

Storage and Retrieval of Individual Genomes

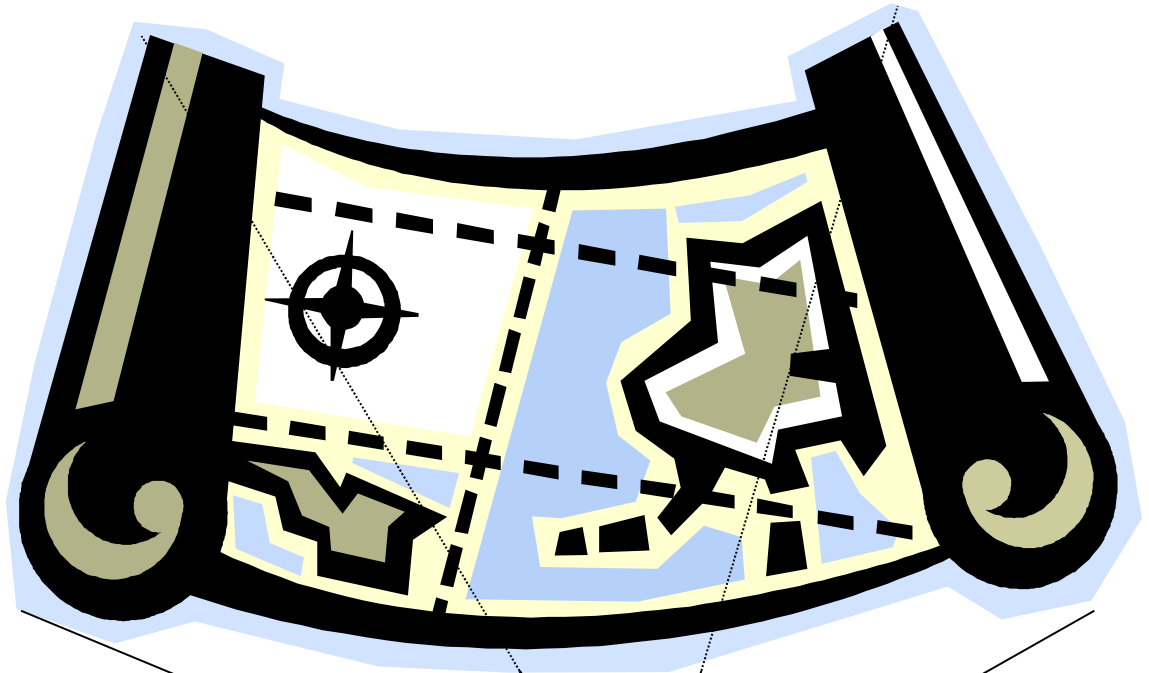
Veli Mäkinen

Department of Computer Science
