1CC2000 – Algorithmic and Complexity

Instructors: Lina Ye
Department: DÉPARTEMENT INFORMATIQUE
Language of instruction: FRANCAIS, ANGLAIS
Campus: CAMPUS DE RENNES, CAMPUS DE PARIS - SACLAY, CAMPUS DE METZ
Workload (HEE): 60
On-site hours (HPE): 36,00

Description

The goal of this course is to introduce computer science methods for engineering problem solving. It presents different problem families using theoretical models. It shows how to solve these problems using sequential or parallel algorithms. We question the existence of a solution and we take great care to the quality of the computed solution. We study the complexity of the problems as well as the complexity of the resolution algorithms.

Quarter number
ST2

Prerequisites (in terms of CS courses)
SG1 – Informations systems and programming (ISP)

Syllabus
Lecture #1 : Introduction – decision and optimization problems, solution, algorithm, algorithm complexity, graph representation, non-weighted graph search
Lecture #2 : Directed Acyclic Graph (DAG) traversal and scheduling, weighted graph search (Dijkstra)
Lecture #3 : Minimum spanning tree, Prim and Kruskal algorithms
Lecture #4 : Maximum flow, Ford-Fulkerson, applications
Lecture #5 : Hashing – general principle, algorithms and applications
Lecture #6 : Dynamic programming
Lecture #7 : Complexity of problems, polynomial reduction, NP-completeness
Lecture #8 : Exact methods for solving NP-hard problems : backtracking, Branch & Bound – Traveling Salesman Problem (TSP)

Class components (lecture, labs, etc.)
8x1h30 lectures

14x1h30 Tutorials, including 4 labs and 2 practices tutorials
3h exam

Grading
Written examination: 70 % - all hand-written documents allowed
Interrogations during tutorials (MCQ) : 30 %

Course support, bibliography
A booklet will be given to the students starting sept. 2019.
All slides will be available online.

Students can refer to the following textbooks:

- *Introduction to Algorithms*, Third Edition. By Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. MIT Press, 2009.
- *Algorithm Design*. By Jon Kleinberg et Éva Tardos. Pearson Ed. (Addison-Wesley), 2006.
- *Programmation Efficace : Les 128 Algorithmes Qu'Il Faut Avoir Compris et Codés en Python au Cours de sa Vie.* Par Christophe Dürr et Jill-Jênn Vie. Ellipse, 2016. (in French)

Resources

- Teaching staff (instructor(s) names):
    - Fabrice POPINEAU
    - Arpad RIMMEL
    - Nicolas SABOURET
    - Joanna TOMASIK
    - Benoit VALIRON
    - Marc-Antoine WEISSER
    - Lina YE
- Maximum enrolment (default 35 students): 25 students max per tutorial group
- Software, number of licenses required: Python Integrated Development Environment (VSCode, etc)

Learning outcomes covered on the course
After this course, students will be capable of :
- Computational thinking, or reasoning with an algorithmic view to solve real-life problems;
- Knowing the general methods to write an algorithm (brute force, divide and conquer, etc) and applying these methods to solve a problem;
- Using exact methods (dynamic programming, branch and bound, etc.) as

well as heuristics (greedy, A*, etc.) to obtain approximate solutions to an optimization problem;
- Algorithm analysis to estimate the complexity in time and space;
- Studying the class of complexity of a problem so as to select the most relevant problem-solving methods.

Description of the skills acquired at the end of the course
By the end of this course, students will be able to:
- Reason in algorithmic terms to solve real-life problems (computational thinking). Given a real-life problem, students will propose a model, determine the complexity class of the associated computational problem, propose an algorithmic solution to the problem and evaluate the quality of the solution.
In relation to skill C6.1 (Solve a problem numerically)
- Know generic techniques for designing algorithms (brute force, divide and conquer, etc.) and apply them to solve a problem;
- Use exact methods (dynamic programming, branch and bound, etc.) and heuristic methods (greedy, A*, etc.) to obtain approximate solutions to an optimization problem;
- Analyze algorithms and estimate their temporal and spatial complexity;
- Determine the complexity class of a problem to choose the right solutions.
In relation to skill C6.2 (Designing software)
- 4 labs and 2 practical exercises will allow students to implement the algorithms they studied through a Python program.