Algorithmics and Programming II

• Lecturers:
  – Jordi Cortadella (jordi.cortadella@upc.edu)
  – Jordi Petit (jpetit@cs.upc.edu)

• Sessions:
  – Theory & Problems (Jordi C.)
  – Lab (Jordi P.)

• Languages:
  – English (Theory)
  – Catalan (Problems & Lab)
Material

• Slides, exercises (still under construction):
  https://www.cs.upc.edu/~jordicf/Teaching/AP2

• Jutge (for lab sessions):
  https://jutge.org

• Lliçons (by J. Petit and S. Roura):
  https://lliçons.jutge.org
Evaluation

• Evaluation items:
  – Project (Proj), Parcial Lab (PLab), Final Theory (FTh), Final (FLab).

• Grading:
  – $N_1 = 0.2 \text{ Proj} + 0.2 \text{ PLab} + 0.3 \text{ FTh} + 0.3 \text{ FLab}$
  – $N_2 = 0.2 \text{ Proj} + 0.4 \text{ FTh} + 0.4 \text{ Flab}$
  – $N = \max(N_1, N_2)$
Projects

• **Class for Polynomials:**
  – Design a class to operate with polynomials (evaluation, addition, multiplication, division, gcd, and Fast Fourier Transform (FFT)).
  – Language: C++.

• **uoogle (micro-Google):**
  – Design a web search application.
  – Optional: implement a simple page rank algorithm.
  – Language: python.
Peer and self assessment

• One of the projects (polynomials) will be evaluated by the students themselves.

• Each project will be evaluated by three students. The grade will be calculated as the average grade given by the students.

• The evaluation will be completely blind.

• Biased evaluations will be detected and penalized.

• Each student will have the right to request the evaluation by the professor (who can upgrade or downgrade the evaluation given by the students).
Objective

Confronting large and difficult problems. How?

- Skills for abstraction and algorithmic reasoning.
- Design and use of complex data structures.
- Techniques for complexity analysis.
- Methodologies for modular programming.
- High-quality code.
Compute the convex hull of \( n \) given points in the plane.
The Closest-Points problem

- **Input:** A list of \( n \) points in the plane \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \)
- **Output:** The pair of closest points
- **Simple approach:** check all pairs \( \rightarrow O(n^2) \)
- We want an \( O(n \log n) \) solution!
Navigation: find the shortest path

- Frankfurt
  - Mannheim: 85 km, 217 km
  - Würzburg: 80 km, 186 km
  - Stuttgart: 173 km, 103 km, 183 km
- Karlsruhe
- Erfurt: 250 km
- Nürnberg: 502 km
- Augsburg: 167 km, 84 km
- München
The secret: training, training, training, training ...