DSA-CL III — Winter Term 2018-19

Data Structures and Algorithms for Computational Linguistics III

Claus Zinn

Çağrı Çöltekin

Eberhard Karls Universität Tübingen

https://dsacl3-2018.github.io

DSA-CL III course overview

What is DSA-CL III?

- Intermediate-level survey course.
- Programming and problem solving, with applications.
  - Algorithm: method for solving a problem.
  - Data structure: method to store information.
- Second part focused on Computational Linguistics

Prerequisites:

- Data Structures and Algorithms for CL I
- Data Structures and Algorithms for CL II

Lecturers:

- Çağrı Çöltekin
- Claus Zinn

Tutors:

- Marko Lozajic
- Michael Watkins

Slots:

- Mon 12:15 & 18:00
- Wed 14:15 — 18:00 (lab)

Course Materials: https://dsacl3-2018.github.io

Why study algorithms?

Their impact is broad and far-reaching.

Internet. Web search, packet routing, distributed file sharing, ...

Biology. Human genome project, protein folding, ...

Computers. Circuit layout, file system, compilers, ...

Computer graphics. Movies, video games, virtual reality, ...

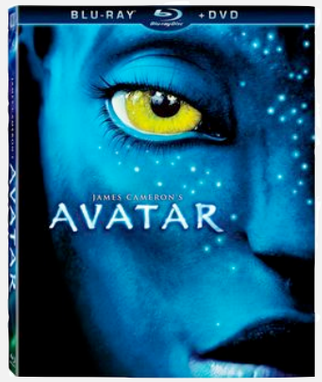
Security. Cell phones, e-commerce, voting machines, ...

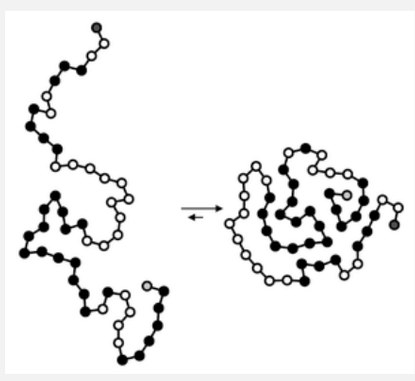
Multimedia. MP3, JPG, DivX, HDTV, face recognition, ...

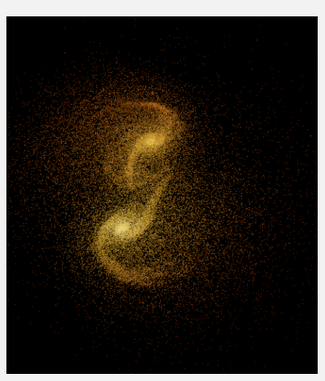
Social networks. Recommendations, news feeds, advertisements, ...

Physics. N-body simulation, particle collision simulation, ...

⋮





Why study algorithms?

Their impact is broad and far-reaching.

Mysterious algorithm was 4% of trading activity last week

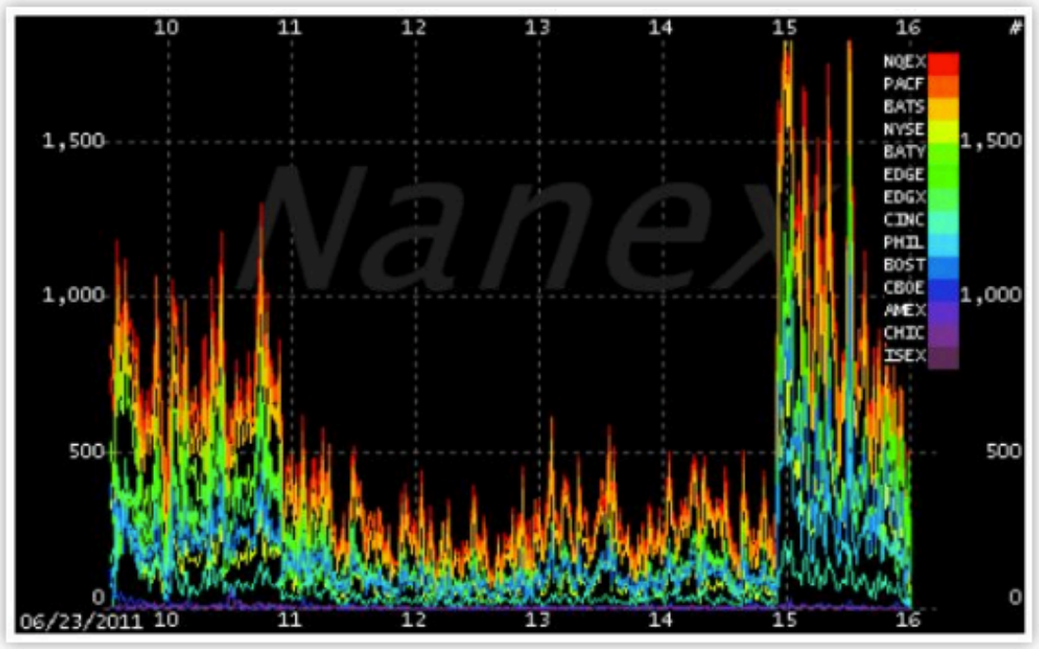
October 11, 2012

A single mysterious computer program that placed orders — and then subsequently canceled them — made up 4 percent of all quote traffic in the U.S. stock market last week, according to the top tracker of high-frequency trading activity.

