Journal of Computer and System Sciences 117 (2021) 1–22

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Block trees ☆

Djamal Belazzougui [a], Manuel Cáceres [b], Travis Gagie [c,*], Paweł Gawrychowski [d], Juha Kärkkäinen [e], Gonzalo Navarro [b], Alberto Ordóñez [f], Simon J. Puglisi [e], Yasuo Tabei [g]

[a] DTISI-CERIST, Algeria
[b] Center for Biotechnology and Bioengineering (CeBiB) and Department of Computer Science, University of Chile, Chile
[c] Faculty of Computer Science, Dalhousie University, Canada
[d] University of Wroclaw, Poland
[e] Helsinki Institute for Information Technology (HIIT) and Department of Computer Science, University of Helsinki, Finland
[f] Pinterest Inc., USA
[g] PRESTO, Japan Science and Technology Agency, Japan

A R T I C L E I N F O

Article history:
Received 17 September 2019
Received in revised form 11 May 2020
Accepted 5 November 2020
Available online 18 November 2020

Keywords:
Compressed data structures
Repetitive string collections
Lempel-Ziv compression

A B S T R A C T

Let string $S[1..n]$ be parsed into z phrases by the Lempel-Ziv algorithm. The corresponding compression algorithm encodes S in $\mathcal{O}(z)$ space, but it does not support random access to S. We introduce a data structure, the *block tree*, that represents S in $\mathcal{O}(z \log(n/z))$ space and extracts any symbol of S in time $\mathcal{O}(\log(n/z))$, among other space-time tradeoffs. The structure also supports other queries that are useful for building compressed data structures on top of S. Further, block trees can be built in linear time and in a scalable manner. Our experiments show that block trees offer relevant space-time tradeoffs compared to other compressed string representations for highly repetitive strings.

1. Introduction

Much of the fastest-growing data these days is highly repetitive: versioned document and software repositories like Wikipedia and GitHub store tens of versions of each document; whole-genome sequencing projects generate thousands of genomes of individuals of the same species; periodic astronomical surveys regularly scan the same portion of the sky. Such repetitiveness makes those large datasets highly compressible with dictionary methods like grammar-based or Lempel-Ziv compression, whereas typical statistical compression fails to capture the repetitiveness [29].

Lempel-Ziv compression [34] of a string $S[1..n]$ parses S into a sequence of z "phrases", where each phrase $S[i..j]$ is a new symbol (and $j = i$) or it appears leftwards in S. Lempel-Ziv compression takes $\mathcal{O}(n)$ time [44] and reduces S to $\mathcal{O}(z)$ space by encoding the phrases. While Lempel-Ziv is the practical method that best exploits repetitiveness, it has the problem that no way is known to access arbitrary substrings of S without decompressing it from the beginning.

All the previous work in the literature [8,10,3,4,24] resorts to grammar-based compression when it comes to provide direct access to compressed highly repetitive strings. Grammar-based compression [33] of S consists in generating a context-free grammar that generates S and only S. When S is repetitive, the size g of the grammar can be much smaller than n.

☆ Supported in part by Basal Funds FB0001 and FONDECYT Grant 1-200038, ANID, Chile, and by the Academy of Finland grants 258308 and 268324. An early partial version of this work appeared in *Proc. DCC'15* [5].

* Corresponding author.
E-mail address: travis.gagie@gmail.com (T. Gagie).

https://doi.org/10.1016/j.jcss.2020.11.002