Response to Review Comments

Comments:

Reviewer #1: Besides, a more complete experimental evaluation is now performed, also including a comparison with existing solutions for building the suffix and LCP arrays. Regarding to this, and just as a minor note, I'd really encourage the authors to better motivate the aim of this last comparison, as they propose two checking methods but the programs they use to compare with are both builder solutions. Finally, as recommended, authors have also identified and discussed further improvements that could be applied to their proposals, with supportive references and explanations.

Response:

On page 6, we explain the reason for comparing our solutions with the two builders as follows:

> Because there is no solution for checking both suffix and LCP arrays in the existing literature, comparing the outputs of two builders could be a way for checking (even though it is not checking in the strict sense, for both outputs may be incorrect). In the next experiment, we compare our programs with two builders for suffix and LCP arrays as follows, where each of them combines an existing suffix sorter with an LCP builder:

In this revision, we also compare our solutions (for checking a pair of suffix and LCP arrays) with the only existing SA checking method in the literature. The results are shown in Fig. 2 and 3, and the reason for this comparison is explained on page 7 as follows:

> Currently, the method described in [26] is the only known in the literature for checking SA. Despite that it can check SA only, in order to give a rough image for the performance of our methods, we implement it by STXXL and compare our implementation with ProgA and ProgB/ProgB+ in Fig. 2 and 3. This method and its program are denoted by "Method C" and "ProgC", respectively. ProgC is about two times faster than ProgA and three times faster than ProgB+, where the runtime is consistent with the I/O volume. This performance gap can be significantly narrowed by improving the algorithm and program designs of Methods A and B, using the techniques discussed below.

In Section 4.3, we give a discussion about the pros and cons of these methods in our experiments, as well as the optimization techniques for improving the performance of our methods. Finally, we update the conclusion with the last paragraph as follows:

> The IS method has been applied to successfully design a number of suffix and LCP arrays construction algorithms. A recent work [30] reports that a careful engineering of the IS in external memory can build a suffix array using around 8n bytes for $n \leq 2^{40}$, which is approaching 6n bytes for the IS in internal memory. Besides, it runs the fastest for large n in the experiments therein. This convinces that the IS method could be a stand for developing potentially optimal solutions for building suffix/LCP arrays. We design here the algorithms for checking a pair of given suffix and LCP arrays. In another paper, we will come up with a solution for building and checking a suffix/LCP array simultaneously using the IS method. By this way, no additional checker is needed to be distributed with a suffix/LCP array builder using the IS method.

Thanks a lot!