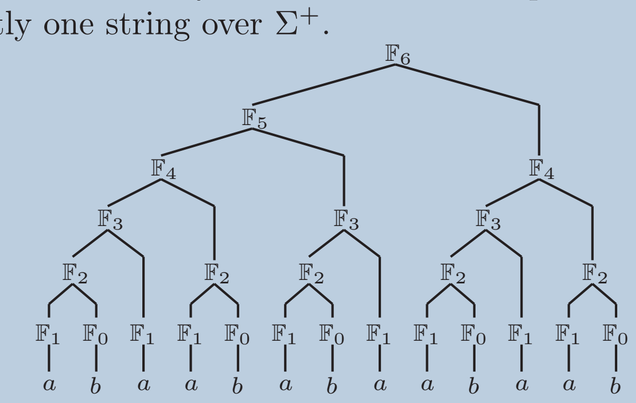
Straight-line Programs: A Short Overview

Eugene B. Kurpilyansky
dembelz@yandex.ru

Ural Federal University
named after the first President of Russia B.N.Yeltsin
Computer Science E-Days

Definitions

Straight-line program (SLP) is a context-free grammar in Chomsky normal form which produces exactly one string over Σ^+.



Main Facts

Theorem 1. *[2] There is no polynomial-time algorithm for the smallest grammar problem with approximation ratio less than* $\frac{8569}{8568}$ *unless* $P = NP$

Theorem 2. *[3] We can construct in* $O(n \log |\Sigma|)$ *time a* $O(\log n)$*-ratio approximation of a minimal grammar-based compression. Given LZ-factorization of length* k *we can construct a correponding grammar of size* $O(k \log n)$ *in time* $O(k \log n)$.

Complexity

There is a class of string problems that can be solved in terms of SLPs. Let $\mathbb{T}$ be an SLP that derives a text and let $\mathbb{P}$ be an SLP that derives a pattern.

Problem	Time
Pattern matching	$O(\lvert\mathbb{T}\rvert^2 \lvert\mathbb{P}\rvert)$
Counting all palindromes	$O(\lvert\mathbb{T}\rvert^4)$
Longest common substring	$O(\lvert\mathbb{T}\rvert^4 \log \lvert\mathbb{T}\rvert)$
Computing all overlaps	$O(\lvert\mathbb{T}\rvert^4 \log \lvert\mathbb{T}\rvert)$

There is a class of string problems that have no polynomial algorithm in terms of SLPs.

Problem	Complexity
Hamming distance	#P-complete
Embedding	θ_2^p-hard
Longest common subsequence	NP-hard

References

[1] *I. Burmistrov, A. Kozlova, E. Kurpilyansky, A. Khvorost*, Straight-line programs: a practical test (extended abstract), Journal of Mathematical Sciences, 192 (2013), 282–294.

[2] *M. Charikar, E. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, A. Shelat*, The smallest grammar problem, IEEE Trans. Information Theory, 51 (2005), 2554–2576.

[3] *W. Rytter*, Application of Lempel-Ziv factorization to the approximation of grammar-based compression, Theor. Comput. Sci., 302 (2003), 211–222.

A Prctical Test

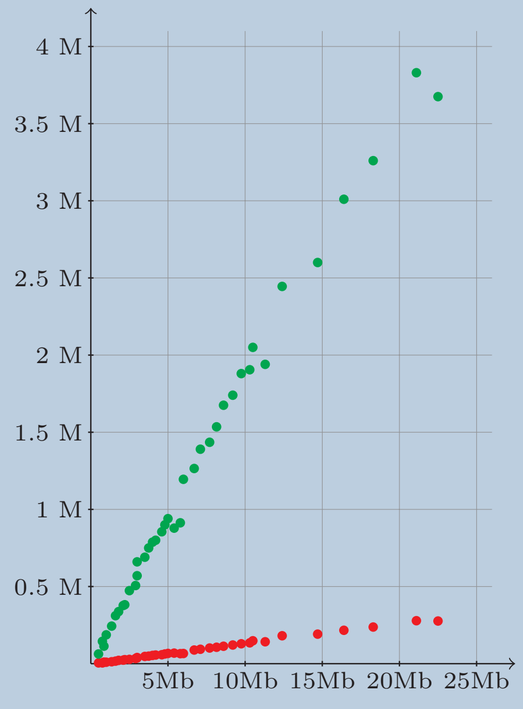
We are investigating different approaches that improve Rytter's algorithm:

- To minimize number of AVL-rotations using dynamic programming [1]
- To spend less time on balancing of nodes using Cartesian trees instead of AVL trees [1]
- To construct AVL trees in parallel.

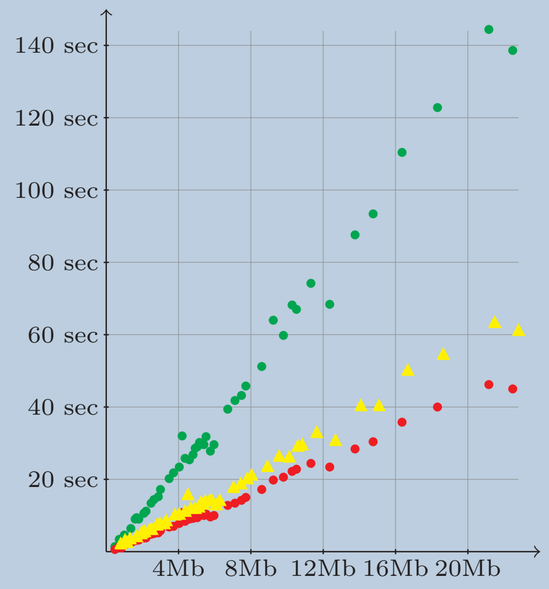
Algorithms notations:

- – Rytter's algorithm [3];
- – modified version of Ritter's algorithm [1];
- – cartesian SLP construction algorithm [1]
- – Lempel-Ziv algorithm with 32Kb search window (lz77);
- – Lempel-Ziv algorithm with infinite search window (lzma);

AVL rotations optimization statistics:



SLPs construction time statistics on DNAs:



Compressions ratio statistics on DNAs: