Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Johannes Lengler, David Steurer
Lucas Slot, Manuel Wiedmer, Hongjie Chen, Ding Jingqiu

2 October 2023

Algorithms & Data Structures — Exercise sheet 2 — HS 23

The solutions for this sheet are submitted at the beginning of the exercise class on 9 October 2023.

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 $n \geq 5$,

$$2^n > n^2 .$$

Solution:

- **Base Case.**
  Let $n = 5$. Then:
  $$2^5 = 32 > 25 = 5^2 .$$
- **Induction Hypothesis.**
  Assume that the property holds for some positive integer $k \geq 5$, that is,
  $$2^k > k^2 .$$
- **Inductive Step.**
  We must show that the property holds for $k + 1$.
  $$\begin{aligned} 2^{k+1} &= 2 \cdot 2^k \\ &\overset{\text{I.H.}}{>} 2 \cdot k^2 \\ &= k^2 + k^2 \\ &\geq k^2 + 5k \\ &= k^2 + 2k + 3k \\ &\geq k^2 + 2k + 15 \\ &> k^2 + 2k + 1 \\ &= (k+1)^2 . \end{aligned}$$

  By the principle of mathematical induction, $2^n > n^2$ is true for every positive integer $n \geq 5$.

(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 $1 \leq i \leq n$,

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

Solution:

We will use the identity from the hint to show (via mathematical induction) that

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

- **Base Case.**
  Let $n = 1$. Then $(1+x)^1 = \binom{1}{0}x^0 + \binom{1}{1}x^1 = \sum_{i=0}^{n} \binom{n}{i} x^i$.
- **Induction Hypothesis.**
  Assume that the property holds for some positive integer k, that is,

  $$(1+x)^k = \sum_{i=0}^{k} \binom{k}{i} x^i.$$
- **Inductive Step.**
  We must show that the property holds for $k+1$.

  $$\begin{aligned}(1+x)^{k+1} &= (1+x)(1+x)^k \\ &\overset{I.H.}{=} (1+x)\sum_{i=0}^{k}\binom{k}{i}x^i \\ &= \Big(\sum_{i=0}^{k}\binom{k}{i}x^i\Big) + \Big(\sum_{i=0}^{k}\binom{k}{i}x^{i+1}\Big) \\ &= \Big(\sum_{i=0}^{k}\binom{k}{i}x^i\Big) + \Big(\sum_{i=1}^{k+1}\binom{k}{i-1}x^i\Big) \\ &= \binom{k}{0}x^0 + \sum_{i=1}^{k}\Big(\binom{k}{i}x^i + \binom{k}{i-1}x^i\Big) + \binom{k}{k}x^{k+1} \\ &= \binom{k+1}{0}x^0 + \sum_{i=1}^{k}\binom{k+1}{i}x^i + \binom{k+1}{k+1}x^{k+1} = \sum_{i=0}^{k+1}\binom{k+1}{i}x^i.\end{aligned}$$

  By the principle of mathematical induction, the property is true for every positive integer n.

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. Let N be a set of possible inputs.

Definition 1 (O-Notation). For $f : N \to \mathbb{R}^+$,

$$O(f) := \{g : N \to \mathbb{R}^+ \mid \exists C > 0\, \forall n \in N\; g(n) \le C \cdot f(n)\}.$$

We write $f \le O(g)$ to denote $f \in O(g)$. Some textbooks use here the notation $f = O(g)$. We believe the notation $f \le O(g)$ helps to avoid some common pitfalls in the context of asymptotic notation.

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 N be an infinite subset of $\mathbb{N}$ and $f : N \to \mathbb{R}^+$ and $g : N \to \mathbb{R}^+$.*

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

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

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

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

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

Exercise 2.2 *O-notation quiz.*

(a) For all the following functions the variable n ranges over $\mathbb{N}$. Prove or disprove the following statements. Justify your answer.

