CSE 417: Algorithms and Computational Complexity

Winter 2001

Instructor: Paul Beame

TA: Gidon Shavit

1

What the course is about

- Design of Algorithms
  - design methods
  - common or important types of problems
  - how to analyze algorithms

2

What the course is about

- Computability
  - Turing machines and ideal computers
  - there are well-defined problems that even ideal computers can't solve
    - e.g. halting problem

3

What the course is about

- Complexity and NP-completeness
  - simply being able to solve problems in principle is not enough
    - algorithms must be efficient, too
  - NP
    - wide class of useful problems whose solutions can be easily checked but not necessarily found efficiently
  - NP-completeness
    - useful for understanding when problems are hard to solve

4

On hardness

- Cryptography (e.g. RSA, SSL in browsers)
  - Secret: p,q prime, say 512 bits each
  - Public: n which equals pxq, 1024 bits
- In principle
  - there is an algorithm that given n will find p and q by trying all 2^{512} possible p's.
- In practice
  - security of RSA depends on the fact that no **efficient** algorithm is known for this

5

Algorithms versus Machines

- We all know about Moore's Law and the exponential improvements in hardware but...
- Example: Numerical linear algebra for weather prediction 1967-1987
- 7 orders of magnitude improvement in speed
  - 3 orders of magnitude improvement in hardware
  - 4 orders of magnitude improvement in algorithms

6

What you'll have to do

- No programming
  - goals of the course are not nitty-gritty programming detail
    - getting them right is of course very important but too time-consuming for the amount of material
- Written homework assignments
  - English exposition and pseudo-code
  - Analysis and argument as well as design
- Midterm & Final Exam

7

Rough Division of Time

- Algorithms (6 weeks)
  - Analysis of Algorithms
  - Basic Algorithmic Design Techniques
  - Graph Algorithms
  - Fast Fourier Transform
  - Pattern Matching & Finite Automata
- Turing Machines & Computability (1.5 weeks)
- Complexity & NP-completeness (2 weeks)

8