Datastructures and Algorithms

Introduction, Logistics, Asymptotics ($\mathcal{O}$, Ω, Θ)

Adel Gavranović — ETH Zürich — 2025

Overview

Learning Objectives
Exercise Management
Repetition Theory
Examples and Quiz on Theory
Quiz on Asymptotic Running Time of Program Fragments
Formulas and their Derivation*
Past Exam Questions
Tips for **code** expert

n.ethz.ch/~agavranovic

Material

Webpage

Mail

1. Intro

Hello, World!

Welcome!

2. Follow-up

Follow-up from last session

- This is where I explain things I missed (or messed up) in last week's session

3. Feedback regarding **code** expert

General things regarding **code** expert

- This is where I mention very common mistakes that were made in the exercises on **code** expert
- Emails are welcome too

Any questions regarding **code** expert on your part?

- This is where you’ll have a chance to ask thigs regarding **code** expert that you think might be relevant for the class (hint: it almost always is)

4. Learning Objectives

Objectives

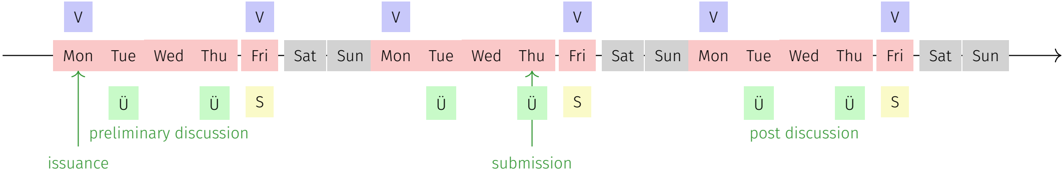
- ☐ Know how the course is built up
- ☐ Understand the definitions of $\mathcal{O}$, Ω, and Θ
- ☐ Understand the uses of $\mathcal{O}$, Ω, and Θ

5. Summary

Getting on the same page

- This is where we could talk about what happened during the week if you want to

Exercises



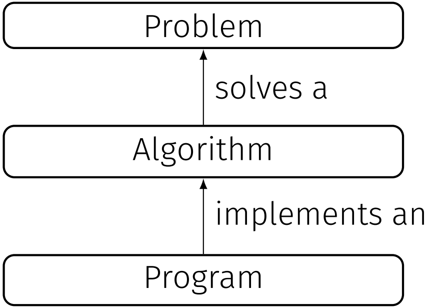
- Exercises available at lecture time.
- Preliminary discussion in the following recitation session
- Submit the exercise at the lecture two weeks later. Exception: for the first exercise you only have one week to finish.
- Dicussion of the exercise in the recitation session after the deadline. Feedback within a week after discussion.

7. Repetition Theory

Warm-up

- What is a problem?
- What is an algorithm?
    - ➜ well-defined computing procedure to compute output data from input data.
- What is a program?
    - ➜ Concrete implementation of an algorithm

Problems, Algorithms and Programs



Warm-up

$$\begin{tikzcd} \text{Problem} \arrow[d, "\text{can be solved by multiple}"] \\ \text{Algorithms} \arrow[d, "\text{can be implemented in various ways}"] \\ \text{Program} \end{tikzcd}$$

Efficiency

Program	Computing time	Measurable value on an actual machine.
Algorithm	Cost	Number of elementary operations
Problem	Complexity	Minimal (asymptotic) cost over all algorithms that solve the problem.

➜ Estimation of cost or computing time depending on the input size, denoted by n.

Asymptotic behavior

- What are $\Omega(g(n))$, $\Theta(g(n))$, $\mathcal{O}(g(n))$?

➜ Sets of functions!

subset	$A \subseteq B$
proper subset	$A \subsetneq B$
intersection	$A \cap B$

Asymptotic behavior

Given: function $f : \mathbb{N} \to \mathbb{R}$.

Definition:

$$\mathcal{O}(g) = \{f : \mathbb{N} \to \mathbb{R} | \exists c > 0, n_0 \in \mathbb{N} | \forall n \geq n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$$
$$\Omega(g) = \{f : \mathbb{N} \to \mathbb{R} | \exists c > 0, n_0 \in \mathbb{N} | \forall n \geq n_0 : 0 \leq c \cdot g(n) \leq f(n)\}$$
$$\Theta(g) = \mathcal{O}(g) \cap \Omega(g)$$

Intuition:

$f \in \mathcal{O}(g)$: f grows asymptotically **not faster** than g. Algorithm with running time f is **not worse** than any other algorithm with g.

$f \in \Omega(g)$: f grows asymptotically **not slower** than g. Algorithm with running time f is **not better** than any other algorithm with g.

$f \in \Theta(g)$: f grows asymptotically **as fast** as g. Algorithm with running time f is **as good as** any other algorithm with g.

