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

30. September 2019

Algorithms & Data Structures — Exercise sheet 2 — HS 19

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 7th.

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

Exercise 2.1 *Iterative squaring* **(1 point)**.

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 $k \in \mathbb{N}$. 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 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 Power$(a, n) = a^n$ for all $n \in \mathbb{N}$ of the form $n = 2^k$ for some $k \in \mathbb{N}$.

*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 2.2 *Induction.*

a) Prove via mathematical induction that for any positive integer n,

$$(1+x)^n = \sum_{i=0}^{n} \binom{n}{i} x^i \,,$$

where n choose i is defined by

$$\binom{n}{i} := \frac{n!}{(n-i)!\, i!} = \frac{n \cdot (n-1) \cdots (n-i+1)}{i \cdot (i-1) \cdots 1}.$$

We set $\binom{n}{k} := 0$ for $k > n$ and for $k < 0$. Notice that $\binom{n}{0} = 1$ because $0! = 1$.

Hint: *Show first the identity*

$$\binom{n}{i} + \binom{n}{i-1} = \binom{n+1}{i}.$$

It can be obtained by fractional arithmetic and will be useful for the proof.

b) Given is a map that is divided into regions by n (pairwise different) straight lines. You want to color the regions on the map (i.e., the areas bordered by the lines), such that no two neighboring regions (i.e., regions that share a common segment of a line as a border) get the same color.

Prove by mathematical induction on n, that you can color every such map with 2 colors.

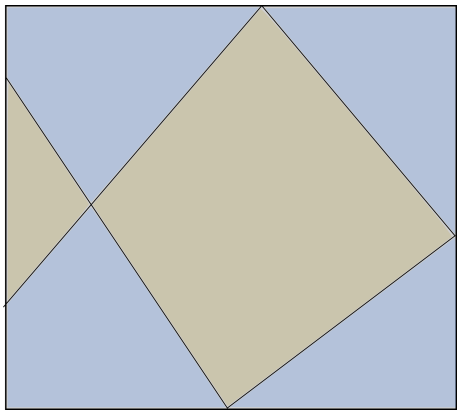

Figure 1: An example map. Defined by $n = 4$ straight lines. The resulting 6 regions are colored by two colors.

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

Recall the definition of $\mathcal{O}$-notation introduced in the lecture:

$$\mathcal{O}(g) = \{f : N \to \mathbb{R}^+ | \text{ There exists a constant } C > 0 \text{ such that for all } n \in N,\ f(n) \leq C \cdot g(n)\}\,.$$

Prove your statements directly with this definition. In particular, you should *not* use theorems from Exercise sheet 0.

For example, to show that $4n + 1 \in \mathcal{O}(n)$ you would look for a constant C such that $4n + 1 \leq Cn$ for all $n > 1$. In this example $C = 5$ is such a constant.

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) $1^2 + 2^2 + 3^2 + \ldots + n^2$.

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

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 *Karatsuba algorithm for polynomial multiplication* **(2 points)**.

In this exercise you should design an efficient algorithm for multiplying polynomials with integer coefficients. For this exercise, we will treat addition, subtraction, and multiplication of two integers as a single elementary operation. As before, you may ignore any further bookkeeping costs.

a) How many elementary operations are necessary to add or subtract two polynomials of degree k?

b) Assume that you already know an efficient algorithm A_k for the multiplication of two polynomials of degree at most $k-1$. Design an efficient algorithm A_{2k} for multiplying two polynomials of degree at most $2k - 1$ that calls A_k at most three times as a subroutine and uses at most $\mathcal{O}(k)$ additional elementary operations. You should describe your algorithm in pseudo-code.

Hint: *Let the input polynomials $P(x), Q(x)$ of degree at most $2k - 1$ be*

$$P(x) = P_0 x^0 + P_1 x^1 + \cdots + P_{2k-1} x^{2k-1},$$
$$Q(x) = Q_0 x^0 + Q_1 x^1 + \cdots + Q_{2k-1} x^{2k-1}.$$

Then write the polynomials as $P(x) = x^k P^*(x) + P^{**}(x)$ and $Q(x) = x^k Q^*(x) + Q^{**}(x)$, where P^*, P^{**}, Q^* and Q^{**} are polynomials of degree at most $k - 1$. Now use the same trick as in Karatsuba's algorithm.

c) Provide a bound for the amount of elementary operations $T(2k)$ required by your algorithm A_{2k} in terms of the amount of elementary operations $T(k)$ required by the algorithm A_k. That is, give a bound of the form $T(2k) \leq a \cdot T(k) + b(k)$. Give as precise bound as possible.

d) Let $k = 2^s$, for $s \in \mathbb{N}_0$ (where $\mathbb{N}_0 = \{0, 1, 2, 3, 4, \ldots\}$). Find an algorithm Product(k, P, Q) for the efficient multiplication of two polynomials P, Q of degree at most $k - 1$. Describe you algorithm using pseudo-code.

e) Provide the asymptotic amount of operations required by your algorithm Product(k, P, Q) in terms of k (i.e., in $\mathcal{O}$-Notation with respect to the degree k). Prove your answer with induction.

f) Prove the correctness of your algorithm Product(k, P, Q) via mathematical induction. In other words: show that Product$(k, P, Q) = P \cdot Q$ for all $k \in \mathbb{N}$ such that $k = 2^s$, where $s \in \mathbb{N}_0$.