The motive of the algorithm is still unclear, *CNBC* reports.

The program placed orders in 25-millisecond bursts involving about 500 stocks, according to Nanex, a market data firm. The algorithm never executed a single trade, and it abruptly ended at about 10:30 a.m. ET Friday.

[+]



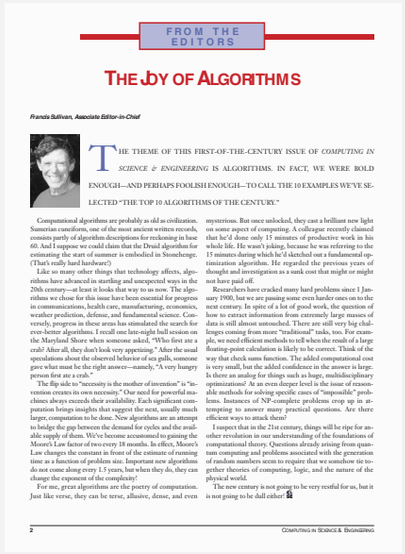
Generic high frequency rrading chart (credit: Nanex)

"My guess is that the algo was testing the market, as high-frequency frequently does," says Jon Najarian, co-founder of TradeMonster.com. "As soon as they add bandwidth, the HFT crowd sees how quickly they can top out to create latency." (*Read More*: Unclear What Caused Kraft Spike: Nanex Founder.)

Why study algorithms?

For intellectual stimulation.

" For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing. " — *Francis Sullivan*



FROM THE EDITORS

THE JOY OF ALGORITHMS

Francis Sullivan, Associate Editor-in-Chief

THE THEME OF THIS FIRST-OF-THE-CENTURY ISSUE OF *COMPUTING IN SCIENCE & ENGINEERING* IS ALGORITHMS. IN FACT, WE WERE BOLD ENOUGH—AND PERHAPS FOOLISH ENOUGH—TO CALL THE 10 EXAMPLES WE'VE SELECTED "THE TOP 10 ALGORITHMS OF THE CENTURY."

Computational algorithms are probably as old as civilization. Sumerian cuneiform, one of the most ancient written records, consists partly of algorithm descriptions for reckoning in base 60. And I suppose we could claim that the Druid algorithm for estimating the start of summer is embodied in Stonehenge. (That's really hard hardware!)

Like so many other things that technology affects, algorithms have advanced in startling and unexpected ways in the 20th century—at least it looks that way to us now. The algorithms we chose for this issue have been essential for progress in communications, health care, manufacturing, economics, weather prediction, defense, and fundamental science. Conversely, progress in these areas has stimulated the search for ever-better algorithms. I recall one late-night bull session on the Maryland Shore when someone asked, "Who first ate a crab? After all, they don't look very appetizing." After the usual speculations about the observed behavior of sea gulls, someone gave what must be the right answer—namely, "A very hungry person first ate a crab."

The flip side to "necessity is the mother of invention" is "invention creates its own necessity." Our need for powerful machines always exceeds their availability. Each significant computation brings insights that suggest the next, usually much larger, computation to be done. New algorithms are an attempt to bridge the gap between the demand for cycles and the available supply of them. We've become accustomed to gaining the Moore's Law factor of two every 18 months. In effect, Moore's Law changes the constant in front of the estimate of running time as a function of problem size. Important new algorithms do not come along every 1.5 years, but when they do, they can change the exponent of the complexity!

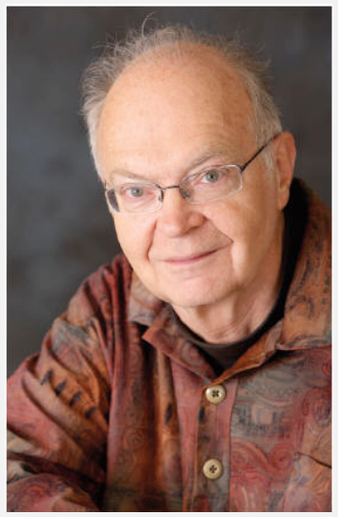
For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing. A colleague recently claimed that he'd done only 15 minutes of productive work in his whole life. He wasn't joking, because he was referring to the 15 minutes during which he'd sketched out a fundamental optimization algorithm. He regarded the previous years of thought and investigation as a sunk cost that might or might not have paid off.

Researchers have cracked many hard problems since 1 January 1900, but we are passing some even harder ones on to the next century. In spite of a lot of good work, the question of how to extract information from extremely large masses of data is still almost untouched. There are still very big challenges coming from more "traditional" tasks, too. For example, we need efficient methods to tell when the result of a large floating-point calculation is likely to be correct. Think of the way that check sums function. The added computational cost is very small, but the added confidence in the answer is large. Is there an analog for things such as huge, multidisciplinary optimizations? At an even deeper level is the issue of reasonable methods for solving specific cases of "impossible" problems. Instances of NP-complete problems crop up in attempting to answer many practical questions. Are there efficient ways to attack these?

I suspect that in the 21st century, things will be ripe for another revolution in our understanding of the foundations of computational theory. Questions already arising from quantum computing and problems associated with the generation of random numbers seem to require that we somehow tie together theories of computing, logic, and the nature of the physical world.

