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

23 September 2019

Algorithms & Data Structures — Exercise sheet 1 — 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 September 30th.

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

Exercise 1.1 *Primality* **(1 point)**.

Recall that a natural number $n > 1$ is called *prime* if it cannot be written as the product of two (strictly) smaller natural numbers.

Consider the following algorithm that, given a natural number $n > 1$ as input, decides whether n is prime or not:

Algorithm 1 SimplePrimalityTest(n)

```
k ← 2
while DIVIDES(k, n) is false do
    k ← k + 1
if k = n then
    return "n is prime"
else
    return "n is not prime"
```

where DIVIDES(k, n) is a procedure that returns true if and only if k divides n without a remainder.

Answer the following questions:

a) Let $T(n)$ be the number of calls of DIVIDES that this algorithm makes on the input n. Draw a graph of $T(n)$ for $n = 2, 3, \ldots, 30$ (i.e., x-axis: n, y-axis: $T(n)$).

b) How many calls of DIVIDES does this algorithm make in the worst case? That is, what is the largest possible number of calls of the procedure DIVIDES that this algorithm can make (in terms of n)? Which numbers n correspond to this case?

c) How many calls of DIVIDES does this algorithm make in the best case? That is, what is the smallest possible number of calls of the procedure DIVIDES that this algorithm can make (in terms of n)? Which numbers n correspond to this case?

d) Show that there exists a function $f(n) < n$ such that if k exceeds $f(n)$, then n is prime. What is the smallest possible function $f(n)$ of the form $f(n) = n^\alpha$ (where $0 < \alpha < 1$) with this property?

Let f be the smallest possible function of the form $f(n) = n^\alpha$ from point d). Consider the following algorithm:

Algorithm 2 ImprovedPrimalityTest(n)

```
k ← 2
while k ≤ f(n) and DIVIDES(k, n) is false do
    k ← k + 1
if k > f(n) then
    return "n is prime"
else
    return "n is not prime"
```

e) Answer the questions from points a), b) and c) for the second algorithm. You can assume that in the while loop we do not call DIVIDES(k, n) if $k > f(n)$.

f)* Is the running time of the first algorithm polynomial in the size of the input? That is, is there any constant integer $m > 0$ such that the (worst-case) running time of the algorithm is $\mathcal{O}(l^m)$, where l is the length of the input? What about the second algorithm?

Exercise 1.2 *Induction.*

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

$$2^1 + 2^2 + \cdots + 2^n = 2^{n+1} - 2\,.$$

b) Prove via mathematical induction that for all integers $n \geq 5$,

$$2^n > n^2\,.$$

Exercise 1.3 *$\mathcal{O}$-Notation.*

a) Prove or disprove the following statements:

1) $n^2 \in \mathcal{O}(3n^4 + n^2 + n)$.
2) $\log_7(n^8) \in \mathcal{O}(\log(n^{\sqrt{n}}))$.
3) $n^{1/3} \in \mathcal{O}(\frac{n}{\log n})$.
4) $\sum_{k=0}^{n} k \in \mathcal{O}(n \log n)$.

b) Place the following functions in order such that if f appears before g, it means that $f \in \mathcal{O}(g)$. If for some functions it holds that $f \in \mathcal{O}(g)$ and $g \in \mathcal{O}(f)$, please indicate so.

$$n^2 + 2n + 1,\ \ n\sum_{k=0}^{n} k,\ \ \frac{n}{\ln n},\ \ n\ln(n^n),\ \ \sqrt{n}\ln(n),\ \ n\ln(n^2),\ \ \sum_{k=0}^{n} k^2,\ \ n\ln(2^n)$$

c)* Prove the following statements about $n!$:

1) $n! \leq n^n$.

2) $\ln(n!) \in \mathcal{O}(n \ln n)$.

3) $\left(\frac{n}{2}\right)^{n/2} \leq n!$.

4) $n \ln n \in \mathcal{O}(\ln(n!))$.

d)* Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$. Is it always true that either $g \in \mathcal{O}(f)$ or $f \in \mathcal{O}(g)$ or both? What if both f and g are strictly increasing?

Exercise 1.4 *Subtraction algorithm* **(2 points)**.

The goal of this exercise is to design a subtraction algorithm of natural numbers that uses additions, comparisons and some simple operations with digits. Consider two natural n-digit numbers a and b. Denote by $a_{n-1} \dots a_0$ and $b_{n-1} \dots b_0$ their decimal expressions.

a) Describe an algorithm that compares a and b. That is, your algorithm should output true if $a \geq b$ and false otherwise. You may assume that you have access to an elementary operation which compares two digits with each other.

Let $\bar{b} = \underbrace{99\dots9}_{n \text{ times}} - b$. Note that $\bar{b}$ is an n-digit number with digits $\bar{b}_i = 9 - b_i$ for $i = 0, \dots, n-1$.

b) Show that $a - b + 10^n$ can be computed using one computation of $\bar{b}$ and two additions of (at most) $(n+1)$-digit numbers.

c) Show that if $a \geq b$, then $a - b$ can be obtained from $a - b + 10^n$ by removing the first digit.

d) Using points a), b) and c), design a subtraction algorithm of natural numbers. Specifically, given two decimal representations of n-digit numbers a and b as input, your algorithm should output the decimal representation of $a - b$. Here we assume that negative numbers are represented as decimal expressions that come after the '$-$' sign.

e) How many elementary operations does your algorithm take in the worst case? For the addition operations, you may use the general bound from the lecture for adding $(n+1)$-digit numbers. Elementary operations are comparing two digits, adding two digits (possibly with a carry bit), removing the leading digit, inverting a digit (i.e., computing $\bar{b}_i$) and writing the '$-$' sign before the decimal representation of the number.