University of Helsinki - Finland

*Joint work with J. Sirén, N. Välimäki,
G. Navarro, and J. Fischer.*

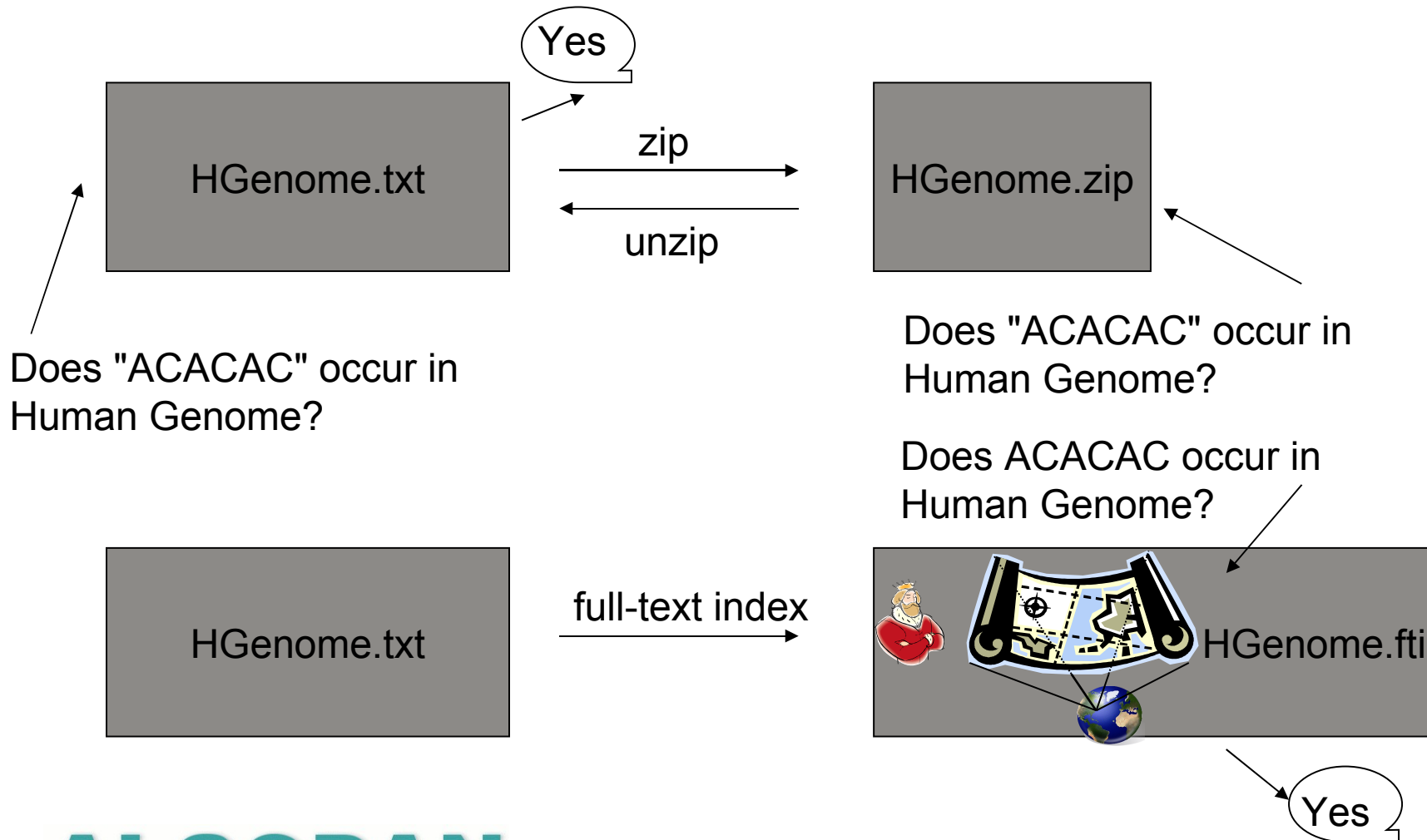
Mission:

