Exercise 2.3 *$\mathcal{O}$-Notation.*

a) Write the following in the asymptotic $\mathcal{O}$-notation. Your answer should be simplified as much as possible. Unless otherwise stated, we assume $N = \mathbb{N} = \{1, 2, 3, \dots\}$. You do not need to check that the involved functions take values in $\mathbb{R}^+$.

1) $5n^3 + 40n^2 + 100$.

2) $2n \log_3 n^4$ with $N = \{2, 3, 4, \dots\}$.

b) Prove that if $f_1(x), f_2(x) \leq \mathcal{O}(g(x))$, then $f_1(x) + f_2(x) \leq \mathcal{O}(g(x))$.

c) Let $f_1(x), f_2(x), g(x) > 0$. Prove or disprove the following.

1) If $f_1(x), f_2(x) \leq \mathcal{O}(g(x))$ then $\frac{f_1(x)}{f_2(x)} \leq \mathcal{O}(1)$.

2) If $f_1(x) \leq \mathcal{O}(g(x))$ and $f_2(x) \leq \mathcal{O}(\frac{1}{g(x)})$, then $f_1(x)f_2(x) \leq \mathcal{O}(1)$.

Exercise 2.4 *Towers of Hanoi* **(2 points)**.

In this exercise you should design a recursive divide-and-conquer algorithm for solving the *Tower of Hanoi* puzzle. The puzzle consists of three rods A, B and C, and n disks of different sizes, which we number from 1 (smallest) to n (largest). The disks can slide onto any rod. The puzzle starts with all the disks stacked in ascending order (largest on bottom, smallest on top) on rod A (see Figure 1).

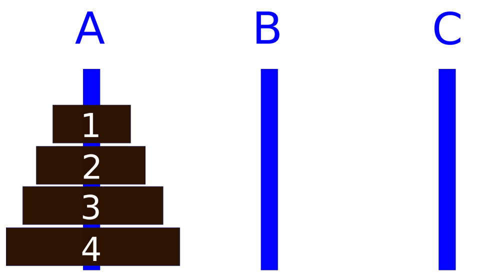

Figure 1: Initial state of the Tower of Hanoi game for $n = 4$.

Now, the goal is to move all the disks to rod C. When moving the disks the following rules must be obeyed:

1. Only one disk can be moved at a time.
2. Each move consists of taking the uppermost disk from one of the stacks and placing it on top of another stack or an empty rod.
3. No larger disk may be placed on a smaller disk.

a) Develop an algorithm that solves the problem for $n = 1$.

b) Assume that you have an algorithm `Move(source, target, spare)` that can move $n - 1$ disks from a source rod to a target rod using a spare rod and use it to solve the puzzle with n disks.

c) Make use of the insights you gained in a) and b) in order to complete the pseudo-code of `SolveHanoi`. Calling `SolveHanoi`(A, C, B, n) should solve the puzzle.

Algorithm 1 `SolveHanoi(source, target, spare,` n`)`

```
if        ...        then
    SolveHanoi(    ...    ,    ...    ,    ...    , n − 1)
    Move the uppermost disk from    ...    to    ...
    SolveHanoi(    ...    ,    ...    ,    ...    , n − 1)
```

d) Proof the correctness of `SolveHanoi`(A, C, B, n) for all $n \in \mathbb{N}$ by induction.

e) How many moves are performed by `SolveHanoi`(A, C, B, n) in order to solve the puzzle?

Hint: Let T_n denote the number of moves required by `SolveHanoi`(A, C, B, n) and T_{n-1} the number of moves required by `SolveHanoi`($*, *, *, n - 1$) (where $*$ is a placeholder). Think about how T_n and T_{n-1} relate to each other. The relationship between T_n and T_{n-1} is called a *recurrence relation.*