Crazy Datastructures and Algorithms

Jan Dreier, Philipp Kuinke, Peter Rossmanith

05.10.18

Outline

- Research
  - You get a paper (or several) from us
  - Read and understand it
  - Independently search for other relevant sources
- Presentation
  - Present the ideas from the paper
  - At most 45 minutes
  - Afterwards a short discussion
- Essay
  - Summarize the ideas from the paper
  - Hand in via email one month after presentation
  - At most 10 pages

If you want feedback to your presentation or essay email us in a timely manner.

Regular Meetings

Weekly presentations take approximately 1.5 hours.

Possible Dates:

- MON 12:30 - 14:00
- MON 13:30 - 15:00
- MON 14:00 - 15:30
- TUE 12:30 - 14:00
- THU 14:30 - 16:00
- THU 15:30 - 17:00

Schedule

To be anounced on the website.

Possible Structure of a Presentation

- Briefly introduce the Topic.
- Why is it interesting? What are typical applications? What techniques are used?
- Give needed background knowledge. Do a quick refresh so everyone is on the same page.
- Present the paper.
- Put the result into context with other research.

Tips:

- Go slooooooowly. You took a couple months to understand the paper. Do not expect everybody to understand everything immediately.
- Keep the slides clean. Often one diagram is better than a wall of text.
- `https://larc.unt.edu/ian/pubs/speaker.pdf`

Essay

- Feel free to use the same structure as for the presentation
- LaTeX is mandatory (tutorial: `https://www.latex-tutorial.com/tutorials/`)

But:

- Do not simply retell the paper!

Deadlines

- For the next two weeks you can resign without any consequences. Just write us an email.
- Essay deadline: one month after presentation
- After submission: we may tell you to fix something in your essay.

The Papers

1. ELIZA (1 Person)

- Simple chat bot that uses pattern matching
- Questions:
  - How could humans be fooled if it was so simple?
- Sources:
  - `https://en.wikipedia.org/wiki/ELIZA`
  - `https://en.wikipedia.org/wiki/ELIZA_effect`

2. Fast Inverse Squareroot (1 Person)

- $1/\sqrt{x}$ has to be computed often during rendering
- 'floating point' trick that yields fast approximation
- Questions:
  - How does it work?
  - Why is it fast?
  - Still usable today?
- Sources:
  - `https://en.wikipedia.org/wiki/Fast_inverse_square_root`
  - `https://en.wikipedia.org/wiki/Methods_of_computing_square_roots#Approximations_that_depend_on_the_floating_point_representation`

3. Hacker's Delights (1 Person)

- Present most interesting tricks from the book
- Mostly low-level programming
- Sources:
  - Hacker's Delight

4. Implicit Datastructures (1 Person)

- Datastructures that just store elements and the connections are implicit
- Example: Heaps
- Questions:
  - What 'standart' datastructures can be stored implicitly?
  - Where are the limits?
- Sources:
  - `https://core.ac.uk/download/pdf/82247166.pdf`
  - `https://core.ac.uk/download/pdf/82745737.pdf`
  - `https://dl.acm.org/citation.cfm?id=322364`

5. Persistent Datastructures (2 Persons)

- Datastructures that store entire history of modifications
- Allow 'rollback' to an earlier date
- Persistent and fully persistent
- Questions:
  - Give overview over the most interesting examples
- Sources:
  - `https://dl.acm.org/citation.cfm?doid=12130.12142`
  - `https://dl.acm.org/citation.cfm?id=6151`
  - `https://dl.acm.org/citation.cfm?id=365528`

6. Bloom-Filter (1 Person)

- Datastructure that has very fast, but probabilistic tests if an element is contained in it.
- False positives but never false negatives
- Questions:
  - How much faster can it be?
  - What are the trade-offs between speed and probability?
- Sources:
  - `https://en.wikipedia.org/wiki/Bloom_filter`
  - `http://www.eecs.harvard.edu/~michaelm/postscripts/im2005b.pdf`

7. Probabilistically Checkable Proofs (1 Person)

- Checking proofs by examining a small number of bits
- A correct proof will never fail to convince
- An incorrect proof only convinces with small probability
- Questions:
  - How can small errors be detected if only a small part is observed?
  - What language can be used for PCP proofs?
- Sources:
  - `https://pdfs.semanticscholar.org/9fc3/134191b320647ba63dcef03473bceb9f77af.pdf`

8. Computer Aided Proofs (1 Persons)

- Proofs that are so long only a computer can generate them
- Mostly huge case distinctions
- Famous example: 4-color theorem
- Questions:
  - How do these solvers work?
  - Can this be considered a publishable proof, since one rarely gains mathematical insight by brute forcing answers?
- Sources:
  - `https://en.wikipedia.org/wiki/Four_color_theorem`
  - `https://coq.inria.fr/about-coq`
  - `https://coq.inria.fr/a-short-introduction-to-coq`
  - `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.3309`

9. Blockchain (2 Persons)

- Two-part talk:
- 1. Theory
- 2. Application
- Questions:
  - What is a blockchain?
  - What are applications beyond ‘coins’?
- Sources:
  - `https://bitcoin.com/bitcoin.pdf`
  - `https://www.economist.com/briefing/2015/10/31/the-great-chain-of-being-sure-about-things`
  - `https://github.com/decrypto-org/blockchain-papers`

10. Property Testing (1 Person)

- Test if a structure has a certain property or is it far away from having the property
- E.g. Is an array sorted (faster than $O(n)$ but approximate)
- Questions:
  - What properties are testable and how fast?
- Sources:
  - `http://www.cse.psu.edu/~sxr48/pubs/TestingSortednessEncyclopedia.pdf`
  - `http://www.cs.ubc.ca/~nickhar/W12/Lecture21Notes.pdf`

11. Computing Smallest Enclosing Circle (1 Person)

- Geometric problem
- Finding a small circle around points in the plane
- Questions:
  - What is so special about the $O(n \log n)$ algorithm, when one was already presented 14 years earlier, and an $O(n)$ algorithm was given 8 years earlier?
- Sources:
  - `http://www.cs.au.dk/~gerth/slides/sven14.pdf`
  - `https://tidsskrift.dk/daimipb/article/view/6704/5821`

12. $L(2,1)$-labelings of graphs (1 Person)

- A simple 4^k solution, then a very complicated $(3.6\ldots)^k$ solution, then a very simple 3^k solution using a counterintuitive idea
- Questions:
  - Why was this solution not found earlier?
- Sources:
  - `https://www.sciencedirect.com/science/article/pii/S0304397512006548?via%3Dihub`

13. Certifiying Algorithms (2 Persons)

- An algorithms that gives an answer and a correctness proof
- Example: Euclid's Algorithm
- Questions:
  - Is this suggested in modern Software Engineering books?
- Sources:
  - `https://people.mpi-inf.mpg.de/~mehlhorn/ftp/CertifyingAlgorithms.pdf`
  - `https://dl.acm.org/citation.cfm?id=200880`