Avoid maps bigger than the Empire!¹

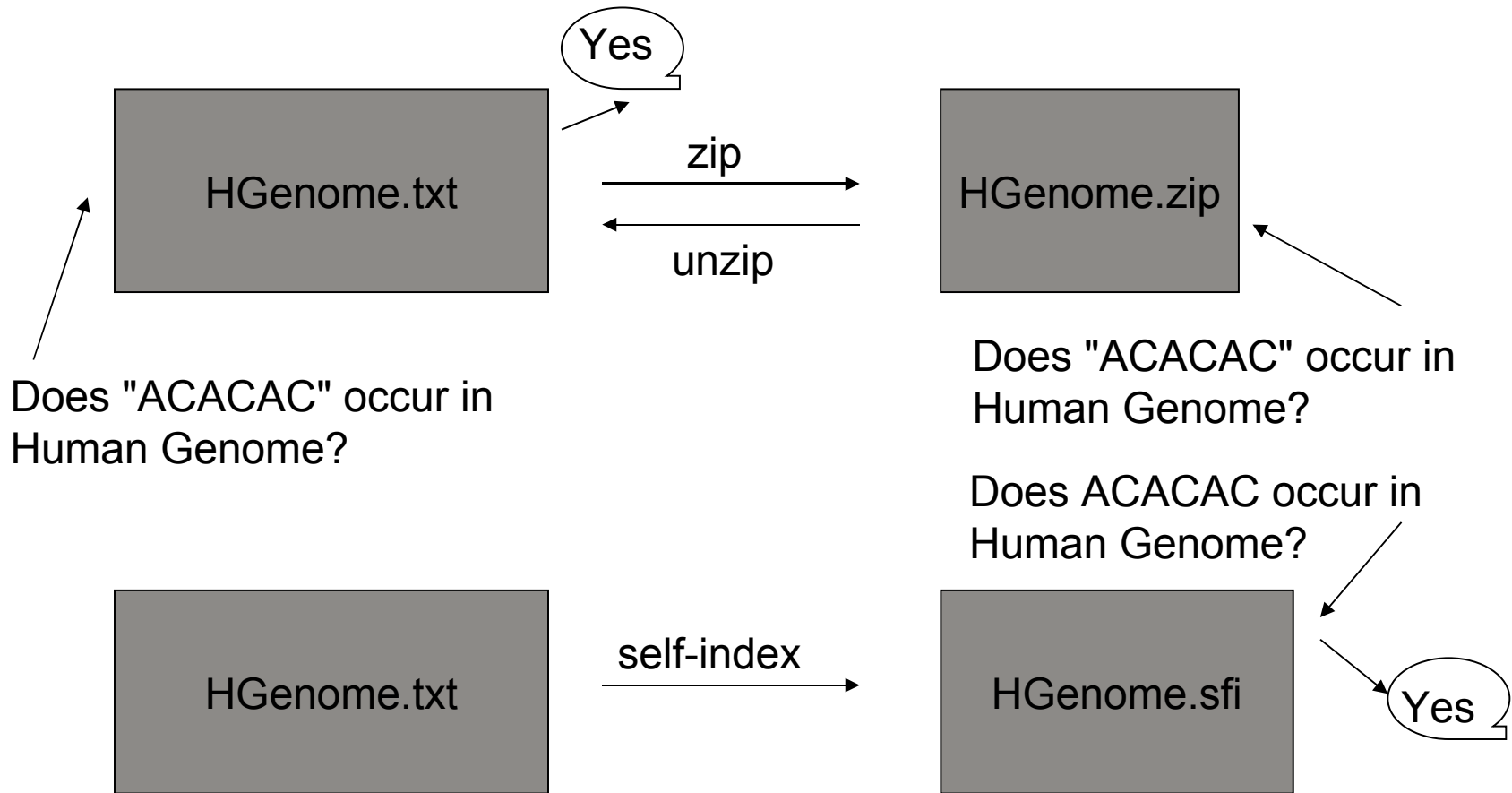


¹Apostolico, SPIRE 2001, invited talk

Example: Full-text indexing



Solution: Self-indexing



Introduction: Full-text indexing

- We consider exact string matching on static text $T[1,n]$.
- The task is to construct an index for T such that the occurrences of a given pattern can be found efficiently.
- Well-known optimal solution exists: build a *full-text index*, such as *suffix tree*, over the text.
- The full-text index -based solution on a text T takes $O(n \log n)$ bits of space.
- Text T itself can be represented in $n \log \sigma$ bits, where σ is the size of the alphabet
 - Or in even less space if text is compressible.



Applications: not just pattern search

- Full-text (self-)indexes can be used as building blocks to obtain data structures applicable for:
 - Text databases
 - Pattern discovery from sequences
 - Sequence analysis tasks in bioinformatics
 - Information retrieval on collections of text documents
 - Data mining on strings

Applications: space bottleneck

- In many applications the space usage is the real bottleneck, not the time efficiency.
- During the last 30 years, many practical / theoretical solutions with reduced space complexities have been proposed.
- The work can roughly be divided into three categories:
 - (1) Reducing constant factors
 - (2) Concrete optimization
 - (3) Abstract optimization

Reducing constant factors

- *Suffix arrays* (Manber & Myers 1990)
- *Suffix cactuses* (Kärkkäinen 1995)
- *Sparse suffix trees* (Kärkkäinen & Ukkonen 1996)
- *Space-efficient suffix trees* (Kurtz 1998)
- *Enhanced suffix arrays* (Abouelhoda & Ohlebusch & Kurtz 2002)

Concrete optimization

- “ \approx Minimizing automata”
- *DAWGS* (Blumer & Blumer & Haussler & McConnel & Ehrenfeucht 1983)
- *Compact DAWGS* (Crochemore & V erin 1997)
- *Compact suffix arrays* (M akinen 2000)

Abstract optimization

- **Objective:** Use as few space as possible to support the functionality of a given abstract definition of a data structure.
- Space is measured in bits and usually given proportional to the entropy of the text.

Abstract optimization: Example

- A *full-text index* for a given text T supports the following operations:
 - $\text{Exists}(P)$: is P a substring of T ?
 - $\text{Count}(P)$: how many times P occurs in T ?
 - $\text{Report}(P)$: list occurrences of P in T .

Abstract optimization...


