UNIVERSITÄT BIELEFELD
Technische Fakultät

M.Sc. Andreas Rempel
UHG U10-137 / +49 521 106-4914
andreas.rempel@uni-bielefeld.de

392041 Implementation of Algorithms (Ü) (WiSe 2021/2022)

Date: Wed 08-10 Room: GZI & ONLINE Period: 11.10.2021-04.02.2022

Motivation

Algorithms are the tools of every computer scientist when it comes to solving many theoretical as well as real-world problems with the help of a computer. Already in the first semesters of the computer science study program, students learn many useful data structures and algorithms that they can apply to typical problems in computer science. This includes classic data structures such as lists, trees, and graphs, as well as algorithms for finding shortest paths, comparing texts, compression, among others. While most courses focus on the theoretical understanding of these algorithms, the focus in this course is on the actual implementation of the algorithms in a common programming language as required in project work and professional practice.

Course description

In this course, students have the opportunity to implement an algorithm of their choice in a programming language they are familiar with. This can be an algorithm that they have already learned about in the course of their studies but have not yet implemented themselves, or a completely new algorithm from current research. The topics are based on the students' wishes and are distributed individually in the first session. In the following sessions, one of the students will first briefly present the algorithm and then their implementation and receive feedback from the other participants. The main focus should be on the efficiency of the implementation, i.e., whether the theoretically achievable smallest upper bounds for runtime / memory are met by the implementation. Students should get a feel for efficient programming and become aware of typical sources of problems such as nested loops and recursions in their own code.

Learning objectives

Upon successful completion of this course, students will be able to...

- *name* and *describe* some basic algorithms of computer science.
- *explain* how these algorithms work using simple hands-on examples.
- *implement* an algorithm efficiently in a chosen programming language.
- *decide on* appropriate data structures and strategies for implementation.
- *examine* runtime and memory behavior in their own and other's code.
- *present* their own code as well as *provide* constructive *feedback* to others.

Module assignments

- Module 39-Inf-AB Algorithms in Bioinformatics
- Module 39-Inf-SAB Special Algorithms in Bioinformatics
  (or *individual supplementary area*: 1 LP)

Recommended previous modules

- Module 39-Inf-1 Algorithms and Data Structures
- Module 39-Inf-2 Object-oriented programming

Course credit and examination

- *Programming with presentation (course credit)*

  Students select an algorithm which they implement in a programming language of their choice in the course of the semester. The assignment is mostly processed independently and should have a scope of about 30 hours.

  The subsequent presentation should be structured as follows:

  – *A short presentation of the algorithm (3-5 slides, approx. 20 min)*

    1. Who invented it? What problem was it intended to solve?
    2. How does the algorithm work? Description and/or pseudocode.
    3. What is the theoretical runtime and memory complexity? Why?
    4. A small example to try the algorithm "by hand".

  – *A presentation of the implementation (source code, approx. 10 min)*

  The evaluation will focus primarily on the efficiency of the implementation. The students should send their code at least one week before the presentation in order to receive feedback and an opportunity for improvement.

- *Active participation* in the course is desirable.
  Students should give each other feedback.

Topic selection

- Boyer-Moore algorithm
- Knuth-Morris-Pratt algorithm
- Waterman-Eggert algorithm
- Gotoh algorithm
- Hirschberg algorithm
- Sellers algorithm
- Center-Star approximation
- Manber-Myers algorithm
- WOTD algorithm
- Neighbour Joining
- Agglomerative Clustering
- Fitch/Sankoff algorithm
- Prim/Kruskal algorithm
- Bellman-Ford algorithm
- Burrows-Wheeler transform
- Nussinov algorithm
- Double-Cut-and-Join distance
- De Bruijn graphs

Literature

- T. Cormen, C. Leiserson: Algorithms · An Introduction
- M. Nebel, S. Wild: Entwurf und Analyse von Algorithmen
- D. Gusfield: Algorithms on Strings, Trees and Sequences