Implementation of tiered vectors, a data structure for maintaining a sequence of $n$ elements supporting access in time $O(1)$ practical – and they also allow for simple optimal parallelization. Fast Dynamic Arrays. We present a highly optimized

nk + $o(nk)$ bits. Beyond Fenwick Trees, the results are primarily based on bit-packing and sampling – making them very

second result shows how optimal time for sum/update can be achieved while only slightly increasing the space usage to

nk + $o(n)$ bits of space while still supporting sum/update in $O(\log b n)$ / $O(\log\log b n)$ time where $2 \leq b \leq \log^{O(1)} n$. The

encoding model where one is allowed to reuse the space from the input data. Our main result is the first that only requires

Wepresent twosuccinct versions of the Fenwick Tree – which is known for its simplicity and practicality. Our results hold in the

subject to updates such that partial sums queries can be efficiently answered.

We consider the well-studied partial sums problem in succinct space where one is to maintain an array of $n$ $k$-bit integers


problems. Plugging in our new results we also immediately obtain new bounds for the string indexing for patterns with

uncompressed (uncompressed) source string. We present new data structures that achieve optimal time for updates and queries while

using space linear in the size of the optimal relative compression, for nearly all combinations of parameters. We also

present solutions for restricted and extended sets of updates. To achieve these results, we revisit the dynamic partial

sums problem and the substring concatenation problem. We present new optimal or near optimal bounds for these

problems. Plugging in our new results we also immediately obtain new bounds for the string indexing for patterns with


We consider the well-studied partial sums problem in succinct space where one is to maintain an array of $n$ $k$-bit integers subject to updates such that partial sums queries can be efficiently answered. We present twosuccinct versions of the Fenwick Tree – which is known for its simplicity and practicality. Our results hold in the

encoding model where one is allowed to reuse the space from the input data. Our main result is the first that only requires

nk + $o(n)$ bits of space while still supporting sum/update in $O(\log b n)$ / $O(\log\log b n)$ time where $2 \leq b \leq \log^{O(1)} n$. The

second result shows how optimal time for sum/update can be achieved while only slightly increasing the space usage to

nk + $o(nk)$ bits. Beyond Fenwick Trees, the results are primarily based on bit-packing and sampling – making them very practical – and they also allow for simple optimal parallelization. Fast Dynamic Arrays. We present a highly optimized implementation of tiered vectors, a data structure for maintaining a sequence of $n$ elements supporting access in time $O(1)$
and insertion and deletion in time $O(n)$ for $> 0$ while using $o(n)$ extra space. We consider several different implementation optimizations in C++ and compare their performance to that of vector and multiset from the standard library on sequences with up to 108 elements. Our fastest implementation uses much less space than multiset while providing speedups of $40\times$ for access operations compared to multiset and speedups of $10,000\times$ compared to vector for insertion and deletion operations while being competitive with both data structures for all other operations. Parallel Lookups in String Indexes. Here we consider the indexing problem on in the parallel random access machine model. Recently, the first PRAM algorithms were presented for looking up a pattern in a suffix tree. We improve the bounds, achieving optimal results for all parameters but the preprocessing. Given a text $T$ of length $n$ we create a data structure of size $O(n)$ that answers pattern matching queries for a pattern $P$ of length $m$ in $O(\log m)$ time and $O(m)$ work.

General information
State: Published
Organisations: Department of Applied Mathematics and Computer Science, Algorithms and Logic
Authors: Christiansen, A. R. (Intern), Bille, P. (Intern), Gørtz, I. L. (Intern)
Number of pages: 149
Publication date: 2018

Publication information
Publisher: DTU Compute
Original language: English

Series: DTU Compute PHD-2017
Volume: 464
ISSN: 0909-3192
Main Research Area: Technical/natural sciences
Electronic versions: phd464_Christiansen_AR.pdf

Relations
Projects:
Compressed and Practical Data Structures for Strings
Publication: Research › Ph.D. thesis – Annual report year: 2018

Dynamic Relative Compression, Dynamic Partial Sums, and Substring Concatenation
Given a static reference string $R$ and a source string $S$, a relative compression of $S$ with respect to $R$ is an encoding of $S$ as a sequence of references to substrings of $R$. Relative compression schemes are a classic model of compression and have recently proved very successful for compressing highly-repetitive massive data sets such as genomes and web-data. We initiate the study of relative compression in a dynamic setting where the compressed source string $S$ is subject to edit operations. The goal is to maintain the compressed representation compactly, while supporting edits and allowing efficient random access to the (uncompressed) source string. We present new data structures that achieve optimal time for updates and queries while using space linear in the size of the optimal relative compression, for nearly all combinations of parameters. We also present solutions for restricted and extended sets of updates. To achieve these results, we revisit the dynamic partial sums problem and the substring concatenation problem. We present new optimal or near optimal bounds for these problems. Plugging in our new results we also immediately obtain new bounds for the string indexing for patterns with wildcards problem and the dynamic text and static pattern matching problem.

