DSK814: Algorithms and Data Structures

SDU - Kolding

Introduction to Algorithms and Data Structures

DSK814: Algorithms and Data Structures

- 10 ECTS

- Class hours 44 hours over 15 weeks
  - (1 □ 3) hours lecture per week

- Training hours 44 hours over 15 weeks
  - (1 □ 2) hours lecture for 6 weeks
  - (2 □ 2) hours lectures for 8 weeks

- Academic Prerequisites:
  - Mathematical Maturity
  - Proficiency in Programming

DSK814: Algorithms and Data Structures

- The Exam

Exam element a)

Project assignment

2.5 ECTS points

Exam element b)

Written exam

7.5 ECTS points

Course Plan

- Introduction
- Asymptotic analysis
- Sorting Algorithms
- Linked Lists
- Stacks
- Queues
- Hash Tables
- Trees
- Graphs

Resources for Algorithms and Data Structures

- Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms, **4th edition**, 2022.

  (and the third edition of the book)

  https://dl.ebooksworld.ir/books/Introduction.to.Algorithms.4th.Leiserson.Stein.Rivest.Cormen.MIT.Press.9780262046305.EBooksWorld.ir.pdf

- https://www.w3schools.com/dsa/index.php

Introduction to Algorithms and Data Structures

Algorithms and Data Structures

Algorithms is about how to solve different problems, often by searching through and manipulating data structures.

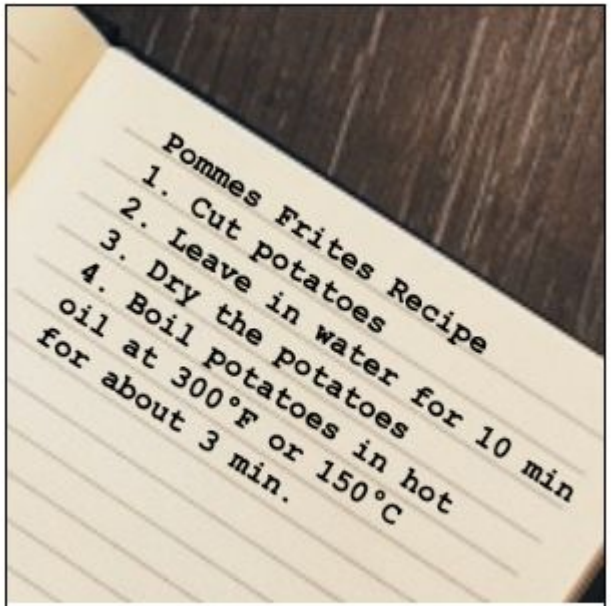
Data Structures is about how data can be stored in different structures.

Theory about Data Structures and Algorithms (DSA) helps us to use large amounts of data to solve problems efficiently.

What are Algorithms?

An algorithm is a set of step-by-step instructions to solve a given problem or achieve a specific goal.A cooking recipe written on a piece of paper is an example of an algorithm, where the goal is to make a certain dinner. The steps needed to make a specific dinner are described exactly.

When we talk about algorithms in **Computer Science,** the step-by-step instructions are written in a programming language, and instead of food ingredients, an **algorithm uses data structures**.



Pommes Frites Recipe

Algorithms

- Algorithms are fundamental to computer programming as they provide step-by-step instructions for executing tasks.

- An efficient algorithm can help us to find the **solution** we are looking for, and to transform a slow program into a faster one.

- By studying algorithms, developers can write better programs.

Algorithm Examples

- Finding the fastest route in a GPS navigation system
- Navigating an airplane or a car (cruise control)
- Finding what users search for (search engine)
- Sorting, for example sorting movies by rating

The algorithms we will look at in this course are designed to solve specific problems, and are often made to work on specific data structures.

For example, the 'Bubble Sort' algorithm is designed to sort values, and is made to work on arrays.

What are Data Structures?

- A data structure is a **way to store data.**
- We structure data in different ways depending on what data we have, and what we want to do with it.
- Data structures give us the possibility to manage large amounts of data efficiently for uses such as large databases and internet indexing services.
- Data structures are essential ingredients in creating fast and powerful algorithms. They help in managing and organizing data, reduce complexity, and increase efficiency.

Data Structures

In Computer Science there are two different kinds of data structures.

