Building and In-Place Checking Suffix Array in External Memory

Yi Wu, Ge Nong, Wai Hong Chan, and Bin Lao

Abstract—The induced sorting (IS) method has been used to design disk-based algorithms for suffix array (SA) construction. A recent engineering of these algorithms achieved nearly optimal disk space, with both the I/O and the time complexities better than the current fastest external-memory suffix sorters. In this paper, we propose a checking algorithm that enables any IS suffix sorting algorithms to check an SA when it is being constructed. Our experimental results indicate that the time, space and I/O volume for verification by our checking program is negiligible in comparison with that for construction by our
Recently, these algorithms have been carefully engineered to achieve nearly optimal disk space, with both the I/O and the time complexities better than the current fastest external-memory suffix sorters. In this paper, we propose a checking algorithm that enables any IS suffix sorting algorithm to build and check an SA simultaneously. Our experimental results indicate that, in comparison with that for building, the time, space and I/O consumptions for checking is negligible. This convinence us that the proposed checking algorithm can be combined with the existing IS suffix sorting algorithms to develop efficient solutions for the situations where checking is a must for a constructed SA.

Index Terms—Suffix array, in-place verification, external memory.

1 Introduction

On RAM models, the suffix array for any text can be built within linear time and space using the SA-IS algorithm [1]. Recently, this algorithm has been successfully adapted to design three external-memory suffix sorting algorithms eSAIS [2], DSA-IS [3] and SAIS-PQ [4].To avoid random I/O accesses, these algorithms adopt different approaches to retrieve the head characters of unsorted suffixes for inducing their lexical order. However, they all suffer from a disk space bottleneck and

Although these algorithms can achieve

These algorithms use different approaches to retrieve the head characters of suffixes to be sorted when when inducing their lexical order from the sorted ones. Although the experimental

This algorithm sort suffixes using the induced sorting method, which sorts two suffixes by comparing their heading characters and the lexical order of the ones starting at the next positions.

the induced sorting principle. The main idea behind the IS method is to

the main idea of which is to induce the lexicographical order of all the suffixes from that of a sorted subset. Recently, the IS method has been successfully applied to designing three suffix sorting algorithms eSAIS [2], SAIS-PQ [4] and DSA-IS [3] for external memory models. These disk-based variants commonly use a priority queue to simulate SA-IS, but they differ from each other in the way of retrieving the preceding character of a sorted suffix when inserting it

when inserting the unsorted ones into the SA.

several works successfully applied the IS method to designing efficient SA construction algorithms on external memory models, where the biggest difference between them is the way of retrieving the preceding characters of sorted suffixes for inducing the order of unsorted ones.

The internal-memory IS algorithms for sorting suffixes can build an SA in linear time and space [1]. Recently, several works successfully applied the IS method to construct suffix arrays for massive datasets using external memory.

For an given string, its suffix array can be built in linear time and space using the induced sorting method.

Suffix array can be

The algorithm SA-IS is currently the fastest SA construction algorithm

SA-IS is currently the fastest

The suffix array (SA) [**?**] can be built in linear time and space using the induced sorting method [1]. Currently, there are three works

The suffix array (SA) [**?**] can be built in linear time and space

These algorithms have been implemented for demonstration and experiment purposes,

The suffix array (SA) [**?**] is an essential data structure for string processing and information retrieval. Among the existing internal-memory algorithms for SA construction, SA-IS [1] achieves the optimal time and space complexities using the induced sorting principle, where the key operation is to retrieve the preceding characters of sorted suffixes in order for inducing the lexical order of unsorted ones. To handle massive datasets , several external memory algorithms have been proposed for building massive suffix arrays in recent years, e.g., DC3 [**?**], bwt-disk [**?**],

- *Y. Wu, G. Nong (corresponding author) and B. Lao are with the Department of Computer Science, Sun Yat-sen University, Guangzhou 510275, China. E-mails: wu.yi.christian@gmail.com, issng@mail.sysu.edu.cn, Laobin@mail3.sysu.edu.cn.*
- *Wai Hong Chan (corresponding author) is with the Department of Mathematics and Information Technology, The Education University of Hong Kong, Hong Kong. E-mail: waihchan@ied.edu.hk.*

SAscan [5], pSAscan [6], eSAIS [**?**], EM-SA-DS [**?**] and DSA-IS [**?**]. Among them, the latter three algorithms are based on the induced sorting (IS) method described in SA-IS [**?**].

