Algorithmics and Complexity
Overview of the course

CentraleSupélec – Gif

2023, ST2

Plan

Goals of this course

Algorithmics...

Computer Science methods to solve problems

- Computer Science method : systematic process that can be automated
- Computer Science problem : representation of a question that a computer must be able to answer
  - ➜ Example : Compute $n!$ for any given n

Goals of this course

Algorithmics...

Computer Science methods to solve problems

- Computer Science method : systematic process that can be automated
- Computer Science problem : representation of a question that a computer must be able to answer
  - ➜ Example : Compute $n!$ for any given n

... and complexity

- Complexity of algorithms
- Complexity of problems

Approach

Problems

- Shortest path
- Minimum spanning tree
- Maximum flow
- Bin packing
- *etc.*

Approach

Problems

- Shortest path
- Minimum spanning tree
- Maximum flow
- Bin packing
- *etc.*

Complexity

P and NP classes, polynomial reduction

Approach

Problems

- Shortest path
- Minimum spanning tree
- Maximum flow
- Bin packing
- *etc.*

Complexity

P and NP classes, polynomial reduction

Algorithmics tools

Data structures :

➜ Lists, Stacks, Queues, Trees, Graphs, Dictionnaries. . .

Known algorithmics :

➜ Graph traversal, Greedy algorithms, Dynamic programming. . .

Approach

Problems

- Shortest path
- Minimum spanning tree
- Maximum flow
- Bin packing
- *etc.*

Complexity

P and NP classes, polynomial reduction

Algorithmics tools

Data structures :

➜ Lists, Stacks, Queues, Trees, Graphs, Dictionnaries...

Known algorithmics :

➜ Graph traversal, Greedy algorithms, Dynamic programming...

Algorithms

- Well-known algorithms : Dijkstra, Ford-Fulkerson, Ford-Bellmann...
- Analysis of the algorithm's complexity
- Optimality of the solution

Competences validated from the course C1 + C6.1 + C6.2

- Computer modeling of an engineering problem (*computational thinking*)
- Choice of suitable data structures and resolution techniques
  - ➜ Learn how to transform to a known problem
  - ➜ Solve it numerically (advanced programming)
- Find an optimal solution :
  - ➜ Exact methods and approximate methods
  - ➜ Solution evaluation
- Calculation of the complexity of an algorithm
  - ➜ Time and memory required by the machine
- Determine the complexity class of a problem

Plan

Outline for lectures

- Graph traversal, path search, minimum spanning tree (3 lectures)
- Maximum flow (1 lecture)
- Dynamic programming (1 lecture)
- Complexity of problems (1 lecture)
- Exact methods for NP-hard problems (1 lecture)

Outline for lectures and tutorials/labs (TDs/TPs)

- Graph traversal, path search, minimum spanning tree (3 lectures)
  - ➜ 2 tutorials
- Data structures
  - ➜ 1 lab session
- Maximum flow (1 lecture)
  - ➜ 1 tutorial
- Dynamic programming (1 lecture)
  - ➜ 1 tutorial_practice session
- Problem solving
  - ➜ 1 double lab session
- Complexity of problems (1 lecture)
  - ➜ 1 tutorial
- Exact methods for NP-hard problems (1 lecture)
  - ➜ 1 tutorial on approximate methods
  - ➜ 1 tutorial_practice session

Schedule

7 lectures

5 tutorials (TD) (5 × 1h30) + 2 tutorial_practice sessions (2 × 3h)

Exercices « similar to the final exam »

2 lab sessions (TP) (1h30 + 3h)

Python language

Slides, exercises, final exam are identical for all groups

(French or English...)

Evaluation

In-term evaluation : 20%

- Lab session, presence is mandatory (December 22)
  - ➜ Deposit : code + answers to questions
- 20% if the mark is higher than that of the written exam

  In case of absence *(whatever the reason)* → final exam score *(two sessions))*

Final exam : 80%

- 3h, written exam (similar to tutorials + questions on lectures)
- No concrete programming (but you will be asked to write algorithms)
- All **hand-written** documents + dictionary

Plan

Online supports

Website for this course, reachable from EduNao :

- All slides of the lectures
- Subjects of tutorials and lab sessions
- Additional material :
  - Practice of tutorials (online python subjects)
  - Additional exercises with correction elements

Online supports

Website for this course, reachable from EduNao :

- All slides of the lectures
- Subjects of tutorials and lab sessions
- Additional material :
  - Practice of tutorials (online python subjects)
  - Additional exercises with correction elements

Development environment for tutorial_practice and lab sessions

Visual Studio Code, PyCharm, Spyder... your choice !

➜ IDE with a **debugger**

➜ Test file on Eduano !