(1) $2n^5 + 10n^2 \le O(\frac{1}{100}n^6)$

Solution:

True by Theorem 1, since

$$\lim_{n\to\infty} \frac{2n^5 + 10n^2}{\frac{1}{100}n^6} = \lim_{n\to\infty} 200\frac{1}{n} + 1000\frac{1}{n^4} = 0.$$

(2) $n^{10} + 2n^2 + 7 \le O(100n^9)$

Solution:

False by Theorem 1, since

$$\lim_{n\to\infty} \frac{n^{10} + 2n^2 + 7}{100n^9} = \lim_{n\to\infty} \frac{1}{100}n + \frac{1}{50}\frac{1}{n^7} + \frac{7}{100}\frac{1}{n^9} = \infty.$$

(3) $e^{1.2n} \leq O(e^n)$

Solution:

False by Theorem 1, since

$$\lim_{n\to\infty} \frac{e^{1.2n}}{e^n} = \lim_{n\to\infty} e^{1.2n-n} = \lim_{n\to\infty} e^{0.2n} = \infty.$$

(4)* $n^{\frac{2n+3}{n+1}} \leq O(n^2)$

Solution:

True by Theorem 1, since

$$\lim_{n\to\infty} \frac{n^{\frac{2n+3}{n+1}}}{n^2} = \lim_{n\to\infty} n^{\frac{2n+3}{n+1}-2} = \lim_{n\to\infty} n^{\frac{2n+3-2n-2}{n+1}} = \lim_{n\to\infty} n^{\frac{1}{n+1}} = \lim_{n\to\infty} e^{\frac{\log n}{n+1}} = 1.$$

(b) Find f and g as in Theorem 1 such that $f \leq O(g)$, 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 sufficient, but not a necessary condition for $f \leq O(g)$.

Solution:

We define the following two functions $f, g : \mathbb{N} \to \mathbb{R}^+$. Let $f(n) = 2 + (-1)^n$ and $g(n) = 1$. We have $\frac{f(n)}{g(n)} = \frac{2+(-1)^n}{1} = 2 + (-1)^n$, which has no limit when $n \to \infty$. However, for any $n \in \mathbb{N}$, $f(n) \leq 3g(n)$ and thus $f \leq O(g)$.

Exercise 2.3 *Asymptotic growth of $\sum_{i=1}^{n} \frac{1}{i}$* (1 point).

The goal of this exercise is to show that the sum $\sum_{i=1}^{n} \frac{1}{i}$ behaves, up to constant factors, as $\log(n)$ when n is large. Formally, we will show $\sum_{i=1}^{n} \frac{1}{i} \leq O(\log n)$ and $\log n \leq O(\sum_{i=1}^{n} \frac{1}{i})$ as functions from $\mathbb{N}_{\geq 2}$ to $\mathbb{R}^+$.

For parts (a) to (c) we assume that $n = 2^k$ is a power of 2 for $k \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$. We will generalise the result to arbitrary $n \in \mathbb{N}$ in part (d). For $j \in \mathbb{N}$, define

$$S_j = \sum_{i=2^{j-1}+1}^{2^j} \frac{1}{i}.$$

(a) For any $j \in \mathbb{N}$, prove that $S_j \leq 1$.

Hint: *Find a common upper bound for all terms in the sum and count the number of terms.*

Solution:

For any i between $2^{j-1} + 1$ and 2^j, we have $\frac{1}{i} \leq \frac{1}{2^{j-1}}$. In the sum $S_j = \sum_{i=2^{j-1}+1}^{2^j} \frac{1}{i}$ there are $2^j - (2^{j-1} + 1) + 1 = 2^j - 2^{j-1} = 2^{j-1}$ terms. Thus, we get

$$S_j = \sum_{i=2^{j-1}+1}^{2^j} \frac{1}{i} \leq 2^{j-1} \cdot \frac{1}{2^{j-1}} = 1.$$

(b) For any $j \in \mathbb{N}$, prove that $S_j \geq \frac{1}{2}$.