The new century is not going to be very restful for us, but it is not going to be dull either!

" An algorithm must be seen to be believed. " — *Donald Knuth*



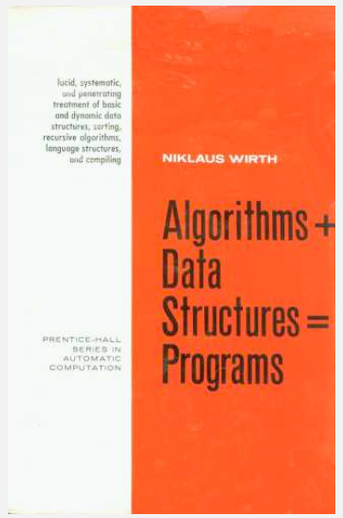
Why study algorithms?

To become a proficient programmer.

" I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships. "

— Linus Torvalds (creator of Linux)

" Algorithms + Data Structures = Programs. " *— Niklaus Wirth*

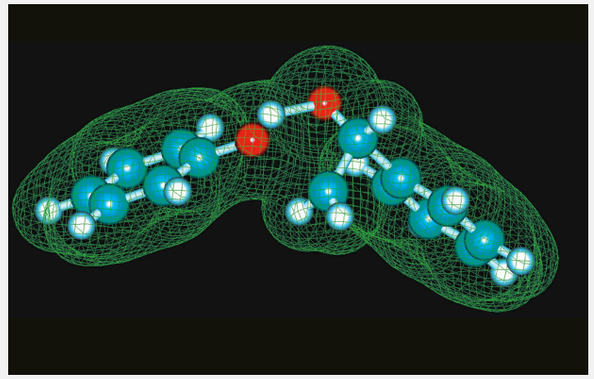


Why study algorithms?

They may unlock the secrets of life and of the universe.

> *" Computer models mirroring real life have become crucial for most advances made in chemistry today…. Today the computer is just as important a tool for chemists as the test tube. "*
>
> — *Royal Swedish Academy of Sciences*
>
> *(Nobel Prize in Chemistry 2013)*

Martin Karplus, Michael Levitt, and Arieh Warshel



Why study algorithms?

For fun and profit.

Google

Apple Computer

facebook

Cisco Systems

IBM

Nintendo

Jane Street

Morgan Stanley

NETFLIX

Adobe

RSA Security

D E Shaw & Co

ORACLE

Pandora

Akamai

YAHOO!

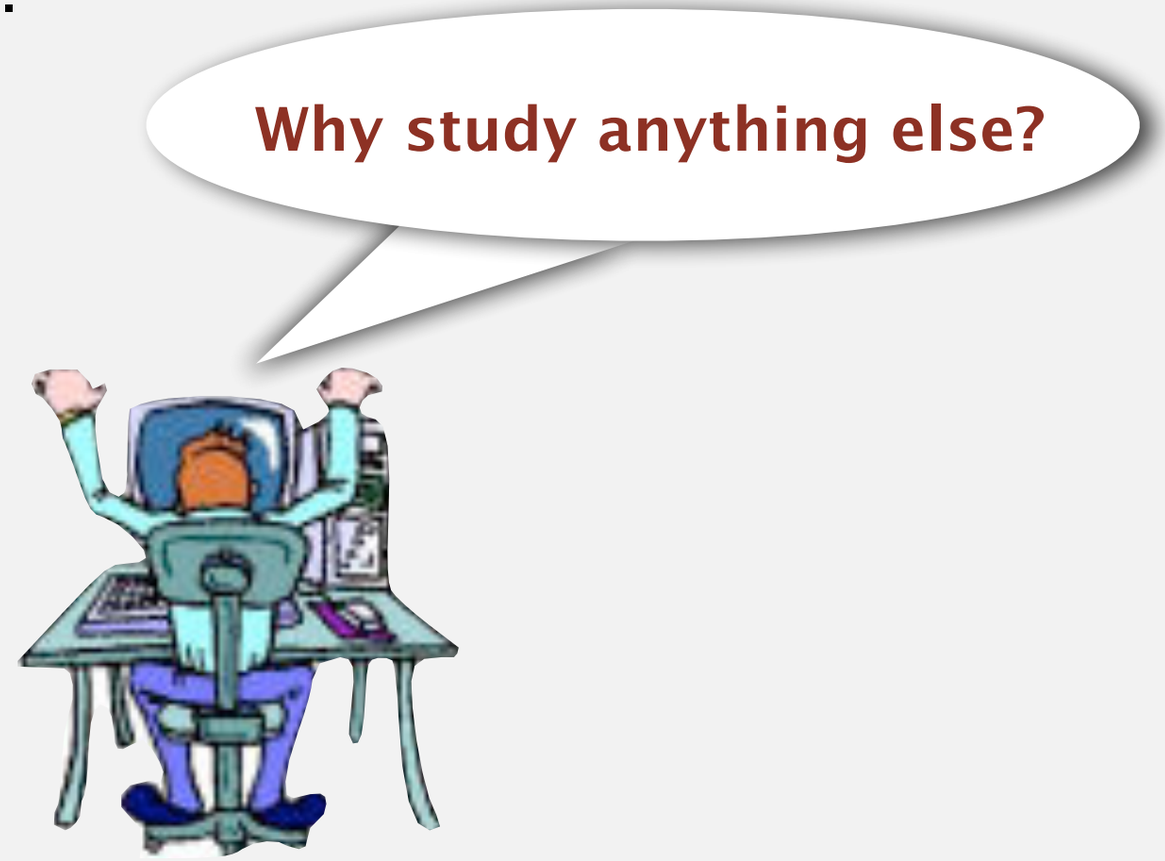
amazon.com

Microsoft

Pixar Animation Studios

Why study algorithms?

