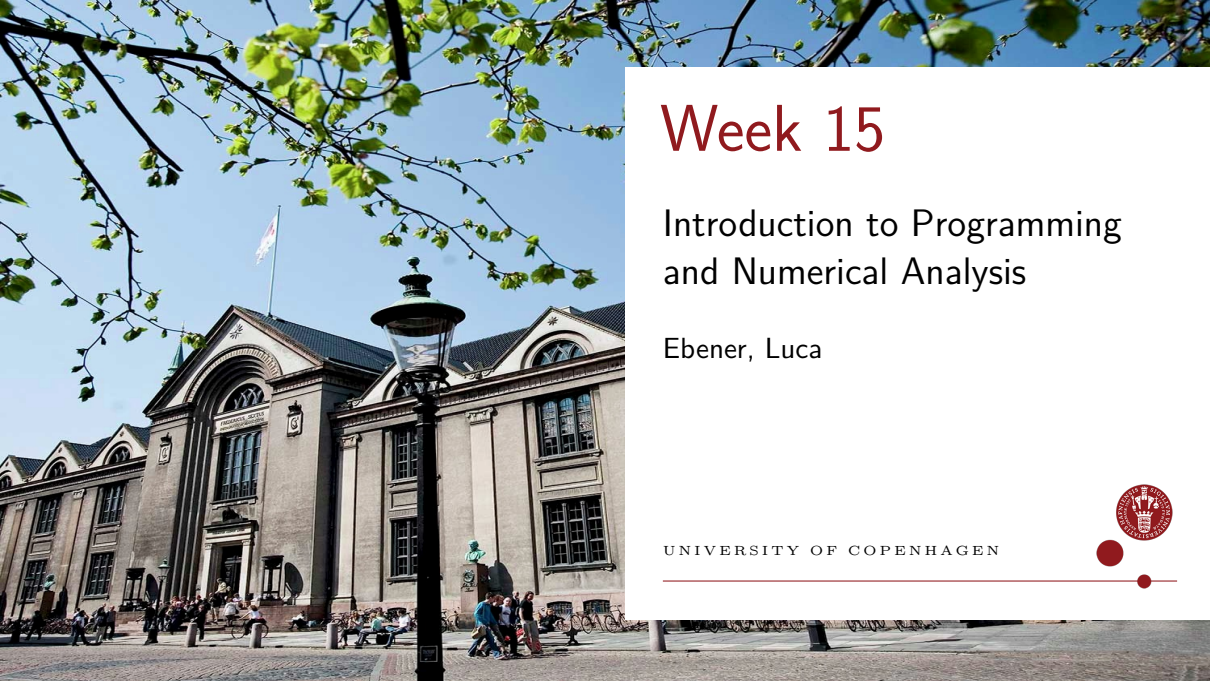

Week 15

Introduction to Programming and Numerical Analysis

Ebener, Luca

UNIVERSITY OF COPENHAGEN

Overview

- Algorithms
- Work on problem set 5

Basics of Algorithms

An algorithm is a step-by-step procedure used to solve a problem.

- An algorithm must be **unambiguous**, meaning each step is clearly defined and leads to only one meaning.
- It should be **executable**, which means it can be implemented and run on a computer.
- An algorithm should **terminate** in a finite amount of time, meaning it must eventually stop and produce a result.
- It should have well-defined inputs and outputs.

Complexity

Complexity analysis is crucial for understanding the performance of algorithms.

- **Complexity analysis** evaluates how the runtime of an algorithm grows with input size. It helps in predicting how an algorithm will perform as the size of the input increases.
- **Time complexity** measures the number of operations an algorithm performs as a function of the input size. It quantifies the amount of time an algorithm takes to run.
- **Space complexity** measures the amount of memory an algorithm uses as a function of the input size. It quantifies the amount of memory required by an algorithm to execute.

Big O Notation

Big O notation is a fundamental concept in algorithm analysis.

- **Big O notation** describes the average, upper or lower bound of an algorithm's time or space complexity. It provides a way to quantify how the runtime or space requirements of an algorithm grow as the size of the input increases.
- $O(f(n))$ represents the set of functions that grow no faster than $f(n)$ as n approaches infinity.
- It's commonly used to **classify algorithms based on their efficiency**. Algorithms with lower order complexities (e.g., $O(1)$, $O(\log n)$) are considered more efficient than those with higher order complexities (e.g., $O(n^2)$, $O(2^n)$).

Recursion

Recursion is a powerful technique in programming that involves a function calling itself.

- **Recursion** is a technique where a function calls itself to solve smaller instances of the same problem. It breaks down a problem into smaller subproblems and solves each subproblem recursively until a base case is reached.
- It often leads to **elegant and concise solutions**, especially for problems that exhibit a recursive structure. Recursive solutions are usually shorter in syntax than their iterative counterparts.
- However, **improper use of recursion** can lead to stack overflow errors, especially when dealing with deep or infinite recursion. Additionally, some recursive algorithms may be less efficient than iterative solutions due to the overhead of function calls.

Project Euler

Project Euler is a platform offering a range of challenging mathematical and programming problems.

- **Project Euler** is a collection of mathematical and computer programming problems. These problems are designed to be challenging and require creative problem-solving skills.
- It is a great place to improve your skills of solving programming problems.
- Problems featured on Project Euler **span a wide range of difficulty levels**, from relatively simple tasks to complex puzzles.