

2. Project in Text Indexing (WS 2015/16)

<http://algo2.iti.kit.edu/2553.php>
gog@kit.edu

Aufgabe 1 (*More space-efficient SA sampling*)

The bitvector which was used in the lecture to indicate whether $SA[i] \equiv 0 \pmod s$ is a considerable overhead for texts over small alphabets like DNA sequences, especially if it is not compressed. Early CSA implementations use a much simpler scheme. The scheme samples every i -th value in SA (i.e. $SA[i]$ is sampled iff $i \equiv 0 \pmod s$). We call the latter scheme *SA-order-sampling* and the first one *text-order-sampling*.

- Does the upper bound of $s - 1$ LF steps to calculate an arbitrary SA value still hold in the new scheme? Proof your answer.
- Present an example where *text-order-sampling* is faster in practice than *SA-order-sampling* and vice versa.

Aufgabe 2 (*H_0 -compressed bitvector*)

In the lecture we have discussed the compressed bitvector of Raman, Raman, and Rao (`rrr_vector<>` in SDSL). We have seen that increasing the block size K decreased the size of the `rrr_vector` representation of different bitvectors.

Is this true in general? Hint: Try to find a counter example.

Aufgabe 3 (*“,2 in 1”*)

We have seen that Ψ and LF can be expressed via SA and ISA . Therefore a naive implementation of a data structure which supports constant time access to Ψ , LF , SA and ISA takes $2n \log n$ bits. Give a solution which takes only $n \log n + o(n \log n)$ bits and still provides access in $O(\log n)$ time.

- Implement your solution.
- Extra point: Integrate your implementation in the SDSL framework.