- Their impact is broad and far-reaching.
- Old roots, new opportunities.
- For intellectual stimulation.
- To become a proficient programmer.
- They may unlock the secrets of life and of the universe.
- To solve problems that could not otherwise be addressed.
- Everybody else is doing it.
- For fun and profit.



Coursework and grading

Reading material for most lectures

Weekly programming assignments

Four graded assignments. 60%

- Due on Tuesdays at 11pm via electronic submission (Github Classroom)
- Collaboration/lateness policies: see web.

Written exam. 40%

- Midterm practice exam 0%
- Final exam 40%

Honesty Statement

Honesty statement:

- Feel free to cooperate on assignments that are not graded.
- Assignments that are graded must be your own work. Do **not**:
  - Copy a program (in whole or in part).
  - Give your solution to a classmate (in whole or in part).
  - Get so much help that you cannot honestly call it your own work.
  - Receive or use outside help.
- Sign your work with the honesty statement (provided on the website).
- Above all: You are here for yourself, practice makes perfection.

Organisational issues

Presence:

- A presence sheet is circulated **purely** for statistics.
- Experience: those who do not attend lectures or do not make the assignments usually fail the course.
- Do not expect us to answer your questions if you were not at the lectures.

Office hours:

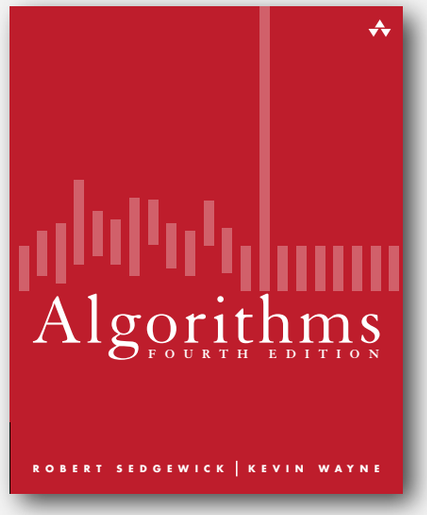
- Office hour: **Monday, 14:00-15:00**, please make an **appointment**!
- Please ask questions about the material presented in the lectures during the lectures — Everyone benefits
- We will discuss each assignment that is not graded during the next lab.

Registration:

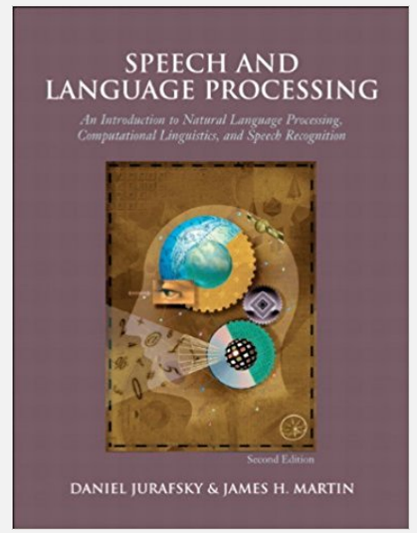
- Do the first assignment, A0.

Resources (textbook)

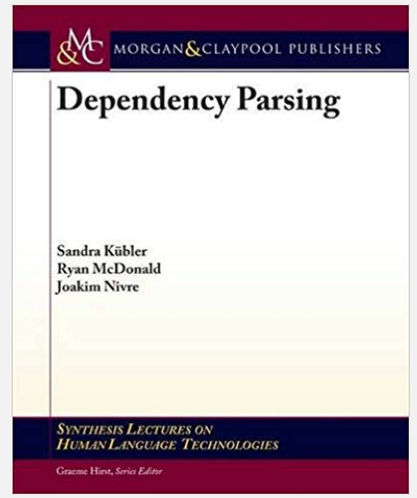
Required reading.



- Algorithms 4th edition by R. Sedgewick and K. Wayne, Addison-Wesley Professional, 2011, ISBN 0-321-57351-X.
  - Readable from university network thru Safari books:
  - see proquest.tech.safaribooksonline.de/9780132762571



- Speech and Language Processing, Jurafsky & Martin, 2nd Edition, Prentice Hall
  - Draft chapters of 3rd. edition available
  - see web.stanford.edu/~jurafsky/slp3/