- *Rank-select queries on bit-vectors* (Jacobson 1989)
- *Rank-select-type structures for suffix trees* (Munro & Raman & Rao & Clark 1996-)
- *Lempel-Ziv index* (Kärkkäinen & Ukkonen 1996)
- *Compressed suffix arrays* (Grossi & Vitter 2000, Sadakane 2000, 2002)
- *FM-index* (Ferragina & Manzini 2000)
- *High-order entropy compressed full-text indexes* (Grossi & Gupta & Vitter 2003, 2004)
- *Alphabet friendly FM-index* (Ferragina & Manzini & Mäkinen & Navarro 2004)

My talk in BAD 2007

■ *Implicit compression boosting*

- Appeared in *SPIRE 2007*, joint work with Gonzalo Navarro
- Features:
 - Optimal asymptotic compression (w.r.t. high-order entropy)
 - Optimal pattern search time on $\text{polylog}(|T|)$ -sized alphabets
 - Simplified design: Burrows-Wheeler transform + Wavelet tree = high-order compression
 - Extra space required during construction is sub-linear in the final structure size

New application: individual genomes





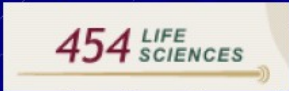


<http://genomics.xprize.org/genomics>

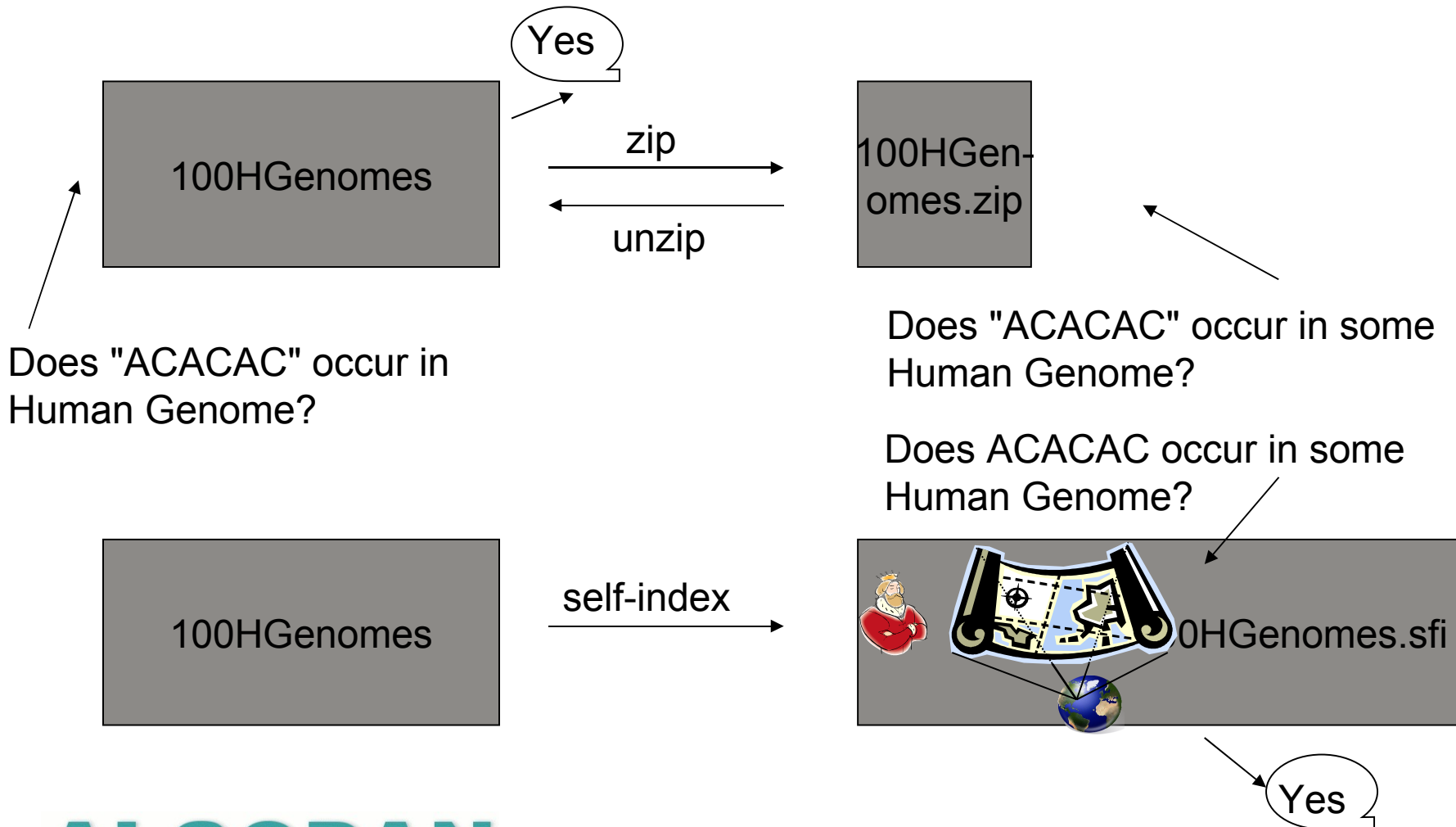
**\$10M to the First Team to Sequence
100 Human Genomes in 10 Days**