Used less often

Given: function $f : \mathbb{N} \to \mathbb{R}$.
Definition:

$$\mathcal{O}(g) = \{f : \mathbb{N} \to \mathbb{R} | \exists c > 0, n_0 \in \mathbb{N} | \forall n \geq n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$$
$$o(g) = \{f : \mathbb{N} \to \mathbb{R} | \forall c > 0 \; \exists n_0 \in \mathbb{N} | \forall n \geq n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$$

$$\Omega(g) = \{f : \mathbb{N} \to \mathbb{R} | \exists c > 0, n_0 \in \mathbb{N} | \forall n \geq n_0 : 0 \leq c \cdot g(n) \leq f(n)\}$$
$$\omega(g) = \{f : \mathbb{N} \to \mathbb{R} | \forall c > 0 \; \exists n_0 \in \mathbb{N} | \forall n \geq n_0 : 0 \leq c \cdot g(n) \leq f(n)\}$$

$f \in o(g)$: f grows much slower than g
$f \in \omega(g)$: f grows much faster than g

Useful information for the exercise

Theorem 1

1. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in \mathcal{O}(g)$, $\mathcal{O}(f) \subsetneq \mathcal{O}(g)$.
2. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = C > 0$ (C constant) $\Rightarrow f \in \Theta(g)$.
3. $\frac{f(n)}{g(n)} \underset{n\to\infty}{\longrightarrow} \infty \Rightarrow g \in \mathcal{O}(f)$, $\mathcal{O}(g) \subsetneq \mathcal{O}(f)$.

Example 2

1. $\lim_{n\to\infty} \frac{n}{n^2} = 0 \Rightarrow n \in \mathcal{O}(n^2)$, $\mathcal{O}(n) \subsetneq \mathcal{O}(n^2)$.
2. $\lim_{n\to\infty} \frac{2n}{n} = 2 > 0 \Rightarrow 2n \in \Theta(n)$.
3. $\frac{n^2}{n} \underset{n\to\infty}{\longrightarrow} \infty \Rightarrow n \in \mathcal{O}(n^2)$, $\mathcal{O}(n) \subsetneq \mathcal{O}(n^2)$.

Property

$$f_1 \in \mathcal{O}(g), f_2 \in \mathcal{O}(g) \Rightarrow f_1 + f_2 \in \mathcal{O}(g)$$

8. Examples and Quiz on Theory

Examples

$$\mathcal{O}(g) = \{f : \mathbb{N} \to \mathbb{R} |\ \exists c > 0, \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$$

$f(n)$	$f \in \mathcal{O}(?)$	Example
$3n + 4$		
$2n$		
$n^2 + 100n$		
$n + \sqrt{n}$		

Examples

- $n \in \mathcal{O}(n^2)$?
- $3n^2 \in \mathcal{O}(2n^2)$?
- $2n^2 \in \mathcal{O}(n)$?
- $\mathcal{O}(n) \subseteq \mathcal{O}(n^2)$?
- $\Theta(n) \subseteq \Theta(n^2)$?

Quiz

$1 \in \mathcal{O}(15)$?

$2n + 1 \in \Theta(n)$?

$\sqrt{n} \in \mathcal{O}(n)$?

$\sqrt{n} \in \Omega(n)$?

$n \in \Omega(\sqrt{n})$?

$\sqrt{n} \notin \Theta(n)$?

$\mathcal{O}(\sqrt{n}) \subset \mathcal{O}(n)$?

$2^n \notin \mathcal{O}(\exp(n))$?

A good strategy?

... Then I simply buy a new machine! If today I can solve a problem of size n, then with a 10 or 100 times faster machine I can solve ...

Komplexität	(speed $\times 10$)	(speed $\times 100$)
$\log_2 n$		
n		
n^2		
2^n		

9. Quiz on Asymptotic Running Time of Program Fragments

Asymptotic Running Times with Θ

```
void run(int n){
  for (int i = 1; i<n; ++i)
  op();
}
```

How often is `op()` called as a function of n?

Asymptotic Running Times with Θ

```
void run(int n){
  for (int i = 1; i<n; ++i)
    for (int j = 1; j<n; ++j)
      op();
}
```

How often is `op()` called as a function of n?

Asymptotic Running Times with Θ

```
void run(int n){
  for (int i = 1; i<n; ++i)
    for (int j = i; j<n; ++j)
      op();
}
```

How often is `op()` called as a function of n?

Asymptotic Running Times with Θ

```
void run(int n){
  for (int i = 1; i<n; ++i){
    op();
    for (int j = i; j<n; ++j)
      op();
    }
}
```

How often is `op()` called?