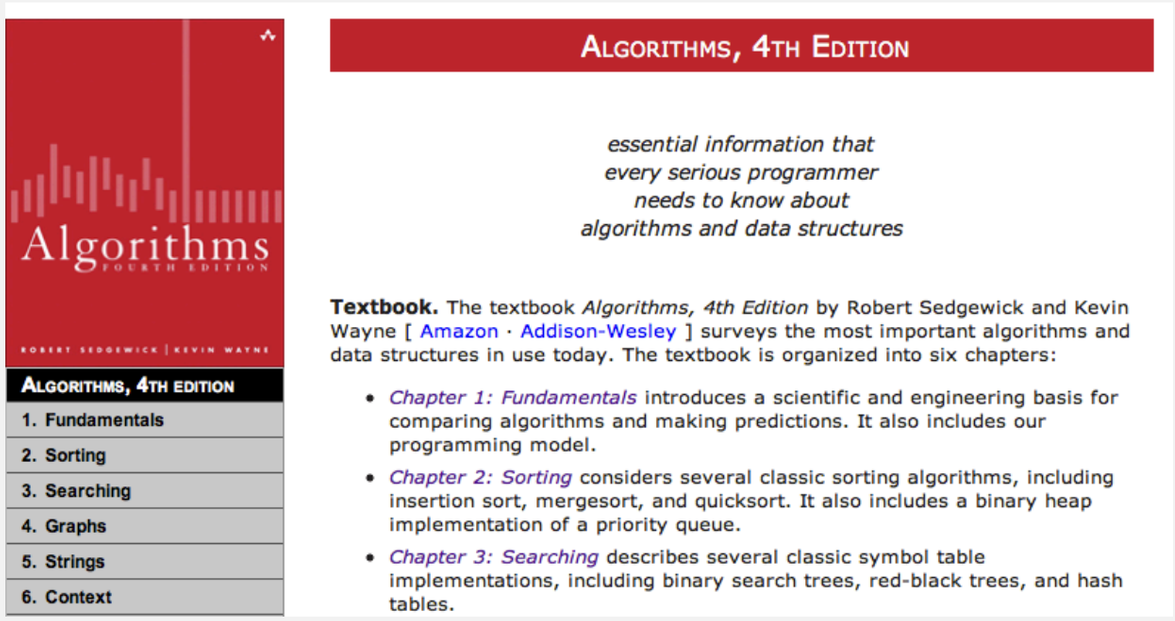


- Dependency Parsing, Kübler, McDonald & Nivre, Morgan & Claypool

Resources (web)

Book site for first part of class

- Brief summary of content.
- Download code from book.
- APIs and Javadoc.



http://algs4.cs.princeton.edu

What's ahead

Week	Monday (lectures)	Wednesday (lab)
01	Oct 22 A: *Introduction* B: *Complexity theory*	Oct 24 lab: *Language Guessing*
02	Oct 29 A: *Elemenary sorts* B: *Quicksort*	Nov 1 *No class*
03	Nov 05 A: *Undirected graphs* B: *Undirected graphs*	Nov 07 lab: *Sorting*
04	Nov 12 A: *Directed graphs* B: *Directed graphs*	Nov 14 lab: *Undirected graphs*
05	Nov 19 A: *Distance measures* B: *Binary heaps & heapsort*	Nov 21 lab: *Directed graphs*
06	Nov 26 A: *Binary heaps & heapsort* B: *Exam practice*	Nov 28 lab: *Burkhard-Keller trees*
07	Dec 03 A: *Formal languages and automata* B: *Formal languages and automata*	Dec 05 lab: *TBA*
08	Dec 10 A: *Regular grammars and finite state automata* B: *Regular grammars and finite state automata*	Dec 12 lab: *TBA*

