Giulia Bernardini
giulia.bernardini@units.it
Office: 3.29 (C5, 3rd floor)

Algorithmic Design / Algorithms for Scientific Computing
a.y. 2023/2024

What can you expect from this course?

Algorithmics tells us:

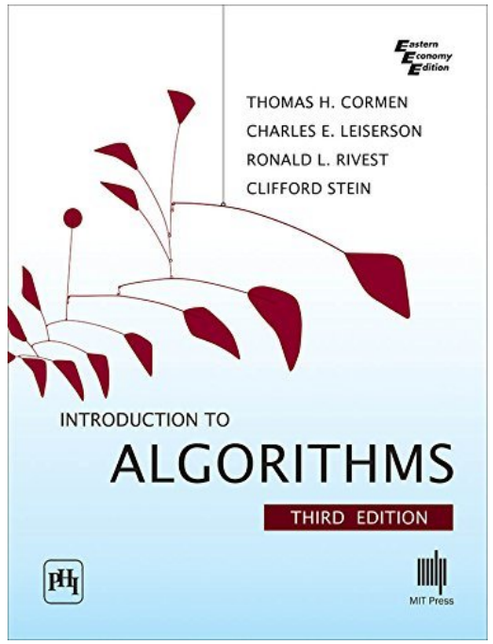
- Whether a program will be effective before coding it
- How to estimate the execution time of a program
- Whether the program strategy can be improved

You will learn how to:

- Abstract the notion of program (pseudocodes)
- Define a measure of efficiency/complexity
- Actually compute this measure for existing algorithms
- Design algorithms that perform well for this measure

Material

- **Textbook:** *Introduction to Algorithms* (3rd Edition) Cormen, Leiserson, Rivest, Stein. MIT Press.



- **Chapters from other books** that I will provide
- **Exercises**
- **Slides (sometimes). Disclaimer:** the slides are not enough to pass the examination. You need to take notes/read the book
- **Video recordings**

Moodle page: https://moodle2.units.it/course/view.php?id=12349

Team: CD2023 587SM-2 ALGORITHMIC DESIGN - code: 7rdiaz2

Lectures

- **Tuesdays** 11.00-13.35 in room 1B of building D, with two breaks of 10 minutes in between

- **Wednesdays** 11.00-13.35 in room 5B of building H2bis with two breaks of 10 minutes in between

- **Thursdays** 14.15-16 in room 4C of building H2bis with one break of 15 minutes in between

- **No lectures on:** 14 March, 28 March - 4 April

- **Last lecture:** 24 April

Examination

- A **written theory test** over the whole content of the course

- **10/15 minutes** oral presentation (provided you passed the test) about a recent paper individually assigned to each of you at the end of the course. **Important**: longer presentations will be penalised. To learn how to identify and present only the most important aspects of a problem and its solution is an essential part of this course.

- The presentation and the theory test are given separate grades. The final grade is given by 40% the presentation's grade + 60% the theory test's grade, **provided they are both above the passing mark**.

Wooclap

Go to **www.wooclap.com** and use the code **BERNARDINI00**

You do not need to create an account and you can answer to questions anonymously!

Answer to Question 4

What quantity does the algorithm below compute?

Algorithm 4 SOME COMPUTATION ON A

INPUT: An array $A = A[1, \ldots, n]$ of integers (positive and negative).
OUTPUT: ???

$i \leftarrow 1$; total $\leftarrow 0$; counter $\leftarrow 0$;
while $i \leq n$ **do**
    if $A[i] > 0$ **then**
        total $\leftarrow$ total $+ A[i]$;
        counter $\leftarrow$ counter $+ 1$;
    $i \leftarrow i + 1$;
if counter > 0 **then**
    return $\frac{\text{total}}{\text{counter}}$;
else
    return FAIL;

Answer to Question 4

What quantity does the algorithm below compute?

- The loop is not endless

Algorithm 4 SOME COMPUTATION ON A

INPUT: An array $A = A[1, \ldots, n]$ of integers (positive and negative).
OUTPUT: ???

```
i ← 1; total ← 0; counter ← 0;
while i ≤ n do
    if A[i] > 0 then
        total ← total + A[i];
        counter ← counter + 1;
    i ← i + 1;
if counter > 0 then
    return total / counter;
else
    return FAIL;
```

i is incremented at every iteration of the loop

Answer to Question 4

What quantity does the algorithm below compute?

- It does not always fail

Algorithm 4 SOME COMPUTATION ON A

INPUT: An array $A = A[1, \ldots, n]$ of integers (positive and negative).
OUTPUT: ???

$i \leftarrow 1$; total $\leftarrow 0$; counter $\leftarrow 0$;
while $i \leq n$ **do**
  if $A[i] > 0$ **then**
    total $\leftarrow$ total $+ A[i]$;
    counter $\leftarrow$ counter $+ 1$;
  $i \leftarrow i + 1$;
if counter > 0 **then**
  return $\frac{\text{total}}{\text{counter}}$;
else
  return FAIL;

If there is at least one positive element, counter is positive and the algorithm does not fail

Answer to Question 4

What quantity does the algorithm below compute?

- It skips the negative elements…

Algorithm 4 SOME COMPUTATION ON A

INPUT: An array $A = A[1, \ldots, n]$ of integers (positive and negative).
OUTPUT: ???

```
i ← 1; total ← 0; counter ← 0;
while i < n do
    if A[i] > 0 then
        total ← total + A[i];
        counter ← counter + 1;
    i ← i + 1;
if counter > 0 then
    return total / counter;
else
    return FAIL;
```

total stores the sum of the positive elements;
count stores the number of positive elements

Answer to Question 4

What quantity does the algorithm below compute?

- It returns the average of the positive elements of A!

Algorithm 4 SOME COMPUTATION ON A

INPUT: An array $A = A[1, \ldots, n]$ of integers (positive and negative).
OUTPUT: ???

```
i ← 1; total ← 0; counter ← 0;
while i ≤ n do
    if A[i] > 0 then
        total ← total + A[i];
        counter ← counter + 1;
    i ← i + 1;
if counter > 0 then
    return total / counter;
else
    return FAIL;
```

This is the average of the positive elements, if there are any!