TPPmark2024 : Permutation Sort

Jacques Garrigue

Graduate School of Mathematics, Nagoya University

TPP 2024, Kyushu University, November 26, 2024

The problem

We want to sort a list in place, using a minimal number of exchanges. We do not count the time needed to decide the sequence of exchanges (but it should probably be tractable.)

1. Define a function which, given a permutation, returns the a minimal sequence of exchanges which realizes it.
2. Prove that this function is correct, and that the number of exchanges is minimal.
3. Define a function which, given a list of natural numbers without repetition, returns a minimal sequence of exchanges which sorts it. Prove it correct and minimal.
4. Define a function which, given a list of natural numbers possibly with repetitions, returns a minimal sequence of exchanges which sorts it. Prove it correct and minimal.

Permutations

You have probably learnt in you linear algebra course that, considering a finite set of n elements, there exists a group S_n, the *symmetric group*, containing all the permutations.
In particular

1. any permutation p has a unique decomposition into *cycles* (permutations defined by a list of elements $(a_1\ a_2 \dots a_m)$ such $p(a_i) = a_{i+1}$ and $p(a_m) = a_1$). All these cycles have disjoint supports, so they commute.
2. A cycle of length m can be realized by the composition of $m - 1$ *exchanges* (permutations that only exchange two elements).

$$(a_1\ a_2 \dots a_m) = (a_1\ a_m) \cdot \dots \cdot (a_1\ a_3) \cdot (a_1\ a_2)$$

Any a_i is first exchanged with a_1, but in the step it is exchanged back to a_{i+1}.

Expected solution

1. Define a function doing the above decomposition.
2. Prove that it is optimal.
3. Use it to provide optimal sorting without duplicates.
4. Problem 4 is actually NP-hard: (citation from Yamamoto-sensei)

   Amihood Amir, Tzvika Hartman, Oren Kapah, Avivit Levy, and Ely Porat.
   On the cost of interchange rearrangement in strings.
   SIAM Journal on Computing, 39(4):1444-1461, 2010.

Alternative solution

Without hours of publishing the problem, Yamamoto-sensei sent me his solution.

1. Any permutation has a decompostion into exchanges (the proof is available in MathComp/fingroup), so just use `argmin` to return the shortest sequence.
2. Problem 3 and 4 can use the same solution: by definition a sorting function can only permute its input, so just return the shortest sequence.

Of course, while this is in theory computable, you will probably have to wait a while for the answer. . . .

The solutions

- Mitsuharu Yamamoto (both `argmin` and direct decomposition, MathComp/fingroup)
- Takefumi Saikawa (`argmin` approach, MathComp/fingroup)
- Kenta Inoue (standard approach?, MathComp/fingroup)
- Myself (standard approach, MathComp/ssreflect)

Main lemma

The idea of the proof of minimality is that

- the identity permutation has n cycles (if we see x such that $p(x) = x$ as a cycle of length 1)
- when composing a permutation p with an exchange t, the number of cycles may not decrease by more than 1
- as a result one cannot do better than $n - m$ exchanges (which is the number of exchanges in the standard decomposition)

We can indeed prove the following lemma.

Lemma

If p has m cycles, then $p \cdot t$ has either $m - 1$ or $m + 1$ cycles.

Actually, this lemma is available in MathComp/fingroup/perm.v, but nobody seems to have used it :)