Base information about the Learning Unit

Fabian Stiewe

Academic Year 2024-2025

Contents

1 School and Year

This learning unit is designed for the 4th year of the **ITT - Istituto Tecnico Tecnologico**. It will be taught at the beginning of the academic year.

2 Materials

For the theory part the students need a notebook, pen and a mobile device to use the platform algo learn. For the practice part we will go to the laboratory, where the students will have access to a computer with their preferred programming language installed.

3 Topics to be covered

The main topic is theoretical computer science, inside this huge field we will focus on the following topics:

- Sorting algorithms
  - Bubble sort
  - Selection sort
  - Merge sort
  - Sorting on presorted data
- Searching algorithms
  - Linear search
  - Binary search
  - Hash tables

Since the students will also implement every algorithm and use them in a real application, they will also learn how to use a programming language to solve problems.

3.1 Objectives Sorting Algorithms

3.1.1 Bubble Sort

The students should understand how the **bubble sort** algorithm works and be able to implement it in their preferred programming language. They should also be able to analyze the time complexity *(not with big-O notation)* of the algorithm and understand why it is not efficient for large data sets.

3.1.2 Selection Sort

The students should understand how the **selection sort** algorithm works and be able to implement it in their preferred programming language. They should also be able to analyze the time complexity *(not with big-O notation)* of the algorithm and understand why it is not efficient for large data sets.

3.1.3 Merge Sort

The students should understand how the **merge sort** algorithm works and be able to implement it in their preferred programming language. A time complexity is **not** required, but the students should understand that it is more efficient than bubble sort and selection sort.

3.1.4 Sorting on Presorted Data

Given different data sets, the students should be able to choose the best sorting algorithm for each data set. They should also be able to explain why a specific sorting algorithm is better for a specific data set.

In example: If the data set is already sorted, except for k elements, which sorting algorithm should be used?

3.2 Objectives Searching Algorithms

3.2.1 Linear Search

The students should understand how the **linear search** algorithm works and be able to implement it in their preferred programming language. They should also be able to analyze the time complexity *(not with big-O notation)* of the algorithm and understand why it is not efficient for large data sets.

3.2.2 Binary Search

The students should understand how the **binary search** algorithm works and be able to implement it in their preferred programming language. They should also be able to analyze the time complexity *(not with big-O notation)* of the algorithm and understand why it is more efficient than linear search.

3.2.3 Hash Tables

The students should understand how **hash tables** work and be able to use them in their preferred programming language. They should understand why it is efficient for searching and inserting elements and be able to explain why it is more efficient than binary search (in special cases).

3.3 Short Objectives Summary

The students will be able to implement different sorting and searching algorithms and will see cases from real application where those algorithms are used.

4 Perquisites

There are no formal prerequisites for this learning unit. However, it is recommended that students have a basic understanding of the following topics:

- Basic programming concepts
  - The programming language does not matter, each student can use the language they are most comfortable with
  - Variables, loops, functions, arrays, etc. should be known
- Basic mathematics concepts
  - Basic algebra
  - log function
  - Very basic probability (there won't be a proof for hash tables, but it helps to understand the concept)

5 Time plan for the Learning Unit

The learning unit will take 9 weeks (6 hours a week). Each week will be divided into two parts: **theory** (2h) and **practice** (4h). The theory part will be taught first, and the practice part will be taught (in the laboratory) in the second half of the week. During the practice part, the students will get a simple problem and should implement a solution using the algorithms they learned in the theory part and adapt them to the corresponding problem (logical thinking), so learn **programming**.

5.1 Week 1: Bubble Sort

Using a presentation, the students will get taught how the bubble sort algorithm works. They will also get some examples and the "Pseudocode" of the algorithm.

The students will play two little games, first they will sort each other by height using the bubble sort algorithm. Second they will learn the algorithm (*also at home*) using the platform algo learn **(Only the part for sorting algorithms)**.

In the laboratory, the students will get a simple problem and should implement a solution using the bubble sort algorithm.

Problem for the laboratory

> To start the course the students will get a simple problem: Given an array of integers, sort the array using the bubble sort algorithm. The array should be sorted in ascending order.
> For those who are fast: Given an array of tuples, the first entry is a string and the second entry is an integer. First sort the array by the integer in ascending order, then sort the array by the string in descending order.