Registered Teams

- 454 Life Sciences (Roche) (www.454.com)
- VisiGen (www.visigenbio.com)
- FfAME (www.ffame.org)
- Reveo (www.reveo.com)
- Base4innovation (www.base4innovation.co.uk/)
- Personal Genome X-Team (PGx), George Church
Lars Paulin Institute of Biotechnology University of Helsinki



Individual Genomes



Some (hand waving) estimates...

Suffix tree of 100 Human Genomes

>6 Terabytes

Compressed suffix tree of 100 Human Genomes

~800
Gigabytes

LZ77 compressed file of 100 Human Genomes

~20
Giga
bytes

Repetitive Collection Indexing problem

- Let $C=\{T^1,T^2,\dots,T^r\}$ be a collection of texts such that
 - $|T^i|=n$ for all i , and
 - $\sum |T^i|=N$.
- Assume T^2,T^3,\dots,T^r are mutated copies of the base sequence T^1 , containing overall s mutations.
- Our goal is to devise a full-text index on C whose space requirement is of the form (measured in bits):

$$n \log \sigma + s \log \sigma + \log \binom{N-n}{s}$$

Our results (simplified)

- We can replace collection C with a self-index (named *CDSA*) that occupies in the expected case

$$O(n \log \sigma + s \log \sigma + s \log_{\sigma} N \log (N / (s \log_{\sigma} N)))$$

bits, and supports counting queries in $O(m \log N)$ time, where m is the length of the pattern P .

- We have another structure (named *RLWT*) that occupies little more and counts in $o(m \log N)$ time.

Our results (simplified)...

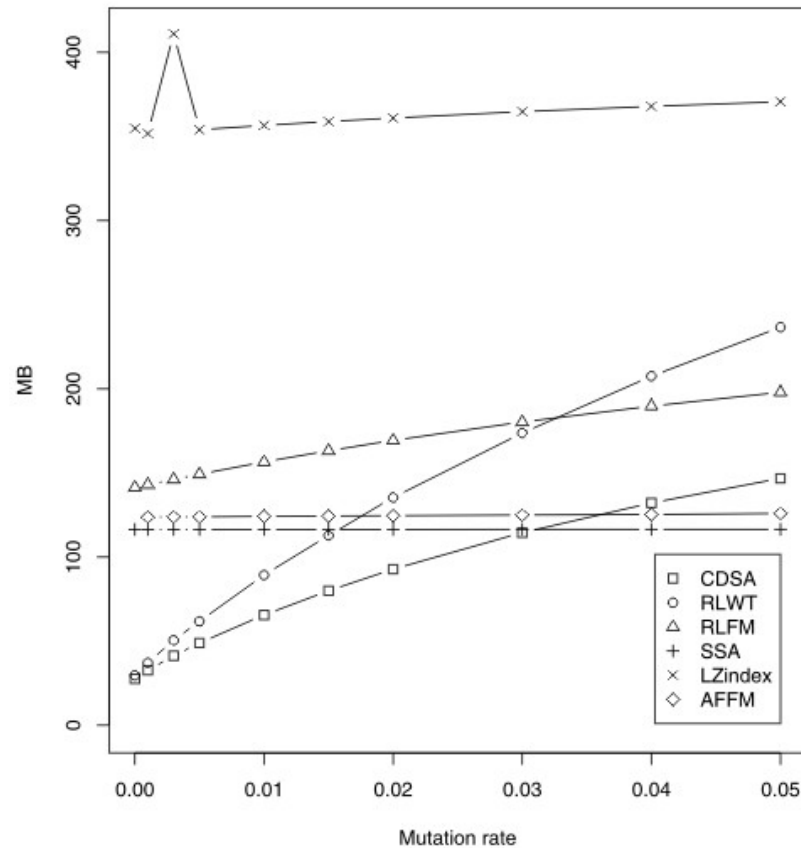
- We can replace the collection C with a *compressed generalized suffix tree* that occupies in the expected case

$$O(n \log \sigma + s \log \sigma + s \log_{\sigma} N \log (N / (s \log_{\sigma} N))) + o(N)$$

