From Peaks to Valleys, Running Up and Down: Fast Permutation Pattern Matching

Marie-Louise Bruner and Martin Lackner
Vienna University of Technology

marie-louise.bruner@tuwien.ac.at, lackner@dbai.tuwien.ac.at

ABSTRACT

Permutations are fundamental objects in combinatorics. An n-permutation is a sequence containing every one of the integers $1, 2, \ldots, n$ exactly once. Within combinatorics, patterns in permutations are an important topic. A permutation T contains a permutation P as a pattern if there exists a subsequence of T that has the same relative order of elements as P. For example 53142 contains 231 as shown by the subsequence 342. On the other hand 53142 avoids 123 since it does not contain an increasing subsequence of length 3. Pattern avoidance is related to problems in bioinformatics such as genome rearrangement, to sorting algorithms, etc., see [3].

A wealth of combinatorial results is known about pattern-avoiding permutations. However, far less is known about computational aspects. The natural problem Permutation Pattern Matching, short PPM, asks whether the pattern P is contained in T. In [1] it was shown that PPM is NP-complete. So far the brute-force algorithm is the fastest algorithm. It requires $\mathcal{O}^*(2^{|T|})$ time, i.e. the exponential runtime depends on the length of T. This result could be improved in two ways: either by reducing the constant 2 or by replacing the exponent by a smaller parameter. One possible parameter is $run(T)$, the number of runs in T. It describes the number of ups and downs in T. For instance 53124876 has three runs: 531 (1 is the first valley), 248 (8 is the following peak) and 76. It always holds that $run(T) < |T|$ and on average T has $2|T|/3$ runs.

BODY

We designed a PPM *algorithm with* $\mathcal{O}^*(1.8^{run(T)})$ *runtime [2]. This improves both base and exponent. For* T *with few runs it is especially fast.*

REFERENCES

[1] P. Bose, J. Buss, and A. Lubiw. Pattern matching for permutations. In F. Dehne, J.-R. Sack, N. Santoro, and S. Whitesides, editors, *Algorithms and Data Structures*, volume 709 of *Lecture Notes in Computer Science*, pages 200–209. Springer Berlin / Heidelberg, 1993.

[2] M.-L. Bruner and M. Lackner. A fast algorithm for permutation pattern matching based on alternating runs. In F. Fomin and P. Kaski, editors, *Algorithm Theory - SWAT 2012*, volume 7357 of *Lecture Notes in Computer Science*, pages 261–270. Springer Berlin / Heidelberg, 2012. Also available at http://arxiv.org/abs/1204.5224.

[3] S. Kitaev. *Patterns in Permutations and Words*. Springer Berlin / Heidelberg, 2011.

Volume 1 of Tiny Transactions on Computer Science