This algorithm employs the IS method to induce the lexical order of all the suffixes from a selected subset.

This algorithm uses the induced sorting method to

In the past decades, great effort has been taken to study efficient algorithms for SA construction. th

extensive works have been put on designing time and space efficient suffix sorting algorithms

data structure that has been widely used in many string processing applications, e.g., biological sequence alignment, time series analysis and text clustering. Given an input string, traversing its suffix tree can be emulated by using the corresponding enhanced suffix array [**?**], which mainly consists of the suffix and the longest common prefix arrays. It has been realized that the application scope of an index mainly depends on the construction speed and the space requirement. This leads to intensive works on designing time and space efficient suffix sorting algorithms over the past decade, assuming different computation models such as internal memory, external memory, parallel and distributed models. Particularly,

The basic idea behind the induced sorting method is to induce the lexicographical order of all the substrings/suffixes from a sorted subset of substrings/suffixes. Following the idea, an IS-based suffix sorting algorithm is typically comprised of a reduction phase for sorting and naming substrings to reduce a string $x[0, n)$ to a short string $x1[0, n1)$ with $n1 \leq \frac{1}{2}n$ and an induction phase for sorting suffixes to induce $\mathsf{SA}(x)$ from $\mathsf{SA}(x1)$. During the two phases, the key operation is to retrieve the preceding character of a sorted substring/suffix. This can be done very quickly when x is fully accommodated in the internal memory, but will become slow when x resides in the external memory, as each operation takes a random disk access. For a high I/O efficiency, eSAIS, EM-SA-DS and DSA-IS use different auxiliary data structures to retrieve the preceding characters in a disk-friendly way. Particularly, both eSAIS and EM-SA-DS split a long substring into pieces and represent each piece by a fixed-size tuple, while DSA-IS does not. With an elaborate arrangement of the I/O operations, the programs for these three algorithms are competitive with those for others in terms of both time and space efficiencies.

Among the existing internal-memory suffix sorters, SA-IS [1] achieves the optimal time and space complexities using the induced sorting method, the key operation of which is to retrieve the preceding characters of sorted suffixes in order for inducing the lexical order of unsorted ones. In the past five years, several works adapted SA-IS to design suffix sorters specific for EM models [**?**], [3], [5], [7]. These variants use different approaches to retrieve the preceding characters by sequential I/O operations. However, they all suffer from a space bottleneck for taking at least twice disk space as SA-IS on real-world datasets. Recently, it was presented in [**?**] a new engineering version of SA-IS that achieves nearly optimal space efficiency, indicating a great potential for improving the other disk-based IS alternatives by engineering them carefully.

Spae

2 Conclusion

xxx

References

[1] G. Nong, S. Zhang, and W. H. Chan, "Two Efficient Algorithms for Linear Time Suffix Array Construction," *IEEE Transactions on Computers*, vol. 60, no. 10, pp. 1471–1484, October 2011.

[2] T. Bingmann, J. Fischer, and V. Osipov, "Inducing Suffix and LCP Arrays in External Memory," in *Proceedings of the 15th Workshop on Algorithm Engineering and Experiments*, 2012, pp. 88–102.

[3] G. Nong, W. H. Chan, S. Q. Hu, and Y. Wu, "Induced Sorting Suffixes in External Memory," *ACM Transactions on Information Systems*, vol. 33, no. 3, pp. 12:1–12:15, March 2015.

[4] W. J. Liu, G. Nong, W. H. Chan, and Y. Wu, "Induced Sorting Suffixes in External Memory with Better Design and Less Space," in *Proceedings of the 22nd International Symposium on String Processing and Information Retrieval*, London, UK, September 2015, pp. 83–94.

[5] J. Kärkkäinen and D. Kempa, "Engineering a Lightweight External Memory Suffix Array Construction Algorithm," in *Proceedings of the 2nd International Conference on Algorithms for Big Data*, Palermo, Italy, April 2014, pp. 53–60.

[6] J. Kärkkäinen, D. Kempa, and S. J. Puglisi, "Parallel External Memory Suffix Sorting," in *In proceedings of the 26th Annual Symposium on Combinatorial Pattern Matching*, Ischia Island, Italy, July 2015, pp. 329–342.

[7] G. Nong, W. H. Chan, S. Zhang, and X. F. Guan, "Suffix Array Construction in External Memory Using D-Critical Substrings," *ACM Transactions on Information Systems*, vol. 32, no. 1, pp. 1:1–1:15, January 2014.