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.
Deterministic indexing for packed strings

Given a string $S$ of length $n$, the classic string indexing problem is to preprocess $S$ into a compact data structure that supports efficient subsequent pattern queries. In the deterministic variant the goal is to solve the string indexing problem without any randomization (at preprocessing time or query time). In the packed variant the strings are stored with several character in a single word, giving us the opportunity to read multiple characters simultaneously. Our main result is a new string index in the deterministic and packed setting. Given a packed string $S$ of length $n$ over an alphabet $\sigma$, we show how to preprocess $S$ in $O(n)$ (deterministic) time and space $O(n)$ such that given a packed pattern string of length $m$ we can support queries in (deterministic) time $O(m/\alpha + \log m + \log \log \sigma)$, where $\alpha = w/\log \sigma$ is the number of characters packed in a word of size $w = 8(\log n)$. Our query time is always at least as good as the previous best known bounds and whenever several characters are packed in a word, i.e., $\log \sigma$.
Matching and Compression of Strings with Automata and Word Packing

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 wildcard problem and the dynamic text and static pattern matching problem. Subsequence Automata with Default Transitions

Given a packed string S of length n with characters from an alphabet of size σ. The subsequence automaton of S (often called the directed acyclic subsequence graph) is the minimal deterministic finite automaton accepting all subsequences of S. A straightforward construction shows that the size (number of states and transitions) of the subsequence automaton is $O(n)$ and that this bound is asymptotically optimal. In this paper, we consider subsequence automata with default transitions, that is, special transitions to be taken only if none of the regular transitions match the current character, and which do not consume the current character. We show that with default transitions, much smaller subsequence automata are possible, and provide a full trade-off between the size of the automaton and the delay, i.e., the maximum number of consecutive default transitions followed before consuming a character. Specifically, given any integer parameter $k$, $1 < k$, we present a subsequence automaton with default transitions of size $O(n k \log k)$ and delay $O(\log k)$. Hence, with $k = 2$ we obtain an automaton of size $O(n \log)$ and delay $O(\log)$. At the other extreme, with $k = m$, we obtain an automaton of size $O(n)$ and delay $O(1)$, thus matching the bound for the standard subsequence automaton construction. Finally, we generalize the result to multiple strings. The key component of our result is a novel hierarchical automata construction of independent interest. Deterministic Indexing for Packed Strings

Given a string S of length n, the classic string indexing problem is to preprocess S into a compact data structure that supports efficient subsequent pattern queries. In the deterministic variant the goal is to solve the string indexing problem without any randomization (at preprocessing time or query time). In the packed variant the strings are stored with several character in a single word, giving us the opportunity to read multiple characters simultaneously. Our main result is a new string index in the deterministic and packed setting. Given a packed string S of length n over an alphabet $\sigma$, we show how to preprocess S in $O(n)$ (deterministic) time and space $O(n)$ such that given a packed pattern string of length m we can support queries in (deterministic) time $O(m + \log m + \log \log n)$, where $w = \log n$ is the number of characters packed in a word of size w = $\log n$. Our query time is always at least as good as the previous best known bounds and whenever several characters are packed in a word, i.e., $\log n$, the query times are faster. Dynamic Partial Sums in Constant Time and Succinct Space

In the ultra wide word-RAM Model the dynamic partial sums problem is to dynamically maintain an array of n integers while supporting efficient access, update and partial sums queries. This classic problem, and its variations, are very well studied in many different computational models [Fre82,FS89,Fen94, HSS11, HR03, HRS96,RRR01, PD04]. We solve the partial sums problem in the ultra wide word-RAM model, recently introduced by Farzan et al. [FLONS15], where we, in constant time, are allowed to manipulate words of size $w^2$ and access w memory locations. Farzan et al. [FLONS15] additionally gave a solution to the dynamic partial sums problem by simulating the RAMBO model to obtain a result by Brodnik et al. [BKMN06]. In this paper we present an improved solution to the dynamic partial sums problem in the ultra wide word-RAM model that supports all operations in either constant or $O(\log \log n)$ time, depending on whether we allow multiplication, and succinct space. We pose as an open problem whether it is possible in the ultra wide word-RAM model to additionally support the classic select operation in constant time.
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 \( O(\log bn) \) 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.

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.
Subsequence Automata with Default Transitions

Let $S$ be a string of length $n$ with characters from an alphabet of size $\sigma$. The \textit{subsequence automaton} of $S$ (often called the \textit{directed acyclic subsequence graph}) is the minimal deterministic finite automaton accepting all subsequences of $S$. A straightforward construction shows that the size (number of states and transitions) of the subsequence automaton is $O(n\sigma)$ and that this bound is asymptotically optimal. In this paper, we consider subsequence automata with\textit{ default transitions}, that is, special transitions to be taken only if none of the regular transitions match the current character, and which do not consume the current character. We show that with default transitions, much smaller subsequence automata are possible, and provide a full trade-off between the size of the automaton and the \textit{delay}, i.e., the maximum number of consecutive default transitions followed before consuming a character. Specifically, given any integer parameter $k$, $1 < k \leq \sigma$, we present a subsequence automaton with default transitions of size $O(nk\log k\sigma)$ and delay $O(\log k\sigma)$. Hence, with $k = 2$ we obtain an automaton of size $O(n \log \sigma)$ and delay $O(\log \sigma)$. On the other extreme, with $k = \sigma$, we obtain an automaton of size $O(n\sigma)$ and delay $O(1)$, thus matching the bound for the standard subsequence automaton construction. The key component of our result is a novel hierarchical automata construction of independent interest.
Projects:

Matching and Compression of Strings with Automata and Word Packing
Technical University of Denmark
Period: 01/03/2014 → 14/06/2017
Number of participants: 7
Phd Student:
Skjoldjensen, Frederik Rye (Intern)
Supervisor:
Gørtz, Inge Li (Intern)
Thomassen, Carsten (Intern)
Main Supervisor:
Bille, Philip (Intern)
Examiner:
Witt, Carsten (Intern)
Landau, Gad M. (Ekstern)
Pagh, Rasmus (Ekstern)

Financing sources
Source: Internal funding (public)
Name of research programme: Forskningsrådsfinansiering

Relations
Publications:
Matching and Compression of Strings with Automata and Word Packing
Project: PhD