Data Structures and Algorithms

Conf. dr. ing. Guillaume Ducoffe

guillaume.ducoffe@fmi.unibuc.ro

About myself...

- Professor at the Univ. of Bucharest, Faculty of Mathematics and Computer Science.

- Research Scientist at the National Institute of Research and Development in Informatics (ICI).

- PhD in Nice-Sophia Antipolis, France.

- Various visits/internships in the US, Chile, Germany,...

Research in Graph **Algorithms** *and* **Data Structures**

Come to see me if...

- **You want to learn more about the class! =)**

Some useful links:

  - https://arxiv.org/list/cs.DS/recent
  - https://www.siam.org/conferences/cm/conference/soda22

- **You need help for your Bachelor/Master thesis**

  - Prototypal Example: implementations of some algorithms and XP on some real benchmarks.

- **You want to discuss about internship/PhD/job opportunities...**

Practical Information

- All materials (slides + documents) to be put on Teams + Moodle.
- My website – with contact information

  https://sites.google.com/view/guillaume-ducoffes-homepage/home

  Additional documents can be put on it – upon request
- **Attendance is NOT mandatory**. But...
  - Seminars/Labs are not registered
  - Bonus for attendance + participation during the labs (max. 1p)

Your **grades**

- Written test ("*Proba scrisa*"): 60 %
  - Either pseudo code or C++/Python-like code
  - A significant part of the grade devoted to algorithm analysis (justify correctness, complexity, etc.)
- Practical exam ("*Colocviu*"): 40 %
  - In C++ or Python.
  - Two sets of problems to be solved.

Final grade: $\min\{10, .6 * \text{Written Test} + .4 * \text{Practical Exam} + \text{Bonus}\}$

There shall be NO "punct din oficiu".

Discussion: Terminology used in this class

We rarely mention "real" problems in this class (*e.g.*, database indexing).

We rather use a specific formal/abstract terminology (*e.g.*, B-trees).

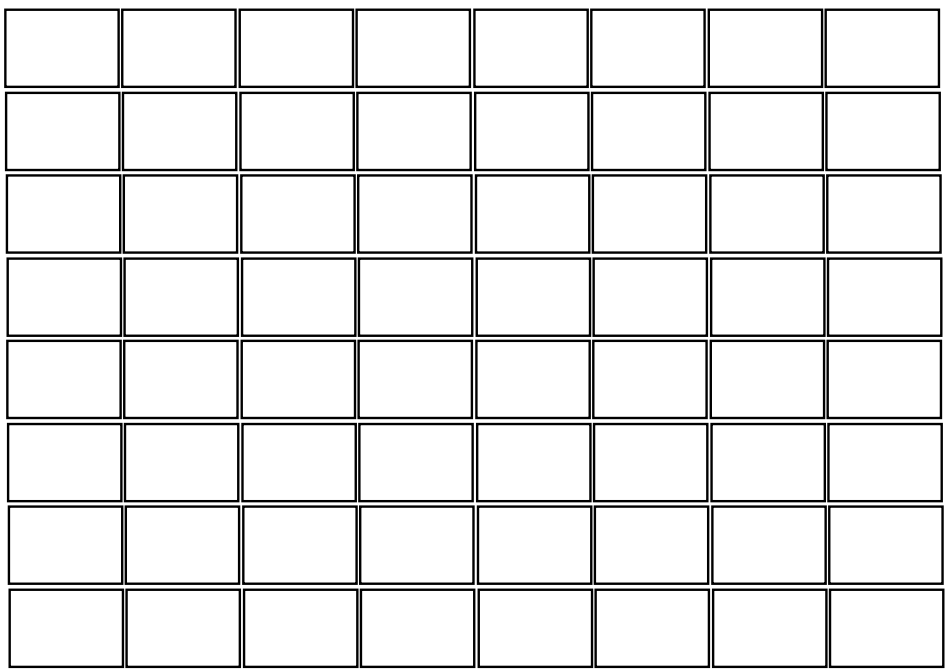
Some reasons for that are:

- **Historical**. Study of these problems started before CS even exists (Euclid, Antic Greece!)
- **Avoiding ill-defined problems**. For instance, what is a "best" road to travel? *Cost? Time? Kilometers? Carbon footprint?*
- **Reusability**. For instance, there are conceptually few differences between a road to travel and routing on the Internet...

Thinking about Data Structures

Storing the grades?

- Model: **R**andom **A**ccess **M**emory: Instruction `Read`/`Write` + Address



Thinking about Data Structures

Storing the grades?

- Model: **R**andom **A**ccess **M**emory: Instruction `Read`/`Write` + Address

Copy 1 Student 1	Copy 1 Student 2	Copy 1 Student 3	Copy 1 Student 4	Copy 1 Student 5			

Thinking about Data Structures

Storing the grades?