General information
State: Published
Organisations: Department of Applied Mathematics and Computer Science, Algorithms and Logic
Number of pages: 18
Pages: 1-18
Publication date: 16 Oct 2017
Main Research Area: Technical/natural sciences

Publication information
Journal: Algorithmica
ISSN (Print): 0178-4617
Ratings:
BFI (2018): BFI-level 2
Web of Science (2018): Indexed yes
BFI (2017): BFI-level 2
Web of Science (2017): Indexed Yes
Fast Dynamic Arrays

We present a highly optimized implementation of tiered vectors, a data structure for maintaining a sequence of n elements supporting access in time $O(1)$ and insertion and deletion in time $O(n)$ for $n > 0$ while using $o(n)$ extra space. We consider several different implementation optimizations in C++ and compare their performance to that of vector and set from the standard library on sequences with up to $10^8$ elements. Our fastest implementation uses much less space than set while providing speedups of 40× for access operations compared to set and speedups of 10,000× compared to vector for insertion and deletion operations while being competitive with both data structures for all other operations.
Succinct partial sums and fenwick trees

We consider the well-studied partial sums problem in succinct space where one is to maintain an array of n k-bit integers subject to updates such that partial sums queries can be efficiently answered. We present two succinct versions of the Fenwick Tree which is known for its simplicity and practicality. Our results hold in the encoding model where one is allowed to reuse the space from the input data. Our main result is the first that only requires nk + o(n) bits of space while still supporting sum/update in O(logbn)/O(blogbn) time where 2 ≤ b ≤ log O(1)n. The second result shows how optimal time for sum/update can be achieved while only slightly increasing the space usage to nk + o(nk) bits. Beyond Fenwick Trees, the results are primarily based on bit-packing and sampling making them very practical and they also allow for simple optimal parallelization.

Finger Search in Grammar-Compressed Strings

Grammar-based compression, where one replaces a long string by a small context-free grammar that generates the string, is a simple and powerful paradigm that captures many popular compression schemes. Given a grammar, the random access problem is to compactly represent the grammar while supporting random access, that is, given a position in the original uncompressed string report the character at that position. In this paper we study the random access problem with the finger search property, that is, the time for a random access query should depend on the distance between a specified index f, called the finger, and the query index i. We consider both a static variant, where we first place a finger and subsequently access indices near the finger efficiently, and a dynamic variant where also moving the finger such that the time depends on the distance moved is supported.

Let n be the size the grammar, and let N be the size of the string. For the static variant we give a linear space representation that supports placing the finger in O(log(N)) time and subsequently accessing in O(log(D)) time, where D is the distance between the finger and the accessed index. For the dynamic variant we give a linear space representation that supports placing the finger in O(log(N)) time and subsequently accessing and moving the finger in O(log(D) + log(log(N))) time. Compared to the best linear space solution to random access, we improve a O(\log(N)) query bound to O(\log(D)) for the static variant and to O(\log(D) + \log(\log(N))) for the dynamic variant, while maintaining linear space. As an application of our results we obtain an improved solution to the longest common extension problem in grammar compressed strings. To
obtain our results, we introduce several new techniques of independent interest, including a novel van Emde Boas style decomposition of grammars.

Parallel Lookups in String Indexes
Recently, the first PRAM algorithms were presented for looking up a pattern in a suffix tree. We improve the bounds, achieving optimal results.
Projects:

**Algorithms for Compressed Computation**

Technical University of Denmark  
Period: 01/11/2014 → 13/03/2018  
Number of participants: 6  
Phd Student: Christiansen, Anders Roy (Intern)  
Supervisor: Gørtz, Inge Li (Intern)  
Main Supervisor: Bille, Philip (Intern)  
Examiner: Witt, Carsten (Intern)  
Grossi, Roberto (Ekstern)  
Puglisi, Simon J. (Ekstern)

**Financing sources**  
Source: Internal funding (public)  
Name of research programme: Samfinansieret - Andet

**Relations**  
Publications: Compressed and Practical Data Structures for Strings  
Project: PhD