What you’ll have to do

- Programming
  - Several small projects and (maybe) a large one
- Written homework assignments
  - English exposition and pseudo-code
  - Analysis and argument as well as design
- Midterm & Final Exam

What the course is about

- Design of Algorithms
  - design methods
  - common or important types of problems
  - how to analyze algorithms
  - correctness proofs
What the course is about

• Complexity and NP-completeness
  – solving problems in principle is not enough
    • algorithms must be **efficient**
  – NP
    • class of useful problems whose solutions can be easily checked but not necessarily found efficiently
  – NP-completeness
    • understanding when problems are hard to solve

Very Rough Division of Time

• Algorithms (7 weeks)
  – Analysis of Algorithms
  – Basic Algorithmic Design Techniques
  – Graph Algorithms
• Complexity & NP-completeness (3 weeks)

Complexity Example

• 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

Algorithms versus Machines

• We all know about Moore’s Law and the exponential improvements in hardware but...
• **Ex:** sparse linear equations over past few decades
  – 10 orders of magnitude improvement in speed
    – 4 orders of magnitude improvement in hardware
    – 6 orders of magnitude improvement in algorithms
Algorithms or Hardware?

Solving sparse linear systems

Source: Sandia, via M. Schultz

Algorithms or Hardware?

Solving sparse linear systems

Source: Sandia, via M. Schultz

Algorithms or Hardware?

The N-Body Problem

Source: T. Quinn