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 0 — HS 20

The solutions for this sheet do not have to be submitted. The sheet will be solved in the first exercise session on 21.09.2020.

Exercises that are marked by * are challenge exercises.

Induction

The first two exercises are about the principle of mathematical induction.

Exercise 0.1 *Induction.*

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

$$1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}\,.$$

b) **(This subtask is from August 2019 exam).** Let $T : \mathbb{N} \to \mathbb{R}$ be a function that satisfies the following two conditions:

$$T(n) \geq 4 \cdot T(\tfrac{n}{2}) + 3n \qquad \text{whenever } n \text{ is divisible by 2;}$$
$$T(1) = 4.$$

Prove by mathematical induction that

$$T(n) \geq 6n^2 - 2n$$

holds whenever n is a power of 2, i.e., $n = 2^k$ with $k \in \mathbb{N}_0$.

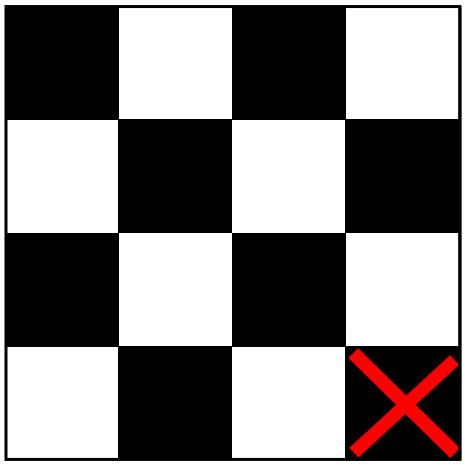
Exercise 0.2 *Divide and Conquer.*

Consider the following problem:

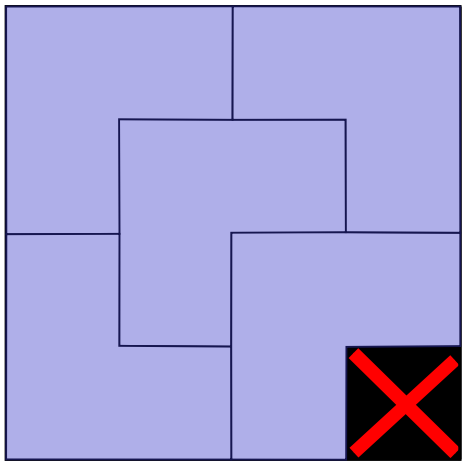
You are given a $2^k \times 2^k$ chessboard with one missing square and as many L-shaped puzzle pieces as you want. Each puzzle-piece can cover exactly three squares of the chessboard. As you will show algorithmically in this exercise, it is always possible to cover such chessboards by L-shaped puzzle pieces. An example is given in Figure 1 for $k = 2$, where the missing piece is a corner piece.

a) Devise a divide-and-conquer algorithm that can cover a $2^k \times 2^k$ chessboard with one missing square at an arbitrary position for $k \in \{1, 2, 3, \dots\}$. Describe your algorithm using words. Make sure to describe how you divide the problem into *subproblems* and how you handle the *base case(s)*. Your description should be *concise* (e.g., it could have a pseudo-code-like form for readability).

You can assume that each square is represented by its coordinates, specifically, the square in the lower left corner has coordinates $(1, 1)$ and the square in the upper right corner has coordinates $(2^k, 2^k)$. The input of your algorithm is (k, a, b), where a and b are coordinates of the missing square.



(a) 4×4 chessboard



(b) covering

Figure 1: Example of a chessboard and its covering by L-shaped puzzle pieces.

b)* Now suppose that we want to cut out the L-shaped puzzle pieces (given by the algorithm of part **a**) with scissors. We are interested in the number of *straight* cuts needed to cut out the pieces for a chessboard of size $n \times n$, where $n = 2^k$, and we call this number $T(n)$. For example, we need 2 straight cuts for a 2×2 chessboard, i.e $T(2) = 2$. Note that one straight cut can be as long as we want (we are not interested in the total length of the cuts). Our goal will be to show the upper bound $T(n) \leq \frac{4n^2-10}{3}$.

1) Find a recursion formula for $T(n)$, i.e. an upper bound for $T(n)$ in terms of $T(n/2)$.

2) Using part **1**, show by induction that $T(n) \leq \frac{4n^2-10}{3}$.

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 $\mathcal{O}$-Notation. We denote by $\mathbb{R}^+$ the set of all (strictly) positive real numbers and by $\mathbb{R}_0^+$ the set of nonnegative real numbers.

