CS-455

Topics in theoretical computer science

Cursus

<table>
<thead>
<tr>
<th>Cybersecurity</th>
<th>Sem.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Science</td>
<td>MA1, MA3</td>
<td>Opt.</td>
</tr>
<tr>
<td>Informatique</td>
<td>MA1, MA3</td>
<td>Opt.</td>
</tr>
<tr>
<td>Mineur en Informatique</td>
<td>H</td>
<td>Opt.</td>
</tr>
<tr>
<td>SC master EPFL</td>
<td>MA1, MA3</td>
<td>Opt.</td>
</tr>
</tbody>
</table>

Language: English
Credits: 4
Session: Winter
Semester: Fall
Exam: During the semester
Workload: 120h
Weeks: 14
Hours: 4 weekly
Lecture: 3 weekly
Exercises: 1 weekly
Number of positions: 1

Remarque
pas donné en 2018-19

Summary
The students gain an in-depth knowledge of several current and emerging areas of theoretical computer science. The course familiarizes them with advanced techniques, and develop an understanding of fundamental questions that underlie some of the key problems of modern computer science.

Content

• Examples of topics to be covered include:

  • Streaming: given a large dataset as a stream, how can we approximate its basic properties using a very small memory footprint? Examples that we will cover include statistical problems such as estimating the number of distinct elements in a stream of data items, finding heavy hitters, frequency moments, as well as graphs problems;

  • Sketching and sampling: what can we learn about the input from a few carefully designed measurements (i.e. a ‘sketch’) of the input, or just a few samples of the input? We will cover results in sparse recovery and property testing that answer this question for several fundamental problems;

  • Sublinear runtime: which problems admit solutions that run faster than it takes to read the entire input? Examples include sublinear time algorithms for graph processing problems, nearest neighbor search and Sparse FFT;

  • Communication: how can we design algorithms for modern distributed computation models (e.g. MapReduce) that have low communication requirements? We will discuss graph sketching, a recently developed approach for designing low communication algorithms for processing dynamically changing graphs.

Keywords
streaming, sketching, sparse recovery, sublinear algorithms

Learning Prerequisites

Required courses
Bachelor courses on algorithms, complexity theory, and discrete mathematics.

Learning Outcomes
By the end of the course, the student must be able to:
• Design efficient algorithms for variations of problems discussed in class;
• Analyze formally space/time/communication complexity of randomized algorithms
• Prove space/time/communication lower bounds for variations of problems discussed in class;
• Select appropriately algorithmic tool for big data analysis problem at hand

Teaching methods
Ex cathedra, homeworks, reading

Expected student activities
Attendance at lectures, completing exercises, reading written material

Assessment methods
• Continuous control

Supervision
Office hours       Yes
Assistants        Yes
Others            Electronique forum : Yes

Resources
Bibliography

There is no textbook for the course. Notes will be posted on the course website.

Ressources en bibliothèque
• Randomized Algorithms / Motwani