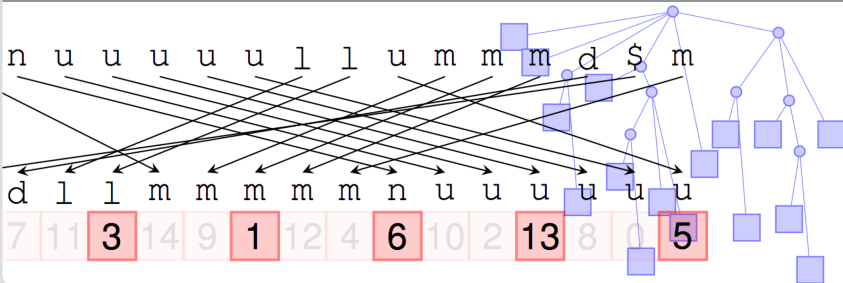


Text Indexing: Content

Simon Gog – gog@kit.edu

Institute of Theoretical Informatics - Algorithmics



Text Indexing

Motivation

Problem (String Matching)

Given a text T of length n , determine how many times a pattern P (of length m) occurs in T .

Sequential string match

E.g. use `fgrep` to search in a 200 MB XML file:

```
>time fgrep -c "<title>Indexing" dblp.xml.200MB  
176
```

```
real    0m5.156s  
user    0m5.113s  
sys     0m0.042s
```

Sequential string match

- Best sequential string matching algorithms (e.g. Knuth-Morris-Pratt) solve problem in $\mathcal{O}(n + m)$ time.
- For multi-pattern searches there exists also efficient algorithms (e.g. Aho-Corasick). Time complexity $\mathcal{O}(n + M)$, where M is the sum of the length of all patterns.
- Drawback: Time complexity depends linearly on text size n .

Text Indexing

Motivation

Indexed String Matching

Pre-process 200 MB XML file:

```
>./index-daemon -f dblp.xml.200MB &
```

Query the constructed index:

```
>time echo "<title>Indexing" | ./index-client  
176
```

```
real    0m0.006s  
user    0m0.002s  
sys     0m0.004s
```

0.006s vs. 5.156s of sequential solution.*

* Code available in the lecture branch of the SDSL repo.

Indexed String Matching

- Pre-compute data structure (*index*)
- Use index to answer the queries
- Time complexity still linearly dependent on m but not linearly dependent on n (or even independent).
- Another factor: alphabet size

Practical concerns

- Size of index (especially for in-memory indexes)
- Resource requirements for index construction

Beyond String Matching

Indexes can solve more complex queries efficiently:

- Find longest repeat (e.g. useful in LZ compression)
- Find suffix-prefix matches (e.g. useful in Bioinformatics)
- Ranked search (for a set of strings, i.e. a collection of documents)
 - Find documents which contain a patten most frequently
 - Find documents which contain a pattern and rank them according to their weight
 - ...

- Classical Text Indexes
 - Suffix Trees
 - String-B-Trees
- Compressed Text Indexes
 - Inverted Indexes
 - Compressed Suffix Arrays (based on Wavelet Trees,)
 - Compressed Suffix Trees
 - Lempel-Ziv Indexes
- Search Engines
 - based on Inverted Indexes
 - based on Self-Indexes

Compressed Text Indexes are composed of basic components which will also be covered in the lecture and are implemented in the SDSL library:

- (compressed) bitvectors
- rank and select data structures
- wavelet trees
- balanced parentheses structures
- range minimum query structures
- range search structures
- ...

The SDSL library is a C++ template library (design close to STL) which

- implement ideas from about 50 research papers
- makes it easy to compose complex index structures
- is well optimized (e.g. exploits bit-parallelism and SSE instructions)
- contains benchmarks and unit tests
- will be used in the practical part of the lecture
- is available at <https://github.com/simongog/sdsl-lite>

Tutorial slides are available on the github website.

- There will be 5-6 theoretical and practical projects.
- Please send me your solutions two days before the lecture in which the solution is presented
- You will get the additional ETC points if you solve over 70 percent of the tasks

- “Compact Data Structures” book by Navarro [1]
- Full-Text Index Survey article of Navarro and Mäkinen [2]
- and other recent articles

- [1] Gonzalo Navarro. *Compact Data Structures – A practical approach*. Cambridge University Press, 2016. ISBN 978-1-107-15238-0. 570 pages.
- [2] Gonzalo Navarro and Veli Mäkinen. Compressed full-text indexes. *ACM Comp. Surv.*, 39(1), 2007.

Text Indexing

Installation of sdsl-lite and running the shown example

```
git clone https://github.com/simongog/sdsl-lite.git
cd sdsl-lite
git checkout lecture
git submodule init
git submodule update
../install.sh ..
cd build
make index-daemon && make index-client
cd lecture
```

Indexing a text file and start a daemon to query the index:

```
./index-daemon -f <text_file>
```

Query the index:

```
time echo <Query> | ./index-client
```

Inputs are available at the Pizza&Chili webpage

<http://pizzachili.dcc.uchile.cl/texts.html>.