Compression with the \texttt{tudocomp} framework
The tudocomp framework

A C++ framework for engineering new text compression algorithms
tudocomp – Modularity

Input

Buffer
File
Stream

Compressor

Strategies
Coder(s)

Output

Buffer
File
Stream

Decompressor/Compressor

Input

Output
tudocomp – Statistics

Time and memory allocation measurement

- Execution partitioned into *phases*
- Logging to JSON dataset, chart visualization
Compressors

- LZ77
- LZ78
- LZW
- lcpcomp (new)
- BWT+MTF+RLE
- Re-Pair
- Longest-First*
- ESP*

Lempel-Ziv family

Integer Coders

- Elias $\gamma$ and $\delta$
- vByte
- Rice

Statistical Coders

- Huffman
- Arithmetic

* in development (theses)
Algorithm selection via the command-line

./tdc -a 'lzw(coder = ascii, lz78trie = ternary)' -g 'fib(4)' --usestdout

LZW compressor

encode factors as ASCII characters

store factors in ternary trie

input is 4th Fibonacci word

print factors to stdout
tudocomp – Benchmarking

- Automatic download of benchmark text collections (including Pizza & Chili corpus)
- Tool to compare multiple compressors on the same input

| Compressor   | C Time | C Memory | C Rate   | D Time | D Memory | chk |
|--------------|--------|----------|----------|--------|----------|-----|
| lcpcomp      | 103.1s | 3.2GiB   | 2.8505%  | 36.6s  | 7.6GiB   | OK  |
| lz77         | 98.5s  | 2.9GiB   | 4.0530%  | 4.3s   | 230.6MiB | OK  |
| bwt+mtf+rle  | 83.6s  | 1.7GiB   | 6.8688%  | 22.6s  | 1.4GiB   | OK  |
| huffman      | 2.7s   | 230.5MiB | 28.1072% | 5.9s   | 30.6MiB  | OK  |
| lzw          | 14.3s  | 480.9MiB | 23.4411% | 5.5s   | 452.6MiB | OK  |
| lz78         | 13.6s  | 480.8MiB | 29.1033% | 10.3s  | 142.9MiB | OK  |
| gzip -9      | 107.6s | 6.6MiB   | 26.2159% | 1.0s   | 6.6MiB   | OK  |
| bzip2 -9     | 13.8s  | 15.4MiB  | 25.2368% | 5.6s   | 11.7MiB  | OK  |
| lzma -9      | 138.6s | 691.7MiB | 1.9047%  | 337.3ms| 82.7MiB  | OK  |
String Generators
- Random (uniform) strings
- Thue-Morse sequences
- Fibonacci words

Succinct Data
- Bit-Compact integer vectors
- Bitwise I/O streaming

Text Data Structures
- Suffix array: divsufsort
- LCP array
  - Kasai / Φ algorithm
  - PLCP array
tudocomp – Website

http://tudocomp.org
Two showcases for tudocomp:

- LZ77 (without window)
- lcpcomp (new algorithm)
LZ77
LZ77

1 2 3 4 5 6 7 8 9

banananabar
LZ77
LZ77

bananabar

1 2 3 4 5 6 7 8 9
Starting from position 2, copy 3 symbols.
LZ77
Starting from position 1, copy 2 symbols.
LZ77

1 2 3 4 5 6 7 8 9
b a n (2,3) (1,2) r
lcpcomp
Starting from position 4, copy 3 symbols.
Starting from position 1, copy 2 symbols.
lcpcomp

1 2 3 4 5 6 7 8 9

b (4,3) na (1,2) r
Icpcomp – Decoding

1 2 3 4 5 6 7 8 9
b 4 5 6 n a 1 2 r
Icpcomp – Decoding
Icpcmp – Decoding

Position 4 not yet decoded → “Wait”
Icpcomp – Decoding

Position 4 not yet decoded → “Wait”
Position 5 is now decoded
→ “Tell everyone who is waiting for 5.”
Position 5 is now decoded
→ “Tell everyone who is waiting for 5.”
Position 6 is now decoded
→ “Tell everyone who is waiting for 6.”
Recursion:
Position 4 is now decoded
→ “Tell everyone who is waiting for 4.”
Positions 1 and 2 are already decoded → “Read”
Icpcomp – Example

1) Compute:
   • Suffix Array $SA$
   • LCP Array $LCP$

2) Put $i$ in an $LCP$-keyed max-heap if $LCP[i] > 1$
Icppcomp – Example

SA[i]  2 / 4
     3

SA[i-1]  7 / 1
     2

LCP[i]  3 / 5
     2

banananaabar
Extract maximum and introduce factor
Icpcomp – Example

Remove overlapped positions from heap

(4,2)
Icptcomp – Example

Extract maximum and introduce factor

7 / 1
2

(4, 2)
lcpcomp – Example

1 2 3 4 5 6 7 8 9

b (4,2) n a (1,2) r
Icpcomp – Practical Results

Icpcomp: pcr_cere (200 MiB, highly repetitive)
Lcpcmp – Comparison: LZ77

LZ77: pcr_cere (200 MiB, highly repetitive)
## lcpcomp – Evaluation

### pcr_cere (200 MiB, $\sigma = 6$, highly repetitive)

| Compressor     | C Time | C Memory | C Rate   | D Time | D Memory | chk  |
|----------------|--------|----------|----------|--------|----------|------|
| lcpcomp        | 103.1s | 3.2GiB   | 2.8505%  | 36.6s  | 7.6GiB   | OK   |
| lz77           | 98.5s  | 2.9GiB   | 4.0530%  | 4.3s   | 230.6MiB | OK   |
| bwt+mtf+rle    | 83.6s  | 1.7GiB   | 6.8688%  | 22.6s  | 1.4GiB   | OK   |
| huffman        | 2.7s   | 230.5MiB | 28.1072% | 5.9s   | 30.6MiB  | OK   |
| lzw            | 14.3s  | 480.9MiB | 23.4411% | 5.5s   | 452.6MiB | OK   |
| lz78           | 13.6s  | 480.8MiB | 29.1033% | 10.3s  | 142.9MiB | OK   |
| gzip -9        | 107.6s | 6.6MiB   | 26.2159% | 1.0s   | 6.6MiB   | OK   |
| bzip2 -9       | 13.8s  | 15.4MiB  | 25.2368% | 5.6s   | 11.7MiB  | OK   |
| lzma -9        | 138.6s | 691.7MiB | 1.9047%  | 337.3ms| 82.7MiB  | OK   |
Summary

**tudocomp**
- Highly modular C++14 framework
- Helpers
  - Benchmarks
  - Memory tracking
  - Data visualization
- Standard library for compression
  - Text data structures (SA, LCP)
  - Bit vectors
  - Bitwise I/O
- Classic compressors (baseline)
- Common coders

**lcpcomp**
- LZ77-based
- forward references allowed
- less factors than LZ77
- runs in $O(n)$ time (see paper)

**Outlook**
- better coders (like ANS, CABAC)
- grammar compressors
- external memory algorithms