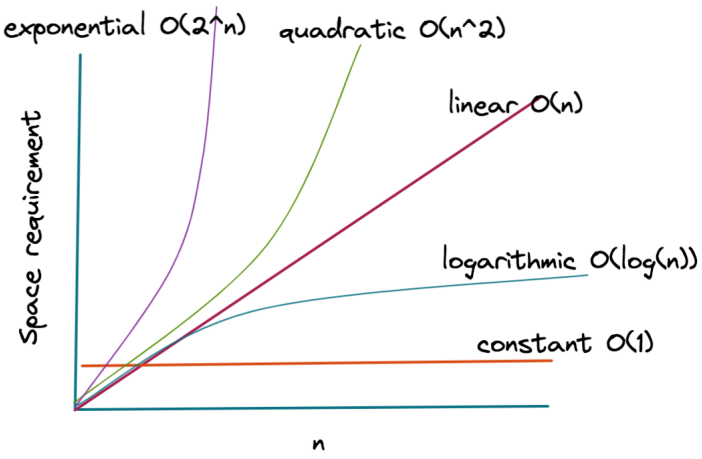

LINMA2111 - Discrete mathematics II

Simon Desmidt
Issambre L'Hermite Dumont

Academic year 2025-2026 - Q1

UCLouvain

Contents

Introduction

1.1 Sorting problems

1.1.1 Introduction

A sorting problem is a problem that consists of taking a sequence of n objects and putting them in order. This kind of problem is made of three main elements:

- Context: set S with a partial order $<$;
- Input: n elements of S;
- Output: permutation of the input elements respecting the order.

To prove the correctness of an algorithm, we generally use the Hoare triple, i.e. a tuple for any input array x_0:

$$\begin{aligned} &\{\text{Algorithm to be used; Precondition; Postcondition}\} \\ &\{IS;\ x = x_0;\ x \text{ is sorted and is a permutation of } x_0\} \end{aligned} \tag{1.1}$$

where IS is the insertion sort algorithm, and x is the sorted array. From now on, we will call "sorting" a sorted permutation.
In practice, to prove the correctness of an algorithm, we define the invariant and the base case, and do an induction step show that the invariant is preserved.

1.1.2 Definition of complexity

For some functions $f, g : \mathbb{N} \to \mathbb{R}^+$,

$$\begin{aligned} &f \in \mathcal{O}(g) \iff \exists c > 0, \exists n_0, \forall n > n_0,\ f(n) \leq cg(n) \\ &f \in \Omega(g) \iff g \in \mathcal{O}(f) \\ &f \in \Theta(g) \iff f \in \mathcal{O}(g) \text{ and } f \in \Theta(g) \\ &f \in o(g) \iff \exists c > 0, \exists n_0, \forall n > n_0,\ f(n) < cg(n) \\ &f \in \omega(g) \iff g \in o(f) \end{aligned} \tag{1.2}$$

- The time complexity is the number of operations as a function of the input size;
- The space complexity is the amount of memory used (in addition to the input) as a function of the input size.

We define the average case complexity as an expectation of the time taken by the algorithm on each input possible, with a dependence in the size of the input.

$$t = \mathbb{E}_{x \sim D}[T(x)] \tag{1.3}$$

where D is the distribution of the input.

1.1.3 Divide and conquer

The divide and conquer method consists in three steps:

1. Divide: create smaller subproblems;
2. Recurse: solve them;
3. Combine: merge the solutions.

In sorting problems, a divide-and-conquer algorithm is the merge sort algorithm. It consists in dividing the array in two, sorting each half recursively, and merging the two sorted halves. The merge operation is done in linear time.

1.2 Complexity of sorting algorithms

No sorting algorithm can be faster than $\Omega(n \log(n))$. This can be proven through the complexity of comparison-based algorithms.