Solution:

For any i between $2^{j-1}+1$ and 2^j, we have $\frac{1}{i} \geq \frac{1}{2^j}$. In the sum $S_j = \sum_{i=2^{j-1}+1}^{2^j} \frac{1}{i}$ there are again 2^{j-1} terms. Thus, we get

$$S_j = \sum_{i=2^{j-1}+1}^{2^j} \frac{1}{i} \geq 2^{j-1} \cdot \frac{1}{2^j} = \frac{1}{2}.$$

(c) For any $k \in \mathbb{N}_0$, prove the following two inequalities

$$\sum_{i=1}^{2^k} \frac{1}{i} \leq k+1$$

and

$$\sum_{i=1}^{2^k} \frac{1}{i} \geq \frac{k+1}{2}.$$

Hint: *You can use that $\sum_{i=1}^{2^k} \frac{1}{i} = 1 + \sum_{j=1}^{k} S_j$. Use this, together with parts (a) and (b), to prove the required inequalities.*

Solution:

By parts (a) and (b), we have that $\frac{1}{2} \leq S_j \leq 1$ for any $j \in \mathbb{N}$. Thus, we get, using the hint,

$$\sum_{i=1}^{2^k} \frac{1}{i} = 1 + \sum_{j=1}^{k} S_j \leq 1 + \sum_{j=1}^{k} 1 = k+1.$$

Similarly, we have

$$\sum_{i=1}^{2^k} \frac{1}{i} = 1 + \sum_{j=1}^{k} S_j \geq 1 + \sum_{j=1}^{k} \frac{1}{2} \geq \frac{1}{2} + \frac{k}{2} = \frac{k+1}{2}.$$

(d)* For arbitrary $n \in \mathbb{N}$, prove that

$$\sum_{i=1}^{n} \frac{1}{i} \leq \log_2(n) + 2$$

and

$$\sum_{i=1}^{n} \frac{1}{i} \geq \frac{\log_2 n}{2}.$$

Hint: *Use the result from part (c) for $k_1 = \lceil \log_2 n \rceil$ and $k_2 = \lfloor \log_2 n \rfloor$. Here, for any $x \in \mathbb{R}$, $\lceil x \rceil$ is the smallest integer that is at least x and $\lfloor x \rfloor$ is the largest integer that is at most x. For example, $\lceil 1.5 \rceil = 2$, $\lfloor 1.5 \rfloor = 1$ and $\lceil 3 \rceil = \lfloor 3 \rfloor = 3$. In particular, for any $x \in \mathbb{R}$, $x \leq \lceil x \rceil < x+1$ and $x \geq \lfloor x \rfloor > x-1$.*

Solution:

We want to apply part (c) to $k_1 = \lceil \log_2 n \rceil$. Note that $n \leq 2^{k_1}$. Now, using that for any $i \in \mathbb{N}$, $\frac{1}{i} \geq 0$, we get

$$\sum_{i=1}^{n} \frac{1}{i} \leq \sum_{i=1}^{2^{k_1}} \frac{1}{i} \leq k_1 + 1 = \lceil \log_2 n \rceil + 1 \leq \log_2(n) + 2.$$

Applying part (c) to $k_2 = \lfloor \log_2 n \rfloor$ (note that $n \geq 2^{k_2}$) and again using that $\frac{1}{i} \geq 0$ for any $i \in \mathbb{N}$, we get

$$\sum_{i=1}^{n} \frac{1}{i} \geq \sum_{i=1}^{2^{k_2}} \frac{1}{i} \geq \frac{k_2 + 1}{2} = \frac{\lfloor \log_2 n \rfloor + 1}{2} \geq \frac{(\log_2(n) - 1) + 1}{2} = \frac{\log_2(n)}{2}.$$

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

Recall that the factorial of a positive integer n is defined as $n! = 1 \times 2 \times \cdots \times (n-1) \times n$. For the following functions n ranges over $\mathbb{N}_{\geq 2}$.

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

Hint: You can use the fact that $n! \leq n^n$ *for* $n \in \mathbb{N}_{\geq 2}$ *without proof.*

Solution:

From the hint, we have $n! \leq n^n$, which implies using the monotonicity of the logarithm that $\ln(n!) \leq n \ln n$ and thus $\ln(n!) \leq O(n \ln n)$.

(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* $n \in \mathbb{N}_{\geq 2}$ *without proof.*

Solution:

From the hint, we have $n! \geq \left(\frac{n}{2}\right)^{n/2}$. Now, by the monotonicity of the logarithm we have

$$\ln(n!) \geq \ln\left(\left(\frac{n}{2}\right)^{n/2}\right) = \frac{n}{2}\left(\ln n - \ln 2\right),$$

so $n \ln n \leq 2\ln(n!) + n \ln 2$. Now, note that $n \ln 2 \leq \ln 2 + \sum_{i=2}^{n} \ln(i) = \ln 2 + \ln(n!) \leq 2\ln(n!)$ since $n \geq 2$. Thus, $n \ln n \leq 2\ln(n!) + n\ln 2 \leq 4\ln(n!)$, which shows $n \ln n \leq O(\ln(n!))$.