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

For this exercise sheet we can assume that $n_0 = 10$. In general, we say that $g \leq \mathcal{O}(f)$ if Definition 1 applies after restricting the domain to *some* $N = \{n_0, n_0 + 1, \ldots\}$. Some sources use the notation $g = \mathcal{O}(f)$ or $g \in \mathcal{O}(f)$ 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 $f \leq \mathcal{O}(g)$ and $g \not\leq \mathcal{O}(f)$.*
- *If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = C \in \mathbb{R}^+$, then $f \leq \mathcal{O}(g)$ and $g \leq \mathcal{O}(f)$.*

- *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$*, then* $f \not\leq \mathcal{O}(g)$ *and* $g \leq \mathcal{O}(f)$.

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.

Exercise 0.3 *Comparison of functions part 1.*

Show that

a) $2n \leq \mathcal{O}(3n)$ and $3n \leq \mathcal{O}(2n)$.

b) $n \leq \mathcal{O}(n \log n)$, but $n \log n \not\leq \mathcal{O}(n)$.

c) $10n^2 + 100n + 1000 \leq \mathcal{O}(n^3)$, but $n^3 \not\leq \mathcal{O}(10n^2 + 100n + 1000)$.

d) $2^n \leq \mathcal{O}(3^n)$, but $3^n \not\leq \mathcal{O}(2^n)$.

The following theorem can be useful to compute some limits.

Theorem 2 (L'Hôpital's rule)**.** *Assume that functions* $f : \mathbb{R}^+ \to \mathbb{R}^+$ *and* $g : \mathbb{R}^+ \to \mathbb{R}^+$ *are differentiable,* $\lim_{x\to\infty} f(x) = \lim_{x\to\infty} g(x) = \infty$ *and for all* $x \in \mathbb{R}^+$, $g'(x) \neq 0$. *If* $\lim_{x\to\infty} \frac{f'(x)}{g'(x)} = C \in \mathbb{R}_0^+$ *or* $\lim_{x\to\infty} \frac{f'(x)}{g'(x)} = \infty$*, then*

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = \lim_{x\to\infty} \frac{f'(x)}{g'(x)}\,.$$

Exercise 0.4 *Comparison of functions part 2.*

Show that

a) $n \ln n \leq \mathcal{O}(n^{1.01})$, but $n^{1.01} \not\leq \mathcal{O}(n \ln n)$.

b) $n \leq \mathcal{O}(e^n)$, but $e^n \not\leq \mathcal{O}(n)$.

c) $n^2 \leq \mathcal{O}(e^n)$, but $e^n \not\leq \mathcal{O}(n^2)$.

d)* $n^{100} \leq \mathcal{O}(1.01^n)$, but $1.01^n \not\leq \mathcal{O}(n^{100})$.

e)* $\log_2^{100} n \leq \mathcal{O}(2^{\sqrt{\log_2 n}})$, but $2^{\sqrt{\log_2 n}} \not\leq \mathcal{O}(\log_2^{100} n)$.

f)* $2^{\sqrt{\log_2 n}} \leq \mathcal{O}(n^{0.01})$, but $n^{0.01} \not\leq \mathcal{O}(2^{\sqrt{\log_2 n}})$.

For the next exercise you may use the following theorem.

Theorem 3. *Let* $f, g, h : \mathbb{N} \to \mathbb{R}^+$. *If* $f \leq \mathcal{O}(h)$ *and* $g \leq \mathcal{O}(h)$*, then*

1. *for any constant* $c \geq 0$, $cf \leq \mathcal{O}(h)$.
2. $f + g \leq \mathcal{O}(h)$.

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 $\log_a n \leq \mathcal{O}(\log_b n)$. So you don't have to write bases of logarithms in asymptotic notation, that is, you can just write $\mathcal{O}(\log n)$.

Exercise 0.5 *Simplifying expressions.*

Write the following in tight asymptotic notation, simplifying them as much as possible. It is guaranteed that all functions in this exercise take values in $\mathbb{R}^+$ (you don't have to prove it).

a) $5n^3 + 40n^2 + 100$

b) $5n + \ln n + 2n^3 + \frac{1}{n}$

c) $n \ln n - 2n + 3n^2$

d) $23n + 4n \log_5 n^6 + 78\sqrt{n} - 9$

e) $\log_2 \sqrt{n^5} + \sqrt{\log_2 n^5}$

f)* $2n^3 + \left(\sqrt[4]{n}\right)^{\log_5 \log_6 n} + \left(\sqrt[7]{n}\right)^{\log_8 \log_9 n}$

Exercise 0.6 *Some properties of $\mathcal{O}$-Notation.*

Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ and $g : \mathbb{R}^+ \to \mathbb{R}^+$.

a) Show that if $f \leq \mathcal{O}(g)$, then $f^2 \leq \mathcal{O}(g^2)$. You can assume that $\lim\limits_{x \to \infty} \frac{f(x)}{g(x)} = C \in \mathbb{R}_0^+$.

b) Give an example where $f \leq \mathcal{O}(g)$, but $2^f \not\leq \mathcal{O}(2^g)$.