Advanced Algorithmics (6EAP)

courses.cs.ut.ee/2012/algorithmics/

MTAT.03.238

Introduction

Jaak Vilo

2012 Spring
Programming on one slide

• **TRUE/FALSE**
• 3, 7, 2.47, 0.333e-02, ‘a’, “text data”, address ...
• Variables: i, j, sum, A[5], B[7,8], a[i].surname, ...
• + - * / > <= ...

• **for/while**, **if**, **then**, **else**, ...
• **function()**

• Input/output
• java.lang.Math.sqrt()
Goals

• To learn the main concepts and techniques of the algorithm design and analysis – the practical skills and basic theoretical basis

• To be able to choose, (design,) analyze and compare algorithms and data structures

• To learn to learn, use wisely, solve, read, write, and present
Algorithms = basics of CS education

• In your future professional life people EXPECT you to know elementary data structures, algorithms, and be able to think in higher categories.

• We need to develop the language to talk about high-level goals, and you need to be able to implement them – if so needed

• At work no-one is going to teach you mathematics or fundamentals anymore
Algorithms

• **Al-Khwārizmī**

Example

- Wind has blown away the +, *, (, ) signs
- What’s the maximal value?
- Minimal?

2 1 7 1 4 3
• 2 1 7 1 4 3

• (2+1)*7*(1+4)*3 = 21*15 = 315
• 2*1 + 7 + 1*4 + 3 = 17
• Q: How to maximize the value of any expression?

2 4 5 1 9 8 12 1 9 8 7 2 4 4 1 1 2 3 = ?
School mathematics: 11*13

But Al Khwarizmi knew another way to multiply, a method which is used today in some European countries. To multiply two decimal numbers $x$ and $y$, write them next to each other, as in the example below. Then repeat the following: divide the first number by 2, rounding down the result (that is, dropping the .5 if the number was odd), and double the second number. Keep going till the first number gets down to 1. Then strike out all the rows in which the first number is even, and add up whatever remains in the second column.

\[
\begin{array}{cc}
11 & 13 \\
5 & 26 \\
2 & 52 & \text{(strike out)} \\
1 & 104 \\
\hline
143 & \text{(answer)}
\end{array}
\]
13 x 11

\[
\begin{array}{cccc}
1 & 1 & 0 & 1 \\
\times & 1 & 0 & 1 \\
\hline
1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 \\
+ & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
\]

(1101 times 1)
(1101 times 1, shifted once)
(1101 times 0, shifted twice)
(1101 times 1, shifted thrice)

(binary 143)

\[O(n^2)\]
is the same as:

\[
\begin{array}{c}
11 & 13 \\
5 & 26 \\
2 & 52 \quad \text{(strike out)} \\
1 & 104 \\
\hline
143 \quad \text{(answer)}
\end{array}
\]
Analysis of algorithms

• *Theoretical study of computer-program performance and resource usage.*
• *What else is important than performance?*

- Modularity
- Correctness
- Maintainability
- Functionality
- Robustness
- User-friendliness
- Programmer time
- Simplicity
- Extensibility
- Reliability
Why study algorithms and performance?

• Algorithms help us to understand scalability.
• Performance often draws the line between what is feasible and what is impossible.
• Algorithmic mathematics provides a language for talking about program behavior.
• Performance is the currency of computing.
• The lessons of program performance generalize to other computing resources.
• Speed is fun!
Contents

• **Algorithms and Data structures – basics**
• Abstract data types (+ STL, LEDA, …)
• Lists, trees, heaps, graphs (TSP; shortest paths, diameter, …)
• Analysis of algorithm complexity
  – $O(n \log n)$, $o(n \log n)$, $\Theta(n \log n)$, $\Omega(n \log n)$
  – recurrences $T(n) = 2 T(n/2) + n$
  – Turing machines, NP, NP-complete, P-SPACE, …
  – Very tough (NP) problems (“Monkey problem”; unsolvable problems?)
Contents (2)

• Basic designs: **Dynamic programming, divide and conquer, randomised algorithms, ...**
• Deterministic or non-deterministic
• Exact or approximate
• Heuristics (approximations):
  – Greedy Algorithms (packing, set cover, weighted set cover, ...
    ...) local search, tabu search
  – Simulated annealing (e.g. TSP),
  – Genetic algorithms (?), ...
  – Monte Carlo, Las Vegas
Contents (3)

- Sorting, search, dictionary, ...
- Text algorithms
- Numerical algorithms
- Computational geometry
- Online algorithms vs offline
- Parallel algorithms
Skills/goals

