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
François Hublet, Goran Zuzic, Tommaso d'Orsi, Jingqiu Ding

3 October 2022

Algorithms & Data Structures — Exercise sheet 2 — HS 22

The solutions for this sheet are submitted at the beginning of the exercise class on 10 October 2022.

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

You can use results from previous parts without solving those parts.

Exercise 2.1 *Induction.*

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

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

(b) Let x be a real number. Prove via mathematical induction that for every positive integer n, we have

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

where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \,.$$

We use a standard convention $0! = 1$, so $\binom{n}{0} = \binom{n}{n} = 1$ for every positive integer n.

Hint: You can use the following fact without justification: for every $1 \leq i \leq n$,

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

Exercise 2.2 *Growth of Fibonacci numbers* **(1 point)**.

There are a lot of neat properties of the Fibonacci numbers that can be proved by induction. Recall that the Fibonacci numbers are defined by $f_0 = 0, f_1 = 1$ and the recursion relation $f_{n+1} = f_n + f_{n-1}$ for all $n \geq 1$. For example, $f_2 = 1, f_5 = 5, f_{10} = 55, f_{15} = 610$.

(a) Prove that $f_{n+1} \leq 1.75^n$ for $n \geq 0$.

(b) Prove that $f_n \geq \frac{1}{3} \cdot 1.5^n$ for $n \geq 1$.

Asymptotic Notation

When we estimate the number of elementary operations executed by algorithms, it is often useful to ignore constant factors and instead use the following kind of asymptotic notation, also called O-Notation. We denote by $\mathbb{R}^+$ the set of all (strictly) positive real numbers and by $\mathbb{N}$ the set of all (strictly) positive integers.

Definition 1 (O-Notation)**.** Let $n_0 \in \mathbb{N}$, $N := \{n_0, n_0 + 1, \ldots\}$ and let $f : N \to \mathbb{R}^+$. $O(f)$ is the set of all functions $g : N \to \mathbb{R}^+$ such that there exists $C > 0$ such that for all $n \in N$, $g(n) \leq Cf(n)$.

In general, we say that $g \leq O(f)$ if Definition 1 applies after restricting the domain to *some* $N = \{n_0, n_0 + 1, \ldots\}$. Some sources use the notation $g = O(f)$ or $g \in O(f)$ instead.

Instead of working with this definition directly, it is often easier to use limits in the way provided by the following theorem.

Theorem 1 (Theorem 1.1 from the script)**.** *Let $f : N \to \mathbb{R}^+$ and $g : N \to \mathbb{R}^+$.*

- *If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, then $f \leq O(g)$ and $g \not\leq O(f)$.*
- *If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = C \in \mathbb{R}^+$, then $f \leq O(g)$ and $g \leq O(f)$.*
- *If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then $f \not\leq O(g)$ and $g \leq O(f)$.*

The theorem holds all the same if the functions are defined on $\mathbb{R}^+$ instead of N. In general, $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ is the same as $\lim_{x\to\infty} \frac{f(x)}{g(x)}$ if the second limit exists.

The following theorem can also be helpful when working with O-notation.

Theorem 2. *Let $f, g, h : \mathbb{N} \to \mathbb{R}^+$. If $f \leq O(h)$ and $g \leq O(h)$, then*

1. *For every constant $c \geq 0$, $c \cdot f \leq O(h)$.*
2. *$f + g \leq O(h)$.*

Notice that for all real numbers $a, b > 1$, $\log_a n = \log_a b \cdot \log_b n$ (where $\log_a b$ is a positive constant). Hence $\log_a n \leq O(\log_b n)$. So you don't have to write bases of logarithms in asymptotic notation, that is, you can just write $O(\log n)$.

Exercise 2.3 *O-notation quiz.*

(a) Prove or disprove the following statements. Justify your answer.

(1) $n^{\frac{2n+3}{n+1}} = O(n^2)$

(2) $e^{1.2n} = O(e^n)$

(3) $\log(n^4 + n^3 + n^2) = O(\log(n^3 + n^2 + n))$

(b) Find f and g as in Theorem 1 such that $f = O(g)$, but the limit $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ does not exist. This proves that the first point of Theorem 1 provides a necessary, but not a sufficient condition for $f = O(g)$.

Exercise 2.4 *Asymptotic growth of* $\ln(n!)$.

Recall that the factorial of a positive integer n is defined as $n! = 1 \times 2 \times \cdots \times (n-1) \times n$.

a) Show that $\ln(n!) \leq O(n \ln n)$.

Hint: *You can use the fact that* $n! \leq n^n$ *for* $n \geq 1$ *without proof.*

b) Show that $n \ln n \leq O(\ln(n!))$.

Hint: *You can use the fact that* $\left(\frac{n}{2}\right)^{\frac{n}{2}} \leq n!$ *for* $n \geq 1$ *without proof.*

Exercise 2.5 *Triplet Search* **(2 points)**.

Given an array of n integers, and an integer t, design an algorithm that checks if there exists three (not necessarily different) elements of the array a, b, c such that $a + b + c = t$.

(a) Design a simple $O(n^3)$ algorithm.

(b) Suppose that elements of the array are integers in the range $[1, 100n]$, and that $t \leq 300n$. Design a better algorithm with runtime $O(n^2)$ to solve the same problem, assuming the constraints.

Hint: *You can use a separate array with* $O(n)$ *entries to help you. Start with the "naive" algorithm from (a) and try removing one of the loops with a smart lookup using the new array.*

Hint: $a + b + c = t$ *implies that* $a = t - b - c$.

(c)* Suppose now that, unlike in (b), we don't have a bound on the size of the integers elements of A nor on t (but we can still perform arithmetic operations on them in $O(1)$ time). However, they are given in increasing order in A, i.e., $A[1] \leq A[2] \leq \ldots A[n]$. Design an $O(n^2)$ algorithm to solve the same problem, assuming the constraints.

Hint: *Exploit the increasing order of* A *to leverage the computation done in the previous step to help you in the next one.*