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

21 September 2020

Algorithms & Data Structures — Exercise sheet 1 — 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 September 28th.

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

Exercise 1.1 *Proper Power* (2 points).

A natural number $n > 1$ is called a *proper power* if there exist two natural numbers $a, b > 1$ such that n can be written as $n = a^b$.

Consider the following algorithm that, given a natural number $n > 1$ as input, decides whether n is a proper power or not. Note that the algorithm depends on a function f that determines how many for-loop iterations are performed at most. For now assume $f(n) = n$.

Algorithm 1 ProperPowerTest(n)

```
for $a = 2, \dots, f(n)$ do
    $b \leftarrow 2$
    while $a^b < n$ do
        $b \leftarrow b + 1$
    if ISPOWER($n, a, b$) then
        return "$n$ is a proper power"
return "$n$ is not a proper power"
```

The procedure ISPOWER(n, a, b) is called **once per iteration of the for-loop** and returns true if and only if $n = a^b$.

Answer the following questions:

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

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

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

d) The for-loop in Algorithm 1 ranges from 2 to $f(n)$. Is it possible to perform fewer than $f(n) = n$ for-loop iterations in Algorithm 1? Determine the smallest value of $f(n)$ such that Algorithm 1 works.

Consider the following algorithm. Note that the algorithm depends on the function $f(n)$, which determines the number of for-loop iterations. For now, please assume that $f(n) = n$.

Algorithm 2 ImprovedProperPowerTest(n)

```
for $b = 2, \dots, f(n)$ do
    $a \leftarrow 2$
    while $a^b < n$ do
        $a \leftarrow a + 1$
    if ISPOWER($n, a, b$) then
      return "$n$ is a proper power"
return "$n$ is not a proper power"
```

e) Find a function f such that $f(n)$ is as small as possible.

Hint: The correct f leads to an algorithm that requires less than $\lceil \sqrt{n} \rceil$ calls of ISPOWER.

f) Answer the questions from points a), b) and c) for the second algorithm using your f from task e).

Exercise 1.2 *Induction.*

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

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}\,.$$

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

Hint: You can use the formula from Exercise 1.2.a.

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

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

Hint: *You can use formulas from Exercise 1.2.a and Exercise 0.1.a.*

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

1) $n! \leq n^n$.
2) $\ln(n!) \leq \mathcal{O}(n\ln n)$.
3) $\left(\frac{n}{2}\right)^{n/2} \leq n!$.
4) $n\ln n \leq \mathcal{O}(\ln(n!))$.

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

Exercise 1.4 *Divisibility check algorithm* **(1 point)**.

Consider the following algorithm that, given an n-digit number $a > 0$ with decimal expression $a_{n-1}\dots a_0$, decides whether a is divisible by 19 or not:

Algorithm 3 DivisibilityCheck(a)

```
if $a < 19$ then
    return "$a$ is not divisible by 19"
else if $a = 19$ then
    return "$a$ is divisible by 19"
else
    $a \leftarrow a_{n-1}\dots a_1$
    $b \leftarrow a + 2 \cdot a_0$
    return DivisibilityCheck($b$)
```

For example, if we feed the number 347 to DivisibilityCheck, it would go over the numbers

$$347 \quad\to\quad 34\cancel{7} + 2\cdot 7 = 48 \quad\to\quad 4\cancel{8} + 2\cdot 8 = 20 \quad\to\quad 2\cancel{0} + 2\cdot 0 = 2,$$

and the algorithm would therefore return "347 is not divisible by 19" since it is called for the last time with the number $2 \neq 19$. The goal of this exercise is to prove the correctness of this algorithm.

a) Show that b is divisible by 19 if and only if a is divisible by 19.

Hint: *Write b in terms of a and a_0 only.*

b) Show that if $a > 19$, then $b < a$.

Hint: *Consider the difference $a - b$.*

c) Prove by mathematical induction that DivisibilityCheck is a correct algorithm.

Hint: *Use mathematical induction in the following variant:*

(a) Prove the statement for some base cases $a = 1, ..., k$.

(b) For $m > k$, prove that if the statement is true for all $a = 1, \ldots, m-1$, then the statement is also true for $a = m$.

Then you can conclude that the statement is true for all natural numbers $a \geq 1$.

d)* Show that the number of recursive calls made by DivisibilityCheck(a) is $\mathcal{O}(\log a)$.