Verified correctness and complexity analysis of the Uno-Yagiura algorithm

Binh-Minh Bui-Xuan Pierre-Évariste Dagand

Abstract

This internship offers to develop a certified implementation of Uno and Yagiura algorithm for enumerating the common intervals of two permutations. This algorithm plays a fundamental role in computational biology: we shall not only strive to prove its correctness – it returns correct results – but also to mechanically check its complexity bounds – its complexity is linear with respect to the permutation's size. The internship will be held at LIP6 (Université Paris 6) jointly in the Whisper and Apr team.

Uno and Yagiura [2000] have developed an efficient algorithm for computing gene clusters across species. To the geneticists, being able to identify common gene clusters provide a wealth of information, such as the evolution of species or their selection mechanisms. To the computer scientist, Uno and Yagiura's algorithm appears to be a delicate and rather puzzling piece of engineering: it is challenging to explain *how* it works and *why* its complexity is linear.

The original publication did not help in that respect: the algorithm's correctness "proof" is described in about 10 rather cryptic lines. Bui-Xuan et al. [2005] revisited the algorithm, expounding its invariant and structural properties. As part of an earlier internship, Fleury [2013] worked toward mechanically verifying its functional correctness in the Coq proof assistant, achieving promising results on a simplified variant of the algorithm. However, this work did not establish the correctness of the full-blown version of the algorithm. Also, being focused on the functional correctness, the complexity of the algorithm was not formally established.

Objectives: This internship aims to tackle these remaining roadblocks. Concretely, we wish to:

- Prove the functional correctness and complexity bounds of a naïve variant of the Uno-Yagiura algorithm in the Coq proof assistant;
- Describe the optimized version through a series of *refinements* over the naïve one, thus paving the way for a conceptually simpler presentation of the algorithm;
- Taking advantage of the various refinements, prove the functional correctness and establish the complexity bounds of each refinement in the Coq proof assistant.

Learning outcomes: First of all, this internship offers an opportunity to learn the ins and outs of certified program development. We shall tackle the difficult task of reasoning about mutable data-structures and formalizing some amortized complexity arguments, following the footsteps of Charguéraud and Pottier [2015] in their work on certifying the union-find algorithm. This internship is also an opportunity to delve into the algorithmic of texts, the combinatorics of permutations and of generative functions over intervals.

Student's profile: We are looking for a student at ease with algorithm design and implementation (complexity analysis, correctness reasoning) and interested in formalizing such results in a proof assistant. Acquaintance with an interactive theorem prover (Coq, or Isabelle) is recommended. Nonetheless, a motivated student with a strong background in functional programming (OCaml, or Haskell) could certainly learn to use Coq along the way [Pierce et al., 2015, Chlipala, 2013].

References

B.-M. Bui-Xuan, M. Habib, and C. Paul. Revisiting T. Uno and M. Yagiura's algorithm. In X. Deng and D.-Z. Du, editors, *Algorithms and Computation*, volume 3827 of *Lecture Notes in Computer Science*, pages 146–155. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30935-2. doi:10.1007/11602613_16.

A. Charguéraud and F. Pottier. Machine-checked verification of the correctness and amortized complexity of an efficient union-find implementation. In C. Urban and X. Zhang, editors, *Interactive Theorem Proving*, volume 9236 of *Lecture Notes in Computer Science*, pages 137–153. Springer International Publishing, 2015. ISBN 978-3-319-22101-4. doi:10.1007/978-3-319-22102-1_9.

A. Chlipala. *Certified Programming with Dependent Types - A Pragmatic Introduction to the Coq Proof Assistant*. MIT Press, 2013. ISBN 978-0-262-02665-9. URL `http://mitpress.mit.edu/books/certified-programming-dependent-types`.

M. Fleury. Formal proof of Uno and Yagiura's algorithm. Technical report, ENS Cachan Bretagne, 2013.

B. C. Pierce, C. Casinghino, M. Gaboardi, M. Greenberg, C. Hriţcu, V. Sjoberg, and B. Yorgey. *Software Foundations*. Electronic textbook, 2015. URL `http://www.cis.upenn.edu/~bcpierce/sf`.

T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000. ISSN 0178-4617. doi:10.1007/s004539910014.