This is a good start to warm up programming skills of each student.

5.2 Week 2: Selection Sort

Using a presentation, the students will get taught how the selection sort algorithm works. They will also get some examples and the "Pseudocode" of the algorithm. Also, an application example will be shown, quite similar to the problem for the laboratory.

They will again use the platform algo learn to learn the **selection sort** algorithm.

Problem for the laboratory

> You are given an array of strings `names`, and an array `heights` that consists of distinct positive integers. Both arrays are of length n. For each index i, `names`$[i]$ and `heights`$[i]$ denote the name and height of the ith person. Return `names` sorted in descending order by the people's heights.

The problem will should be solved using the selection sort algorithm. The students will get test cases to check if their implementation is correct.

5.3 Week 3: Merge Sort

Again using a presentation and the following YouTube video: Merge Sort Animations. The students will get taught how the merge sort algorithm works. They will also get some examples and the "Pseudocode" of the algorithm.

They will again use the platform algo learn to learn the **merge sort** algorithm.

Problem for the laboratory

> You are given an array of integers. Sort the array using the merge sort algorithm. The array should be sorted in ascending order.
> For those who are fast: Count how many inversions are in the array. An inversion is a pair of indices (i, j) such that $i < j$ and $a[i] > a[j]$.

5.4 Week 4: Sorting on Presorted Data

Will be created during the course.

5.5 Week 5: Recap and evaluation of the sorting algorithms

Will be created during the course.

5.6 Week 6: Linear Search

Will be created during the course.

5.7 Week 7: Binary Search

Will be created during the course.

5.8 Week 8: Hash Tables

Will be created during the course.

5.9 Week 9: Recap and evaluation of the searching (and sorting) algorithms

Will be created during the course.

6 Support for Diverse Learning Needs

If a student is learning *slower* than the others, the student will get additional help. There are most of the time many diverse videos and online sources, which can help the student to understand the topic. During the laboratory, the student will get additional help from the teacher, by getting hints and tips to solve the problem.

7 Evaluation

7.1 Outline

The main part of the evaluation will be the programming during the laboratory. The students will get a problem and should implement a solution using the algorithms they learned in the theory part. The implementation should be correct and efficient, or at least the students should be able to explain why their implementation is not efficient. **This will make approximately 60% of the final grade.**

Above that, the Mini-Tests (in week 5 and 9) will be used to evaluate the understanding of the algorithms. The students should be able to explain how the algorithms work and why they are efficient or not efficient.

In each task the students have to score at least 40% points.

7.2 Mini-Test Week 5

7.2.1 No. 1 - Bubble Sort

You are given the following list of numbers:

$$[5, 3, 8, 2, 1, 4, 7, 6]$$

1. Sort the list using the bubble sort algorithm. Write down the list after each iteration. (5P)
2. What is the time complexity of the bubble sort algorithm? (2P)
3. Why is the bubble sort algorithm not efficient for large data sets? (2P)
4. What is the best case time complexity of the bubble sort algorithm? (1P)

7.2.2 No. 2 - Selection Sort

You are given the following list of numbers:

$$[12, 3, 5, 7, 1, 9, 4, 6]$$

1. Please sort the list using the selection sort algorithm. Write down the list after each iteration. (5P)
2. Explain the difference between the bubble sort and the selection sort algorithm. (3P)

7.2.3 No. 3 - Merge Sort

You are given the following list of numbers:

$$[8, 2, 4, 6, 3, 1, 5, 7]$$

1. Please sort the list using the merge sort algorithm. Write down the list after each iteration. (4P)
2. What is the time complexity of the merge sort algorithm? (2P)
3. Why is the merge sort algorithm more efficient than the bubble sort and selection sort algorithm? (2P)

7.2.4 No. 4 - Sorting on Presorted Data

Given will be the following list of numbers, choose the best sorting algorithm for each list and explain why you choose this algorithm. (each 3P)

1. $[1, 2, 3, 4, 5, 6, 7, 8]$
2. $[8, 7, 6, 5, 4, 3, 2, 1]$
3. $[8, 2, 4, 6, 3, 1, 5, 7]$

7.3 Mini-Test Week 9

This will be created during the course.