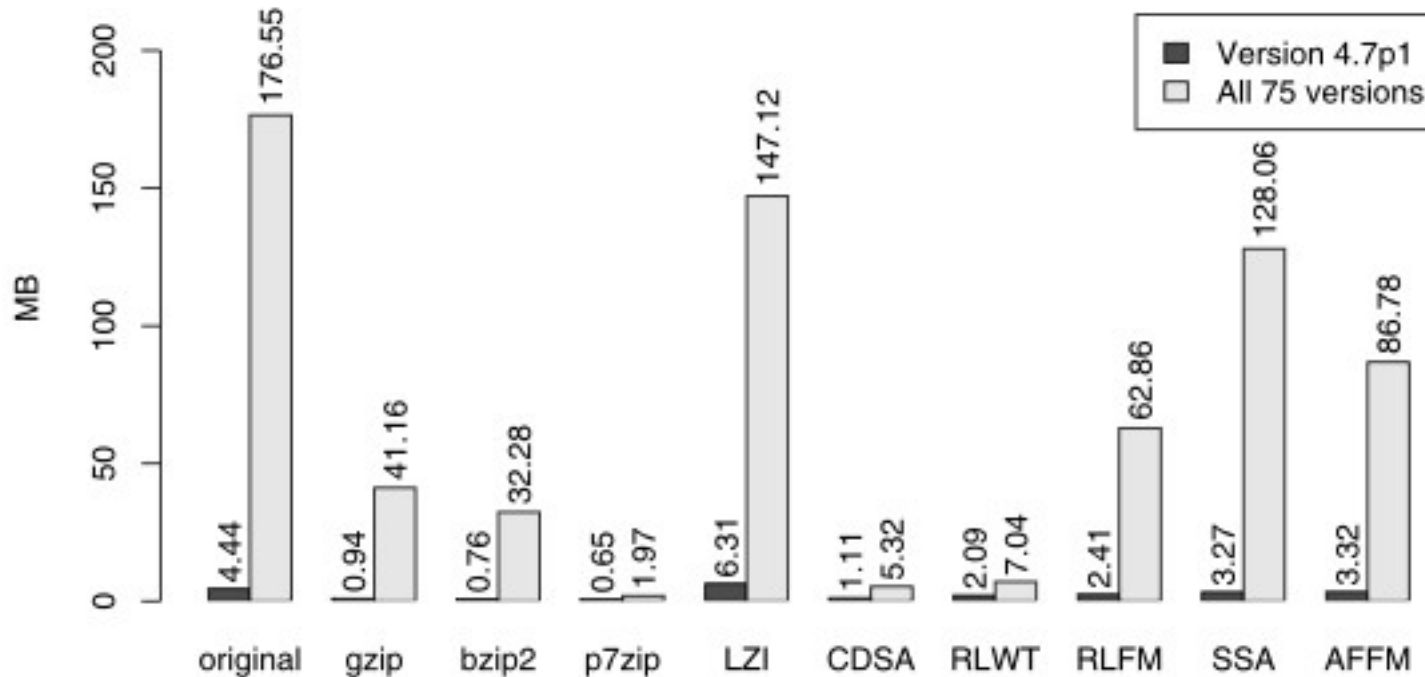
bits, and supports an extended¹ set of suffix tree operations in $O(\text{polylog}(N))$ time.

- 1) navigating the tree, (iterated) suffix links, edge labels, lowest common ancestors, text positions, string depth, tree depth, subtree size,..