You must have it run before the first practical session (TP)

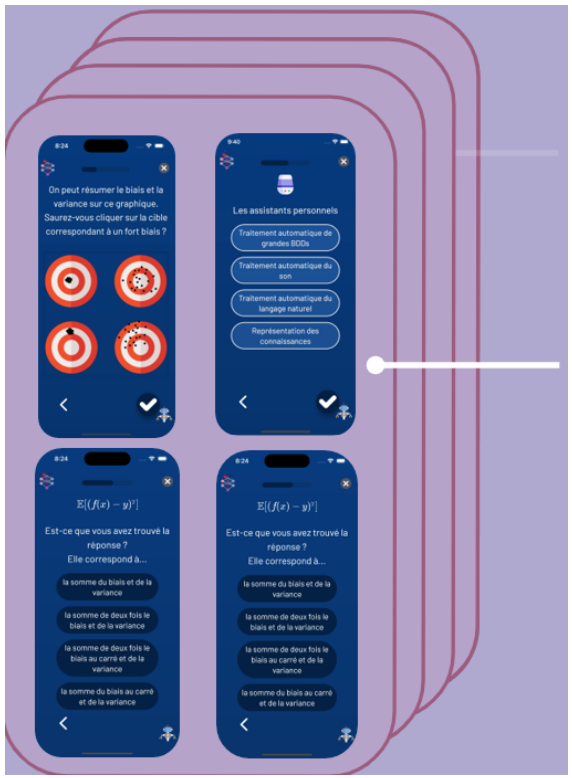
Micro-learning tool : Ariago

Ariago

Micro-learning tool

- Support : website, mobile application, tablet application

➜ You will receive email with link, login and password



Ariago

Micro-learning tool

- Support : website, mobile application, tablet application

➜ You will receive email with link, login and password

Why use Ariago

- Recall important notions of lectures
- Provide additional real-life applications for classical algorithms
- Short exercises to help you better understand the main concepts from the course, and better prepare for the exam

Some advices

About the lectures

- Do not hesitate to ask questions during the lectures !
- Revise the lecture before each tutorial

Some advices

About the lectures

- Do not hesitate to ask questions during the lectures !
- Revise the lecture before each tutorial

Tutorials (TD)

- Participate actively to propose solutions (even erroneous ones) !
- Redo the exercises to train yourself
- Programming : see online practical subjects of tutorials

Some advices

About the lectures

- Do not hesitate to ask questions during the lectures !
- Revise the lecture before each tutorial

Tutorials (TD)

- Participate actively to propose solutions (even erroneous ones) !
- Redo the exercises to train yourself
- Programming : see online practical subjects of tutorials

After each tutorial of all groups, one version with corrective elements will be put online, but that doesn't exempt you from taking notes !

Group levels

- Each track (voie) includes :
  - 1 “beginner” level group (assured by the lecturer)
  - 2 “standard” level groups
  - 1 “confirmed” level group

Group levels

- Each track (voie) includes :
  - 1 “beginner” level group (assured by the lecturer)
  - 2 “standard” level groups
  - 1 “confirmed” level group
- The beginner group will benefit from 5 reinforcement sessions :
  - 1h each : 5 specific prolonged sessions, 1Lecture/2Tutorials/2Lab
  - sessions available on your Géode
  - ➜ take the time to explain further

Group levels

- Each track (voie) includes :
  - 1 “beginner” level group (assured by the lecturer)
  - 2 “standard” level groups
  - 1 “confirmed” level group
- The beginner group will benefit from 5 reinforcement sessions :
  - 1h each : 5 specific prolonged sessions, 1Lecture/2Tutorials/2Lab
  - sessions available on your Géode
  - ➜ take the time to explain further
- The confirmed group will start at the end of the tutorial :
  - additional exercises
  - the practical part

Group levels

- Each track (voie) includes :
  - 1 “beginner” level group (assured by the lecturer)
  - 2 “standard” level groups
  - 1 “confirmed” level group
- The beginner group will benefit from 5 reinforcement sessions :
  - 1h each : 5 specific prolonged sessions, 1Lecture/2Tutorials/2Lab
  - sessions available on your Géode
  - ➜ take the time to explain further
- The confirmed group will start at the end of the tutorial :
  - additional exercises
  - the practical part
- Same exam for everyone, same educational objectives
  - ➜ You will be better prepared if you choose a group suitable for your level

Going further...

Three reference textbooks

- *Introduction to Algorithms, Third Edition*. By Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. MIT Press, 2009.

  The previous edition (1996) is also available in French at Dunod with the title « Introduction à l'algorithmique ».

- *Algorithm Design*. By Jon Kleinberg and Éva Tardos. Pearson Ed. (Addison-Wesley), 2006.

  The PDF version of this book is available on the Internet.

- *Programmation efficace : les 128 algorithmes qu'il faut avoir compris et codés en Python au cours de sa vie*. By Christophe Dürr and Jill-Jênn Vie. Ellipse, 2016.

  In French, Chinese, English (soon)...