What's Ahead

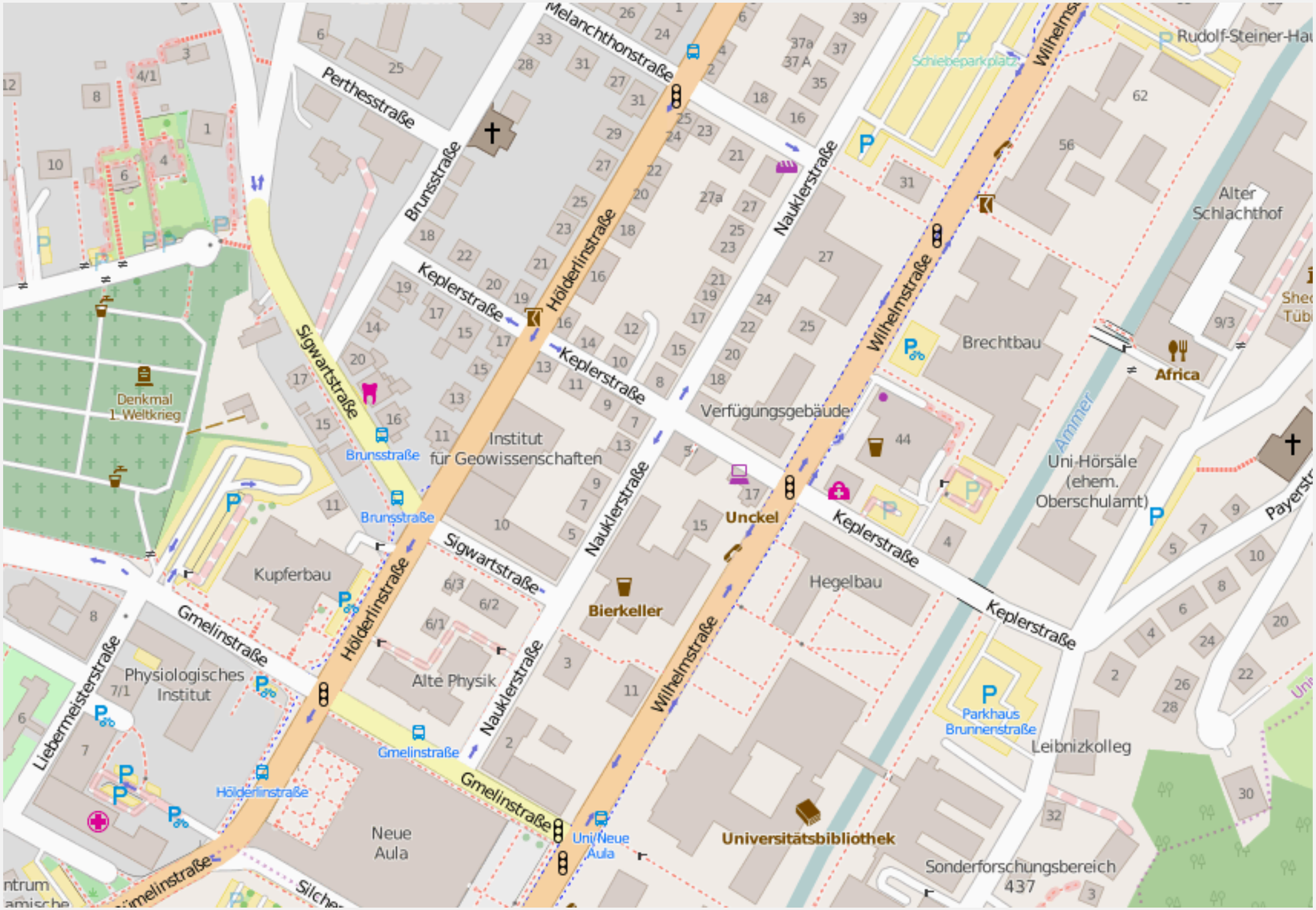
09	Dec 17 A: *Finite-state transducers* B: *Finite-state transducers and computational morphology*	Dec 19 lab: *TBA*
Sem. break	*No class*	*No class*
10	Jan 7 A: *Context-free languages and constituency parsing* B: *Context-free languages and constituency parsing*	Jan 9 lab: *TBA*
11	Jan 14 A: *Dependency grammars and treebanks* B: *Dependency grammars and treebanks*	Jan 16 lab: *TBA*
12	Jan 21 A: *Dependency parsing* B: *Dependency parsing*	Jan 23 lab: *TBA*
13	Jan 28 A: *A gentle introduction to classification* B: *Transition-based dependency parsing*	Jan 30 lab: *TBA*
14	Feb 04 A: *Exam review / practice* B: *Exam review / practice*	Feb 06 lab: *TBA*
15	Feb 11 A: *Exam* B:	Feb 13 lab: *TBA*

Sorting

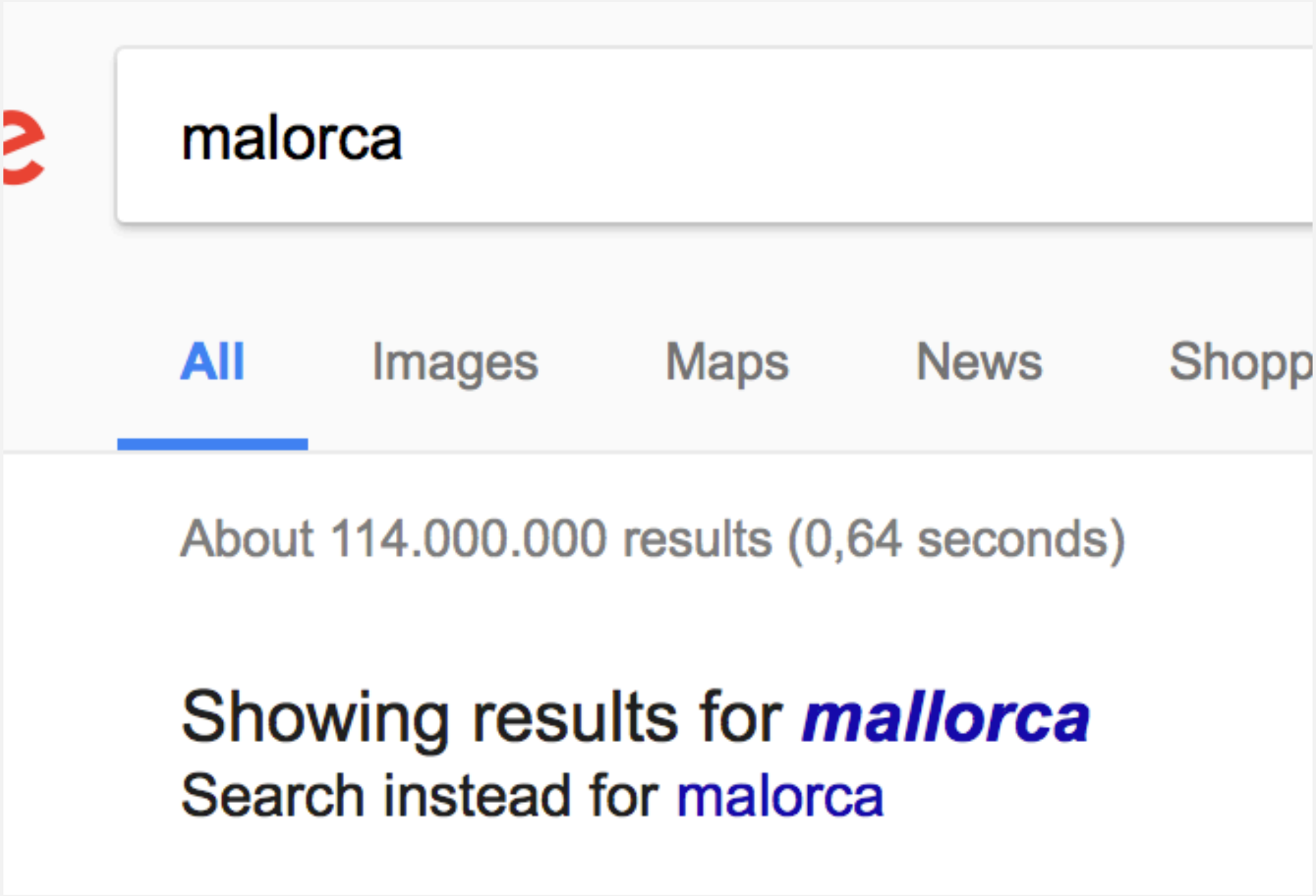
Bundesarchiv, Bild 183-22350-0001
Foto: Junge, Peter Heinz | 20. November 1953