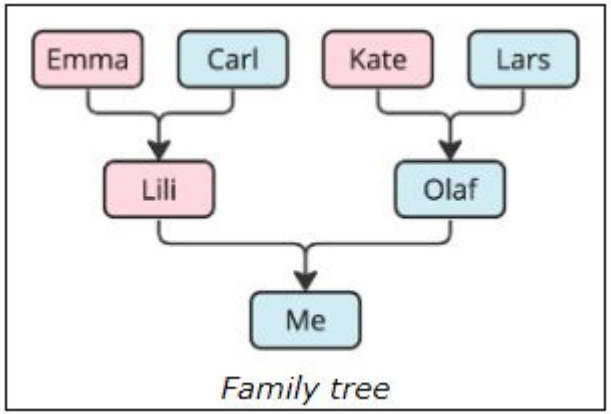
Primitive Data Structures are basic data structures provided by programming languages to represent single values, such as integers, floating-point numbers, characters, and booleans.

Abstract Data Structures are higher-level data structures that are built using primitive data types and provide more complex and specialized operations. Some common examples of abstract data structures include arrays, linked lists, stacks, queues, trees, and graphs.

Data Structures Example

If we want to store data about people we are related to, we use a family tree as the data structure.

We choose a family tree as the data structure because we have information about people we are related to and how they are related, and we want an overview so that we can easily find a specific family member, several generations back.



Family tree

Data Structures together with Algorithms

- Data structures and algorithms (DSA) go hand in hand.

- A data structure is not worth much if you cannot search through it or manipulate it efficiently using algorithms, and the algorithms are not worth much without a data structure to work on.

- DSA is about finding efficient ways to store and retrieve data, to perform operations on data, and to solve specific problems.

Data Structures together with Algorithms

By understanding DSA, you can:

- ➢ Decide which data structure or algorithm is best for a given situation.
- ➢ Make programs that run faster or use less memory.
- ➢ Understand how to approach complex problems and solve them in a systematic way.

Where is Data Structures and Algorithms Needed?

- For managing large amounts of data, such as in a social network or a search engine.
- For scheduling tasks, to decide which task a computer should do first.
- For planning routes, like in a GPS system to find the shortest path from A to B.
- For optimizing processes, such as arranging tasks so they can be completed as quickly as possible.
- For solving complex problems: From finding the best way to pack a truck to making a computer 'learn' from data.

Data Structures

DSA is fundamental in nearly every part of the software world:

- Operating Systems
- Database Systems
- Web Applications
- Search Engines
- Machine Learning
- Video Games
- Cryptographic Systems
- Data Analysis

Theory and Terminology

As we go along in this tutorial, new theoretical concepts and terminology (new words) will be needed so that we can better understand the data structures and algorithms we will be working on.

These new words and concepts will be introduced and explained properly when they are needed, but here is a list of some key terms, just to get an overview of what is coming:

Theory and Terminology

Term	Description
Algorithm	A set of step-by-step instructions to solve a specific problem.
Data Structure	A way of organizing data so it can be used efficiently. Common data structures include arrays, linked lists, and binary trees.
Time Complexity	A measure of the amount of time an algorithm takes to run, depending on the amount of data the algorithm is working on.
Space Complexity	A measure of the amount of memory an algorithm uses, depending on the amount of data the algorithm is working on.

Theory and Terminology

Term	Description
Big O Notation	A mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity. Used in this tutorial to describe the time complexity of an algorithm.
Recursion	A programming technique where a function calls itself.
Divide and Conquer	A method of solving complex problems by breaking them into smaller, more manageable sub-problems, solving the sub-problems, and combining the solutions. Recursion is often used when using this method in an algorithm.
Brute Force	A simple and straight forward way an algorithm can work by simply trying all possible solutions and then choosing the best one.

In-class Discussion

- Give a real-world example that requires sorting.

In-class Discussion

- Give a real-world example that requires sorting.

- Other than speed, what other measures of efficiency might one use in a real-world setting?

In-class Discussion

- Give a real-world example that requires sorting.

- Other than speed, what other measures of efficiency might one use in a real-world setting?

- Select a data structure that you have seen previously, and discuss its strengths and limitations.

In-class Discussion

- Give a real-world example that requires sorting.

- Other than speed, what other measures of efficiency might one use in a real-world setting?

- Select a data structure that you have seen previously, and discuss its strengths and limitations.

- Come up with a real-world problem in which only the best solution will do. Then come up with one in which a solution that is “approximately” the best is good enough.