Basics

Chapters 2.1, 2.2 and 3.2 of Cormen’s book

Giulia Bernardini
giulia.bernardini@units.it

Algorithmic Design a.y. 2024/2025

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \dots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
while $j \leq n$ **do**
  if $A[j] = q$ **then**
    return j;
  else
    $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
while $j \leq n$ **do**
    if $A[j] = q$ **then**
        return j;
    else
        $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

Problem definition

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
while $j \leq n$ **do**
    if $A[j] = q$ **then**
        return j;
    else
        $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

Variable initialization: all variables must be initialized to some value.

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
while $j \leq n$ **do**
    if $A[j] = q$ **then**
        return j;
    else
        $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

Return: returns the value and terminates the algorithm. Correct algorithms must **always return something**

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
while $j \leq n$ **do**
    if $A[j] = q$ **then**
        return j;
    else
        $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

Indentation: the if/else and while conditions only refer to the instructions more indented than them. All instructions on the same vertical line are executed anyway (until the algorithm terminates)

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
while $j \leq n$ **do**
    if $A[j] = q$ **then**
        return j;
    else
        $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

q=6, *n*=5
A[1,…,5]=[3,6,5,4,9]

Instance of the problem

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$\boxed{j \leftarrow 1;}$
while $j \leq n$ **do**
    if $A[j] = q$ **then**
        return j;
    else
        $j \leftarrow j + 1;$
if $j = n + 1$ **then return : FAIL**;

---

q=6, n=5
A[1,…,5]=[3,6,5,4,9]
j=1

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
OK
while $\boxed{j \leq n}$ **do**
  if $A[j] = q$ **then**
    return j;
  else
    $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

q=6, n=5
A[1,…,5]=[3,6,5,4,9]
j=1

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
while $j \leq n$ **do**
    if $\boxed{A[j] = q}$ **then**
NO   **return** j;
    else
        $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

q=6, n=5
A[1,…,5]=[3,6,5,4,9]
j=1
A[1]=3≠q

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1,\ldots,n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
while $j \leq n$ **do**
  if $A[j] = q$ **then**
    return j;
  else
    $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

q=6, *n*=5
A[1,…,5]=[3,6,5,4,9]
j=2

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
OK
while $\boxed{j \leq n}$ **do**
  if $\boxed{A[j] = q}$ **then**
  YES! **return** j;
  else
    $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

q=6, n=5
A[1,…,5]=[3,6,5,4,9]
j=2
A[3]=6=q!

What is an Algorithm?

An algorithm is a finite sequence of step-by-step, well-defined instructions that takes some value, or set of values, as input and produces some value, or set of values, as output.

An algorithm can be described by a pseudocode, which can then be converted in any programming language to obtain a working software.

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
while $j \leq n$ **do**
    if $A[j] = q$ **then**
        $\boxed{\textbf{return } j;}$ → OUTPUT: index 2
    else
        $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

---

q=6, n=5
A[1,…,5]=[3,6,5,4,9]
↑
j=2
A[2]=6=q!

Quiz Time

Please go to **www.wooclap.com,** use the code **BERNARDINI1** and answer the question (it is anonymous unless you decide to use your name). You do not need to create an account!

Complexity of Algorithms

The time complexity of an algorithm is the number of “steps” done by the algorithm as a function of the number of input elements.

To offer guarantees to the user on the performance of the algorithm even with the most unfortunate input, we carry on a worst-case analysis. So we want to compute upper bounds on the running time.

Complexity of Algorithms

We are mainly interested in the order of growth of an algorithm: the fastest-growing term of the function that expresses the number of steps done by the algorithm in the worst case.

- Linear algorithm: $an + b$ steps on the worst-case input of size n
- Quadratic algorithm: $cn^2 + dn + e$ steps on the worst-case input of size n
- Logarithmic algorithm: $k\log(n) + h$ steps on the worst-case input of size n
- …

Complexity of Algorithms

We are mainly interested in the order of growth of an algorithm: the fastest-growing term of the function that expresses the number of steps done by the algorithm in the worst case.

- Linear algorithm: $an + b$ steps on the worst-case input of size n
- Quadratic algorithm: $cn^2 + dn + e$ steps on the worst-case input of size n
- Logarithmic algorithm: $k\log(n) + h$ steps on the worst-case input of size n
- …

We just don't care about the multiplicative constants and the lower-order terms!

Why Asymptotic Time Matters

Snellius (Amsterdam):
10^{16} instructions/second

My laptop:
10^{10} instructions/second

Snellius is 1 million times faster than my laptop!

You would believe that the most stupid algorithm executed by Snellius takes less time than a very efficient algorithm run on my laptop, but…

Why Asymptotic Time Matters

Snellius (Amsterdam):
10^{16} instructions/second

My laptop:
10^{10} instructions/second

Suppose we have two algorithms for the same task:

Algorithm A requires $2n^2$ instructions for an input of size n;

Algorithm B requires $120n$ instructions for the same input.

Let's take n=10^{10} and run Algo A on Snellius, Algo B on my laptop.

Why Asymptotic Time Matters

Snellius (Amsterdam):
10^{16} instructions/second

My laptop:
10^{10} instructions/second

Suppose we have two algorithms for the same task:

Algorithm A requires $2n^2$ instructions for an input of size *n;*

Algorithm B requires $120n$ instructions for the same input.

Let's take n=10^{10} and run Algo A on Snellius, Algo B on my laptop.

$$\frac{2 \cdot (10^{10})^2 \text{instructions}}{10^{16} \text{instructions/second}} = 20000 \text{seconds}$$

$$\frac{120 \cdot (10^{10}) \text{instructions}}{10^{10} \text{instructions/second}} = 120 \text{seconds}$$

Counting the Instructions

How many instructions does this algorithm execute in the worst case?

