Introduction

Grau-AA

Carme Àlvarez i *Maria Serna*

- *e-mail*: alvarez@cs.upc.edu
- *Consultes*: Hores a convenir
- *Nota Contínua*:

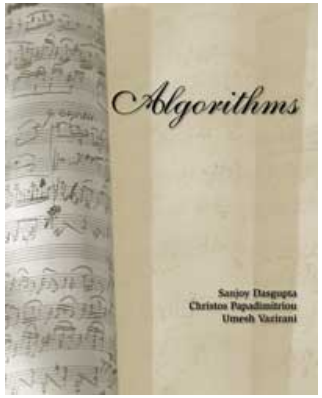
  Problemes 20% +1er Parcial 40% + 2on Parcial 40%
- *Nota del curs*:

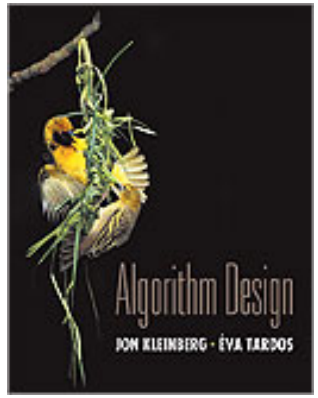
  Si no es va a l'examen final: *Nota Contínua*

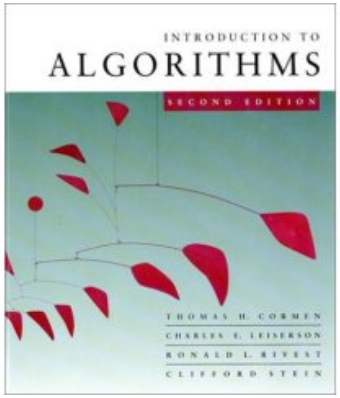
  Si s'entrega l'examen final:
  max{ Nota Exfinal, (Nota Contínua + Nota ExFinal)/2}

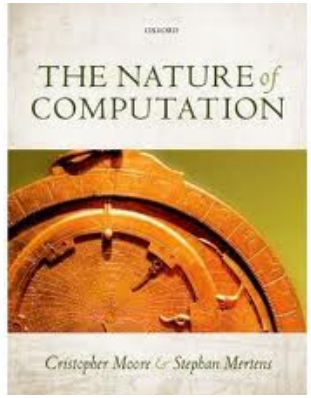
Bibliografia.

Referències principals:









Algorithms.

> *Technology improves things by a constant factor, good algorithm design can do better.* R. Sedgewick

Algorithm: method for solving a problem,

An algorithm is correct if for any given input it returns a solution.

Given an algorithm, we have to study:

- Is it correct? (Correctness)
- How much computational resources it uses? (Efficiency)

Our aim: To design algorithms to solve problems *efficiently*.

In this course, we emphasize the theoretical study of efficient algorithms to solve computationally hard problems.

- *Analysis of the problem:* Theoretical study of the computational complexity, i.e. the amount of resources needed to solve the problem.

  Hard problems versus Easy problems
  LONGEST PATH versus SHORTEST PATH
  HAMILTONIAN GRAPH versus EULERIAN GRAPH
  ...

- *Design:* an algorithm, which is correct and efficient (*polynomial time or less*)

Coping with intractability

Let us imagine that your current task is to write code for solving a simple-looking problem involving graphs and numbers.
What are you supposed to do?

- If you are very lucky your problem will be among shortest path, minimum spanning tree, maximum flow, etc.) i.e. Tractable Problems
- But chances are that nothing like this will happen: NP-complete or worst ... Longest Path, Hamiltonian Graph, Traveling Salesman Problem, etc.) i.e. Intractable Problems

What to do next?

NP-completeness is not a death certificate - it is only the beginning of a fascinating aventure *S. Dasgupta, C. Papadimitriou, U. Vazirani*

- Randomized Algorithms
  Some of the fastest and most clever algorithms we have, rely on *chance*.
- Parameterized Algorithms
  The problem is really hard for only a very small fraction of inputs.
- Approximation Algorithms
  Algorithms for optimization problems that falls short of the optimum but *never too much*.
- Local Search, Heuristics
  Algorithms with no guarantees on either the running time or the degree of approximation