Exercise 2.5 *Testing equations* **(2 points)**.

Your friend sends you a piece of code that computes his favorite function $f : \mathbb{N} \to \mathbb{N}$. For $n \in \mathbb{N}$, we want to test if the equation $f(a) + f(b) + f(c) = f(d)$ can be satisfied using positive integers $1 \leq a, b, c, d \leq n$. Your friend completed Algorithms and Data Structures last year, and so you may assume that his code computes $f(k)$ in $O(1)$ for any $k \in \mathbb{N}$. You may also assume simple arithmetic operations on integers can be performed in $O(1)$. Finally, you may initialize an array of size k in time $O(k)$.

(a) Design a simple $O(n^4)$ algorithm that outputs "YES" if there exist integers $1 \leq a, b, c, d \leq n$ such that $f(a) + f(b) + f(c) = f(d)$ and "NO" otherwise.

Solution:

The algorithm can simply check all n^4 tuples (a, b, c, d) of positive integers by using four nested loops with indices (a, b, c, d) that iterate over all integers in $[1, n]$. For each such tuple, we check whether $f(a) + f(b) + f(c) = f(d)$ in time $O(1)$ and report success ("YES") if we ever find a satisfying tuple. Otherwise, we return failure ("NO").
The algorithm checks all n^4 possibilities, hence it is trivially correct and its runtime is clearly $O(n^4)$.

(b) Assume that $f(k) \leq k^3$ for all $k \in \mathbb{N}$. Modify your previous algorithm so that it works in time $O(n^3)$ under this assumption. Motivate briefly why it still works.

Hint: *You could use a helper array of size n^3 to get rid of one of the loops in your previous algorithm. The helper array could save which values the function f can take.*

Solution:

We create a helper array $H[1, \dots, n^3]$ of size n^3, whose entries are initially all zero. Using a single loop, we set $H[f(d)] = 1$ for all $1 \leq d \leq n$. Then we loop over the tuples (a, b, c) with $1 \leq a, b, c \leq n$. For each such tuple, we compute $\alpha = f(a) + f(b) + f(c)$ in time $O(1)$ and report success ("YES") if we ever find a tuple with $H[\alpha] = 1$. Otherwise, we return failure ("NO").
The runtime of the algorithm is $O(n^3)$: we need $O(n^3)$ to initialize the array, $O(n)$ to set the correct entries to 1, and $O(n^3)$ to loop over the integers $1 \leq a, b, c \leq n$. For correctness, note that $1 \leq f(d) \leq n^3$ for all $1 \leq d \leq n$, and so we do not run into 'out of bounds' errors.

(c)* Assume that $f(k) \leq k^2$ for all $k \in \mathbb{N}$. Modify your previous algorithm so that it works in time $O(n^2)$ under this assumption. Motivate briefly why it still works.

Hint: *You could use a helper array again. Note that $f(a) + f(b) + f(c) = f(d)$ implies that $f(a) + f(b) = f(d) - f(c)$.*

Solution:

The approach is similar to before, but now we create an array $H[1, \dots, 2n^2]$ of size $2n^2$ whose entries are initially all zero. Then, using a double loop we set $H[f(a) + f(b)] = 1$ for all tuples (a, b) with $1 \leq a, b \leq n$. Then we loop over the tuples (c, d) with $1 \leq c, d \leq n$. For each such tuple, we compute $\alpha = f(d) - f(c)$ in time $O(1)$ and report success ("YES") if we ever find a tuple with $\alpha \geq 1$ and $H[\alpha] = 1$. Otherwise, we return failure ("NO").
The runtime of the algorithm is $O(n^2)$: We need $O(n^2)$ to initialize the array, $O(n^2)$ to set the correct entries to 1, and $O(n^2)$ to loop over the integers $1 \leq c, d \leq n$. For correctness, note that $1 \leq f(a) + f(b) \leq 2n^2$ for all $1 \leq a, b \leq n$, and so we do not run into 'out of bounds' errors.