

# GENOME-SCALE ALGORITHM DESIGN

by Veli Mäkinen, Djamel Belazzougui, Fabio Cunial and Alexandru I. Tomescu

Cambridge University Press, 2nd edition, 2023

www.genome-scale.info

---

## Exercises for Chapter 12. Compression of genome collections

- 12.1 Recall from Exercise 11.7 that a  $\sigma$ -ary de Bruijn sequence of order  $k$  is a string of length  $\sigma^k + k - 1$  over an alphabet of size  $\sigma$  that contains as substrings all the  $\sigma^k$  possible strings of length  $k$ . What is the number of phrases in the Lempel–Ziv parse of such a sequence? What does this tell you about the asymptotic worst-case complexity of a Lempel–Ziv parse? *Hint.* Combine your result with Lemma 12.2.
- 12.2 Consider the solution of Exercise 12.1. Is the size of the Lempel–Ziv parse of a string a good approximation of its Kolmogorov complexity, roughly defined in Section 11.3.5?
- 12.3 Show how to build in linear time the augmented suffix tree used in Section 12.1.1.
- 12.4 Show how to build the vector  $R$  described in Section 12.1.2 in  $O(n \log \sigma)$  time and in  $O(n \log \sigma)$  bits of space.
- 12.5 Prove Lemma 12.5 shown in Section 12.1.2.
- 12.6 Consider a generalization of Lempel–Ziv parsing where a phrase can point to its previous occurrence as a *reverse complement*. Show that the space-efficient parsing algorithms of Section 12.1 can be generalized to this setting. Can you modify also the bit-optimal variant to cope with reverse complements?
- 12.7 Prove the monotonicity of cost property for parse graphs described in Section 12.2.
- 12.8 Prove Lemma 12.12 by contradiction, working on the shortest path  $P = v_{x_1}v_{x_2} \dots v_{x_m}$  of the parse graph  $G$  such that its first nonmaximal arc  $(v_{x_i}, v_{x_{i+1}})$  occurs as late as possible.
- 12.9 Prove Theorem 12.13, using the algorithm for computing the single-source shortest path in a DAG described in Section 4.1.2.
- 12.10 Consider all arcs in the parse graph  $G$  of Definition 12.10 that start from a given vertex, and let  $b_{i_1}, b_{i_2}, \dots, b_{i_k}$  be a partition of such arcs into equivalence classes according to the number of bits required to encode their copy distances using encoder  $f$ . Describe an algorithm that, given the length of the distance-maximal arc of class  $b_{i_p}$  and the length of the distance-maximal arc of the previous class  $b_{i_{p-1}}$ , where  $p \in [2..k]$ , returns all the length-maximal arcs inside class  $b_{i_p}$ , in constant time per length-maximal arc.
- 12.11 Recall from Section 12.2 the relationship between the length  $\ell_e$  of a distance-maximal arc  $e$  and the LCP of the suffixes of  $S$  that start at those positions which are addressable with the number of bits used to encode the copy distance  $d_e$  of arc  $e$ . Prove this property, using the definition of a distance-maximal arc.

- 12.12 With reference to Lemma 12.18, show that sorting the set of suffixes  $S'[h..|S|]\#$  with  $h \in [p..q]$  coincides with computing the suffix array of string  $W$ .
- 12.13 Adapt Lemma 12.18 to the case in which **source** and **target** do not intersect. *Hint.* Sort the suffixes that start inside each of the two intervals separately, and merge the results using LCP queries.
- 12.14 Describe how suffixes can be efficiently partitioned into extra, determined, and undetermined in the prefix-free parsing scheme for sorting the suffixes of a text.
- 12.15 Give pseudocode for suffix sorting using prefix-free parsing.
- 12.16 Give pseudocode for run-length BWT construction using prefix-free parsing.