• Bit-level understanding of low-level implementation
• Creative use of existing implementations
• Formal analysis, complexity theory
• Typical heuristics and algorithm design elements
- $67 + 56 = ?$
SUM( 10 7 98 82 84 66 62 94 6 25 73 62 65 5 36 983 73 93 77 10 7 98 82 84 66 62 94 6 25 73 62 65 5 36 1 983 1 11 173 93 77 10 7 98 82 84 66 62 94 6 25 73 62 65 5 36 983 73 93 77 10 7 98 1 82 66 62 1 1 94 6 25 73 62 65 5 36 983 73 1 93 77 10 7 98 82 84 66 62 94 6 25 73 62 65 5 36 1 73 93 77 10 7 98 82 84 66 62 94 6 25 73 62 65 5 36 1 65 5 36 983 73 93 77 ) = ?
Answer

• 13,000

• Is checking faster than answering the actual question?
  – SAT(isfiability)
  – check if answer satisfies the formula – quick as compared to finding an answer
Main data models

- Linear
- Tree
- Graph
- Relational data model
All fitted on linear RAM, disk, tape

- Linear
- Tree
- Graph

Linear storage: address of a “slot”
Programming languages:

- Pseudocode
  - primarily this is about pseudocode
- Pointers, arrays, memory handling
  - Low- and high level abstractions
- C/C++
- Java, Python, perl, ...
  - Explicit data structures and algorithms
- Whichever choice: **you must be able to explain it**
- Clarity and simplicity is a key
“The Textbook”

- **Introduction to Algorithms,** Second Edition  
  Thomas H. **Cormen,** Charles E. **Leiserson,** Ronald L. **Rivest** and Clifford **Stein**

  - [http://mitpress.mit.edu/algorithmgs/](http://mitpress.mit.edu/algorithmgs/)  
  - [http://books.google.com/books?id=NLngYyWFI_YC](http://books.google.com/books?id=NLngYyWFI_YC)
• **Algorithmics: The Spirit of Computing** (3rd Edition)

• **David Harel, Yishai Feldman**

• Addison Wesley; 3 edition (June 11, 2004)

  • [http://books.google.com/books?id=txxLovFWkCUC](http://books.google.com/books?id=txxLovFWkCUC)
  • [http://www.wisdom.weizmann.ac.il/~dharel/algorithmics.html](http://www.wisdom.weizmann.ac.il/~dharel/algorithmics.html)
• Foundations of Computer Science: C Edition

• Alfred V. Aho, Jeffrey D. Ullman

• W. H. Freeman (October 15, 1994)
• **The Art of Computer Programming (TAOCP)**
• Donald E. Knuth.
• The Algorithm Design Manual
  – Steven S. Skiena

• Algorithms [ILLUSTRATED]
  – Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani
Mathematics

• **Concrete Mathematics: A Foundation for Computer Science (2nd Edition) (Hardcover)**

  **Ronald L. Graham, Donald E. Knuth, Oren Patashnik**

  • **Hardcover:** 672 pages
  • **Addison-Wesley Professional;**
  • **2 edition (March 10, 1994)**
  • **ISBN-10:** 0201558025 **ISBN-13:** 978-0201558029
• Robert Sedgewick
• Safari Bookshelf ! (UT)
More books

• Parallel algorithms
  – An Introduction to Parallel Algorithms Joseph Jaja

• Heuristics
  – How to Solve it: Modern Heuristics. By Zbigniew Michalewicz, David B. Fogel
  – Edition: 2, illustrated Published by Springer, 2004

• Randomized algorithms

• Complexity theory
  This book is a classic, developing the theory, then cataloguing many NP-Complete problems.
Wikipedia:

- [Algorithm](http://en.wikipedia.org/wiki/Algorithm)
- [List of algorithm general topics](http://en.wikipedia.org/wiki/List_of_algorithm_general_topics)
- [List of algorithms](http://en.wikipedia.org/wiki/List_of_algorithms)
- [List of data structures](http://en.wikipedia.org/wiki/List_of_data_structures)
- [Computational complexity theory](http://en.wikipedia.org/wiki/Computational_complexity_theory)
- [List of NP-complete problems](http://en.wikipedia.org/wiki/List_of_NP-complete_problems)
- [Randomized algorithm](http://en.wikipedia.org/wiki/Randomized_algorithm)
- [Parallel algorithm](http://en.wikipedia.org/wiki/Parallel_algorithm)
- ...