Undirected Graphs

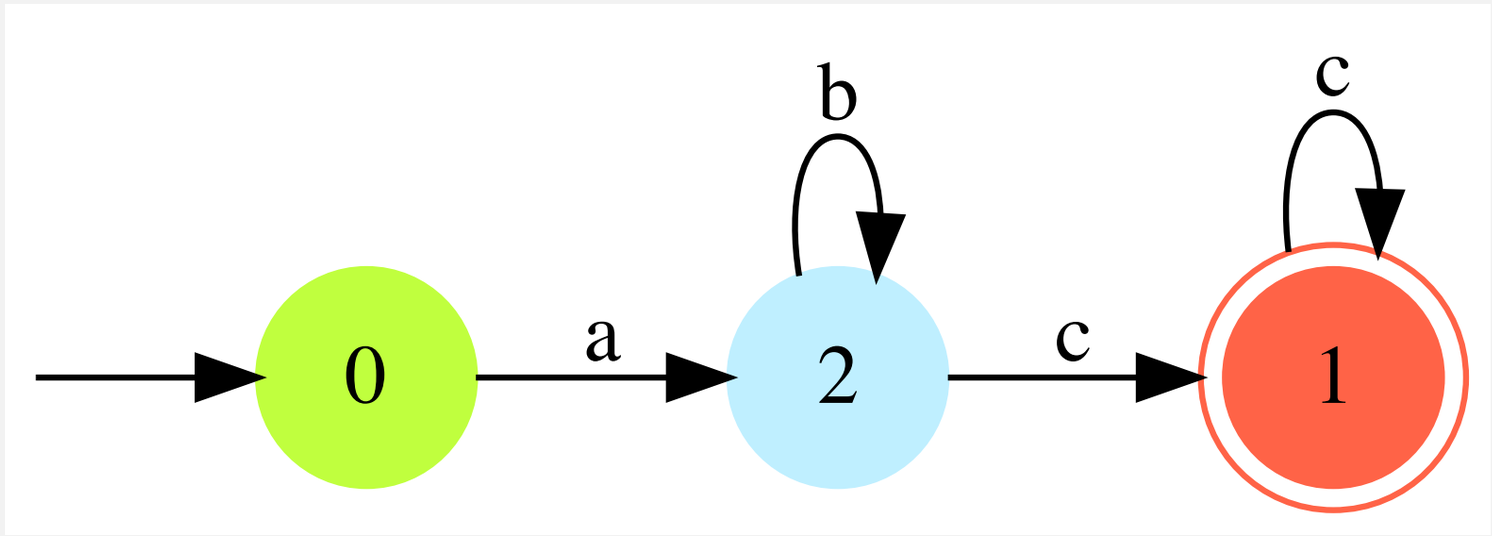
Directed Graphs

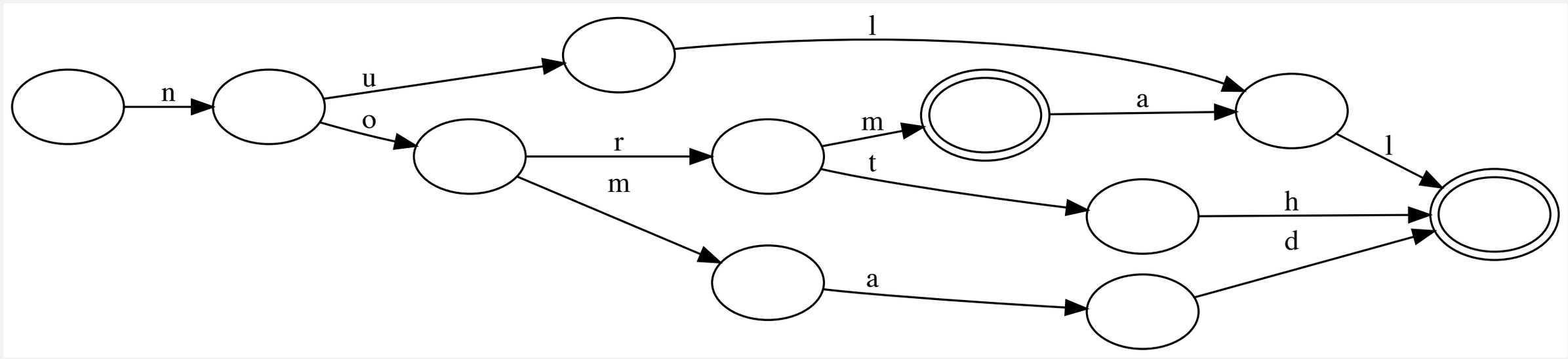


String Distance

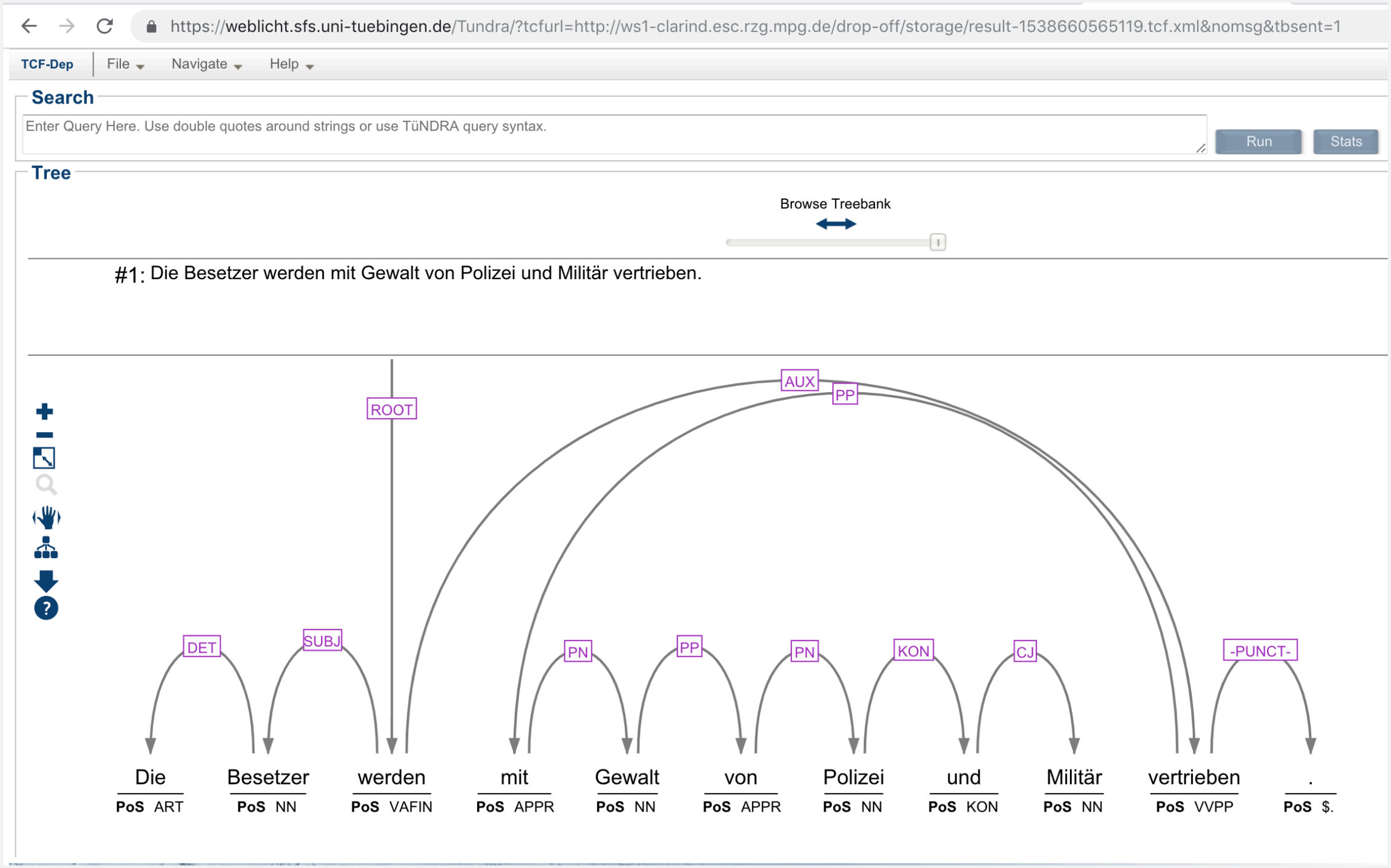


Finite State Automata





Parsing


https://weblicht.sfs.uni-tuebingen.de/Tundra/?tcfurl=http://ws1-clarind.esc.rzg.mpg.de/drop-off/storage/result-1538660565119.tcf.xml&nomsg&tbsent=1
TCF-Dep
File
Navigate
Help
Search
Enter Query Here. Use double quotes around strings or use TüNDRA query syntax.
Run
Stats
Tree
Browse Treebank
#1: Die Besetzer werden mit Gewalt von Polizei und Militär vertrieben.
ROOT
AUX
PP
DET
SUBJ
PN
PP
PN
KON
CJ
-PUNCT-
Die PoS ART
Besetzer PoS NN
werden PoS VAFIN
mit PoS APPR
Gewalt PoS NN
von PoS APPR
Polizei PoS NN
und PoS KON
Militär PoS NN
vertrieben PoS VVPP
. PoS $.