In practice: 25 x 16 MB DNA



In practice: Version control data



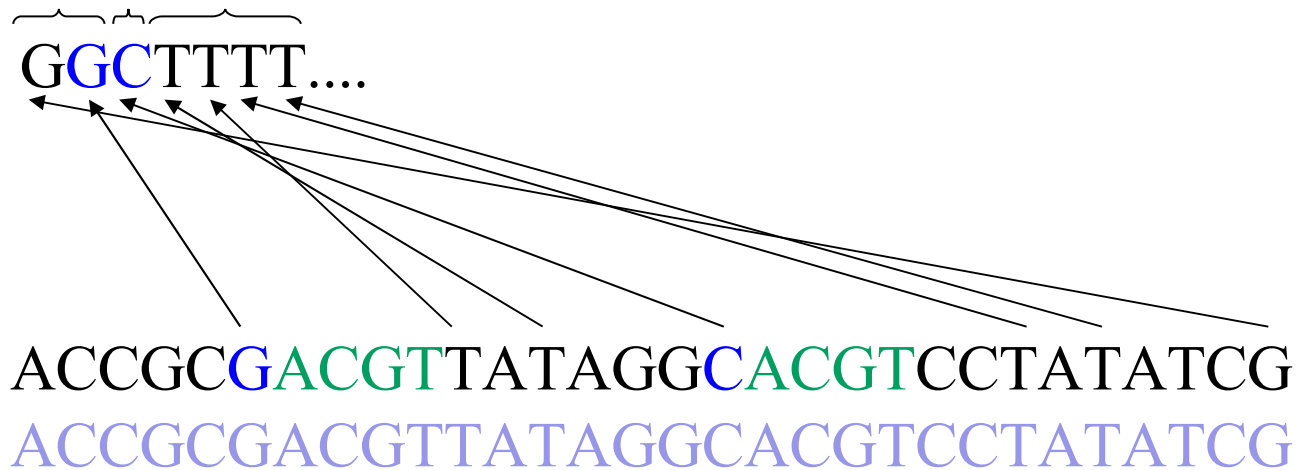
Why earlier self-indexes fail?

- So far self-indexes have been analyzed for the *k-th order entropy*, and some have been shown asymptotically optimal on any fixed $k < \log_{\sigma} n$.
- However, when the same sequence is repeated many times, the distribution of symbols in any k -symbol context does not change at!

ACCGCGACGTTATAGGCACGTCCTATATCG
ACCGCGACGTTATAGGCACGTCCTATATCG

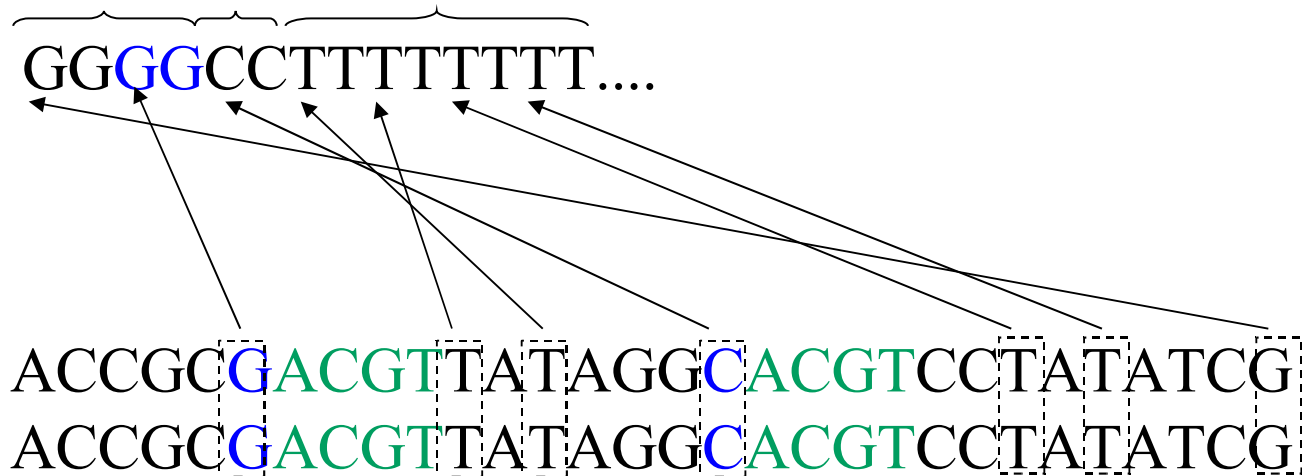
What is behind our results?

- Everything reduces to analyzing *runs in Burrows-Wheeler transform* (or equivalently *runs in Ψ*)!