Algorithm 1 Linear Search

INPUT: An array $A[1, \ldots, n]$ of numbers and a number q.
OUTPUT: An index i such that $A[i] = q$; or **FAIL** if no such index exists.
$j \leftarrow 1$;
while $j \leq n$ **do**
  if $A[j] = q$ **then**
    return j;
  else
    $j \leftarrow j + 1$;
if $j = n + 1$ **then return** : **FAIL**;

Quiz Time

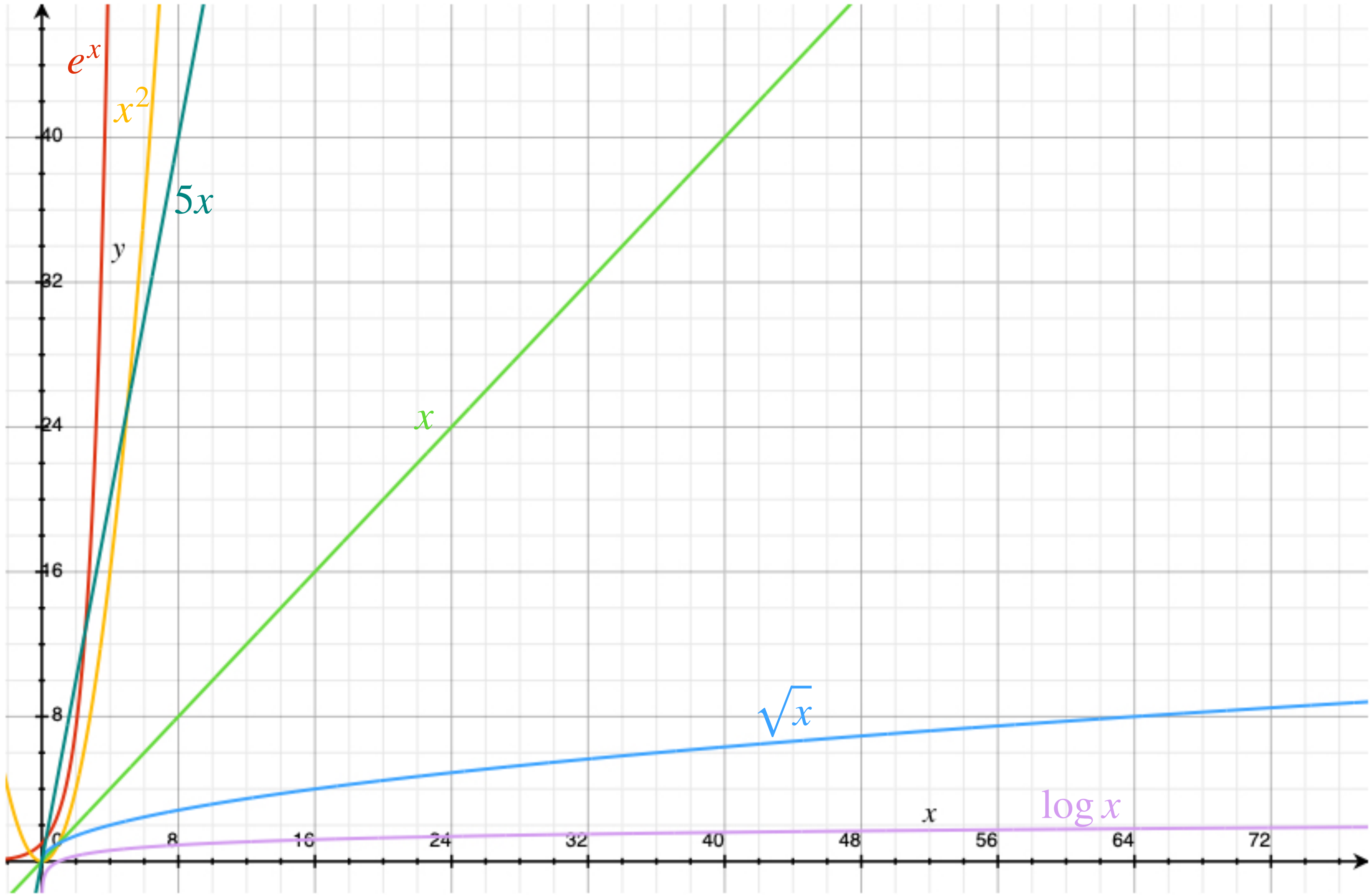
Please go to **www.wooclap.com,** use the code **BERNARDINI1** and answer the question (it is anonymous unless you decide to use your name). You do not need to create an account!

Asymptotic Analysis

Basic idea of asymptotic analysis:

- Ignore the machine-dependent constants
- Look at the growth of the number of instructions as a function of the input size n, when $n \to \infty$.
- We will express the time complexity of algorithms using the asymptotic notation O, Θ, Ω, o, ω (defined in Cormen, chapter 3.1 and at the blackboard).

Orders of Growth



$$\log x = O(\sqrt{x}) = O(x) = O(x^2) = O(e^x)$$

The RAM Model

In order to analyse algorithms we need to fix a model of computation.

The Random Access Machine model is the most widely used. It models Random Access Memory of real computers.

In the RAM model we can access, compute and store a constant number of words in constant time.

Quiz Time

Please go to **www.wooclap.com,** use the code **BERNARDINI1** and answer the question (it is anonymous unless you decide to use your name). You do not need to create an account!