Asymptotic Running Times with Θ

```
void run(int n){
  for (int i = 1; i<n; ++i){
    op();
    for (int j = 1; j<i*i; ++j)
      op();
    }
}
```

How often is `op()` called?

Asymptotic Running Times with Θ

```
void run(int n){
  for(int i = 1; i <= n; ++i)
    for(int j = 1; j*j <= n; ++j)
      for(int k = n; k >= 2; --k)
        op();
}
```

How often is `op()` called as a function of n?

Asymptotic Running Times with Θ

```
int f(int n){
    i=1;
    while (i <= n*n*n){
        i = i*2;
        op();
    }
    return i;
}
```

10. Formulas and their Derivation*

Sums

$$\sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}$$

Why?

Intuition

$$1 + ... + 100 = (1 + 100) + (2 + 99) + (3 + 98) + ... + (50 + 51)$$

More formally?

Sums

$$\sum_{i=0}^{n}(n-i) = \sum_{i=0}^{n} i$$

$$\begin{aligned}
\Rightarrow 2 \cdot \sum_{i=0}^{n} i &= \sum_{i=0}^{n} i + \sum_{i=0}^{n}(n-i) \\
&= \sum_{i=0}^{n}(i + (n-i)) = \sum_{i=0}^{n} n = (n+1) \cdot n
\end{aligned}$$

Sums

$$\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

This you do not need to know by heart. But you should know that it is a polynomial of third degree.

Sums

How do you derive something like this? Interesting Trick: On the one hand

$$\sum_{i=0}^{n} i^3 - \sum_{i=1}^{n} (i-1)^3 = \sum_{i=0}^{n} i^3 - \sum_{i=0}^{n-1} i^3 = n^3,$$

on the other hand

$$\begin{aligned} \sum_{i=0}^{n} i^3 - \sum_{i=1}^{n} (i-1)^3 &= \sum_{i=1}^{n} i^3 - \sum_{i=1}^{n} (i-1)^3 \\ &= \sum_{i=1}^{n} i^3 - (i-1)^3 = \sum_{i=1}^{n} 3 \cdot i^2 - 3 \cdot i + 1 \end{aligned}$$

Exponents and Logarithms

$$\log_a y = x \Leftrightarrow a^x = y \quad (a > 0, y > 0)$$

$$a^x \cdot a^y = a^{x+y}$$

$$\frac{a^x}{a^y} = a^{x-y}$$

$$a^{x \cdot y} = (a^x)^y$$

$$\log_b x = \log_b a \cdot \log_a x$$

$$\log_a (x \cdot y) = \log_a x + \log_a y$$

$$\log_a \frac{x}{y} = \log_a x - \log_a y$$

$$\log_a x^y = y \log_a x$$

$$\log_a n! = \sum_{i=1}^{n} \log i$$

$$a^{\log_b x} = x^{\log_b a}$$

To see the last line, replace $x \to a^{\log_a x}$

Comparisons

$$\frac{n^2}{2^n} \xrightarrow[n \to \infty]{} 0$$

Comparisons

$$\frac{n^{10000}}{2^n} \xrightarrow[n\to\infty]{} 0$$

Comparisons

$d > 1, c > 0$

$$\frac{n^c}{d^n} \xrightarrow[n \to \infty]{} 0$$

because

$$\frac{n^c}{d^n} = \frac{2^{\log_2 n^c}}{2^{\log_2 d^n}} = 2^{c \cdot \log_2 n - n \log_2 d}$$

Comparisons

$$\frac{n}{\log n} \xrightarrow[n\to\infty]{} \infty$$

Comparisons

$$\frac{n \log n}{\sqrt{n}} \xrightarrow[n \to \infty]{} \infty$$

Comparisons

$$\frac{\log_2 n^2}{\sqrt{n}} \xrightarrow[n\to\infty]{} 0$$

$$\begin{aligned}\log_2 n^2 &= 2\log_2 n\\ \sqrt{n} &= n^{1/2} = 2^{\log_2 n^{1/2}} = \left(\sqrt{2}\right)^{\log_2 n}\\ \frac{\log n^2}{\sqrt{n}} &= 2\frac{\log_2 n}{\left(\sqrt{2}\right)^{\log_2 n}}\end{aligned}$$

which behaves because of $\log_2 n \to \infty$ for $n \to \infty$ like $2\frac{n}{\left(\sqrt{2}\right)^n}$

11. Past Exam Questions

Past Exam Questions

- If time allows, this is where we could have a look at old exam questions and go over them together

Past Exams

12. Tips for **code** expert

Tips for upcoming **code** expert exercises

Task "Taskname"

- This is where I give you hints and tips for the upcoming **code** expert exercises

13. Outro

General Questions?

- This is where you can ask general questions regarding the course or bring up things I didn’t cover during the session

See you next time!

Have a nice week!