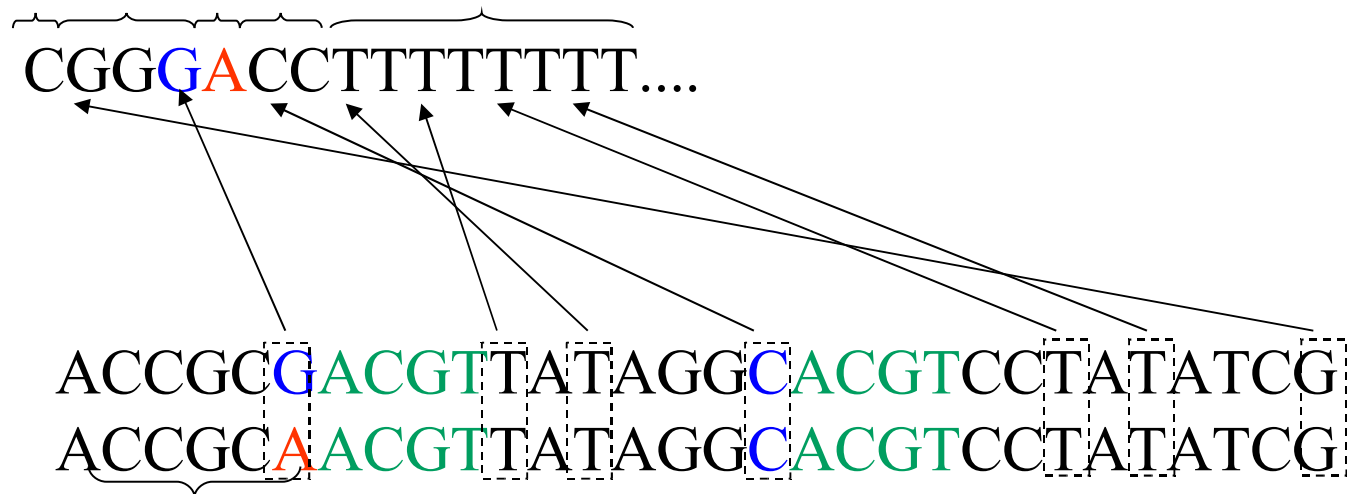
What is behind our results?

- The number of runs does not change when more copies of the same sequence are added:



What is behind our results?

- In the expected case the lexicographic order of a suffix is determined by its $O(\log_\sigma n)$ size context.



- *It follows that s mutations create $O(s \log_\sigma n)$ new runs in the expected case.*

Compressed disk suffix array (CDSA)

- Variant of Sadakane's compressed suffix array with a simple structure that makes it work fast on disk.
- Encodes the runs in Ψ (that is dual of the Burrows-Wheeler transform).
- Regular pointers to Ψ are stored to enable random access and backward search simulation.
- Requires $(R \log \sigma + R \log N/R)(1+o(1))$ bits of space, where R is the number of runs in Ψ .
- Using the pessimistic estimate $R=n+O(s \log_{\sigma} n)$ gives the result mentioned earlier for the repetitive collection indexing problem.

Run-Length Wavelet Tree (RLWT)

- Encodes the 0 and 1 runs in the *wavelet tree* of the Burrows-Wheeler transform by a reduction to the *data-aware dictionary* data structures by Gupta et al. (DCC'06).
- Occupies $R \log \sigma \log (2N/R)(1+o(1))$ bits, where R is the number of runs in the Burrows-Wheeler transform.
- Supports *rank* and *select* of symbols in $o(\log N)^1$ time, and hence the backward search in $o(m \log N)$ time, giving the result mentioned earlier.

1) complicated formula, bounded by $O(\log \sigma \log \log^2 N)$.

New compressed suffix tree

- Sadakane (*Theory of Computing Systems*, 2007) presented a compressed suffix tree (CST) that takes $|CSA| + 6N + o(N)$ bits, where $|CSA|$ is the size of the underlying compressed suffix array used.
- By adding $o(N)$ bits to CDSA or RLWT, we can simulate CSA in $O(\text{polylog}(N))$ time.
- By adding $o(N)$ bits to support so-called *previous/next smaller value queries*, we can remove $4N$ bits.
- The remaining $2N$ bits for *LCP values* can be replaced by an encoding of $2R \log(N/R) + o(N)$ bits.
- This gives the result mentioned earlier.

Future work

- Remove $o(N)$ bits from compressed suffix tree bound.
- Make the structures dynamic.
- Can one get closer to the lower-bound (also in the worst case)?

More details in...

- Veli Mäkinen, Jouni Sirén, and Niko Välimäki. *Storage and Retrieval of Individual Genomes and other Repetitive Sequence Collections*.
Technical report C-2008-1, Department of Computer Science, University of Helsinki, Finland, January 2008.
- Johannes Fischer, Veli Mäkinen, and Gonzalo Navarro. *An(other) Entropy-Compressed Suffix Tree*. Submitted.