- Model: **R**andom **A**ccess **M**emory: Instruction `Read`/`Write` + Address

Copy 1 Student 1	Copy 1 Student 2	Copy 1 Student 3	Copy 1 Student 4	Copy 1 Student 5	Copy 2 Student 1	Copy 2 Student 3	Copy 2 Student 4
Copy 2 Student 2	Copy 2 Student 5						

Thinking about Data Structures

Storing the grades?

- Model: **R**andom **A**ccess **M**emory: Instruction `Read`/`Write` + Address

How to find all the grades of Student 2?

Copy 1 Student 1	Copy 1 Student 2	Copy 1 Student 3	Copy 1 Student 4	Copy 1 Student 5	Copy 2 Student 1	Copy 2 Student 3	Copy 2 Student 4
Copy 2 Student 2	Copy 2 Student 5						

Thinking about Data Structures

Storing the grades?

- Model: **R**andom **A**ccess **M**emory: Instruction `Read`/`Write` + Address

How to find all the grades of Student 2?
Sorting? $\longrightarrow$ method?

Copy 1 Student 1	Copy 1 Student 2	Copy 1 Student 3	Copy 1 Student 4	Copy 1 Student 5	Copy 2 Student 1	Copy 2 Student 2	Copy 2 Student 3
Copy 2 Student 4	Copy 2 Student 5						

Thinking about Data Structures

Storing the grades?

- Model: **R**andom **A**ccess **M**emory: Instruction `Read`/`Write` + Address

How to find all the grades of Student 2?
Sorting? $\longrightarrow$ method?

Copy 1 Student 1	Copy 1 Student 2	Copy 1 Student 3	Copy 1 Student 4	Copy 1 Student 5	Copy 2 Student 1	Copy 2 Student 2	Copy 2 Student 3
Copy 2 Student 4	Copy 2 Student 5						

What if. . . a new student comes? one misses a session?

Thinking about Data Structures

Lessons learned

- Naive storage is not robust against some natural types of events

  Addition/Deletion

- Robustness can be improved through the help of auxiliary operations

  E.g., Sorting

$\Longrightarrow$ Algorithm

Question: is there a "better way" to store the data?

$\Longrightarrow$ Data Structure

Definitions

Data Structures

Data + Structure

- The Data: an arbitrary collection of objects/values

for all this class (unless stated otherwise): Data = Integers

- The data is stored in a Structured manner so as to answer, as efficiently as possible, to some set of **queries**.

a **query** = *an operation to modify and/or access to some information about the data* under some requirements (*e.g.*, complexity)

typical queries: insert/delete some elements with a specified property (max., min., etc.)

a data structure is defined by its set of queries.

Definitions

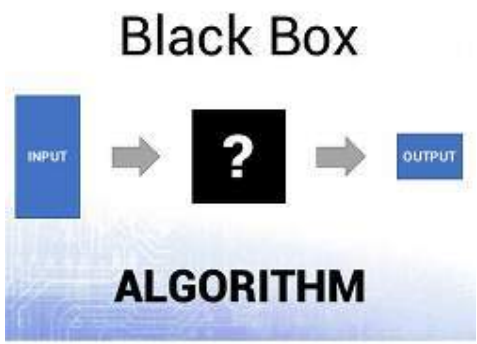
Algorithms

a **problem** = obtaining from some **input** data a specified **output** (*e.g.*, information, transformed data)

Definition (Algorithm)

succession of elementary operations to go from the input to the output

- elementary operations: arithmetic operations, comparisons, if-else, for-while, ...
- Data Structures allow us to define new sets of (non elementary) operations



Objectives for this semester

- Design and Analysis of Data Structures
- Design and Analysis of Algorithms

Design:

- Review of basic data structures & their characteristics
- *Programming techniques*

Analysis:

- *Correctness* of a program
- Complexity

A tentative list of the data structures considered

- Arrays/Vectors
- Stacks
- Tree Data Structures
  - Cartesian trees
  - Link-cut trees
- Heaps
  - Binary Heaps
  - Binomial Heaps
  - Fibonacci Heaps
- Lists
- Queues
- Search trees
  - AVL
  - $2-3-4$ trees
  - B-trees
- Hashing tables
- Union Find
- Partition Refinement

Classification of Data Structures

- Static vs Dynamic
- Data Access (FIFO vs LIFO vs . . .)
- Operations available and their **costs**
- Space requirements, etc.
- Not covered here: persistence, eager/lazy evaluation, . . .

A tentative list of the main problems considered

- Searching
  - Binary search in sorted and "almost sorted" arrays
  - Median and Order statistics
- Sorting
  - "The classics"
  - Cycle sort, Block sort, Tim sort, etc.

Almost all data structures considered have been invented in order to optimize these two fundamental problems!

A tentative list of the techniques considered

Disclaimer: these paradigms are also discussed in more advanced classes

- Divide and Conquer

  Ex.: **tree decompositions** (Heavy-Path, Centroid, Rake-compress)

- Dynamic programming

  Ex.: Mo's algorithm and its variations

- Backtracking

  Applications to Search trees

Questions