Internet Problem Solving Contest

IPSC 2016
Solution to Problem C – Counting swaps

- Correct output for C1
- Correct output for C2
- Python 3 program solving both subproblems

For the easy subproblem, we can view all permutations as vertices in a graph and swaps as edges between them. We're looking for the number of shortest paths from the identity permutation to the input permutation (or vice versa), which can be computed using a single breadth-first search.

In the hard subproblem we can use a different graph idea – the graph of a permutation. Consider a graph with vertices $1, \ldots, n$ and n directed edges: from each i to the corresponding p_i. Clearly, for any permutation this graph is a collection of disjoint cycles. We can easily verify that swapping two numbers from different cycles merges those cycles together. By symmetry, swapping two numbers from the same cycle breaks the cycle into two smaller ones.

The identity permutation consists of n single-element cycles. From this, it's apparent that if the input permutation consists of c cycles, the minimum number of swaps needed to sort it is $n - c$. Furthermore, a sequence of $n - c$ swaps is a solution if and only if each of them swaps two numbers that currently lie on the same cycle.

Obviously, each cycle is completely independent from the others. Also, we do not care about the particular elements a cycle contains. Hence, the answer to our problem is completely determined by the sequence of cycle lengths. The cycle lengths are the only thing that matters.

Suppose we have a cycle of length l and we want to make a swap that splits it into two cycles of lengths l_1 and l_2. How many such swaps exist? Clearly, the answer is l, excepf for one special case. If l is even and $l_1 = l_2$, there are only $l/2$ such swaps. We will use $S(l, l_1)$ to denote this number of swaps.

How many ways are there to split one cycle of length l into l cycles of length 1 using a sequence of exactly $l - 1$ swaps? Let's use $C(l)$ to denote the answer to this question.

We can use dynamic programming to compute $C(l)$. We have $C(1)=1$. Now let $l > 1$. In the first step we will produce two cycles of lengths l_1 and $l - l_1$. We can try all l_1 from 1 to $\lfloor l/2 \rfloor$. For each l_1 we can then continue splitting each smaller cycle separately. There are $C(l_1)$ ways to split the first cycle, $C(l - l_1)$ ways to split the second cycle, and $\binom{l-2}{l_1-1}$ ways to interleave those two independent sequences of swaps. The above gives us the following recurrence:

$$C(l) = \sum_{l_1} S(l, l_1) \cdot C(l_1) \cdot C(l - l_1) \cdot \binom{l-2}{l_1-1}.$$

We can precompute the binomial coefficients and then evaluate this recurrence in $O(l^2)$. That is still too slow for the hard subproblem, but it is already fast enough to tell us a lot of values of our sequence – more than enough to guess the pattern, or to make a lookup in the Online Encyclopedia of Integer

Sequences (OEIS). It turns out that for $l \geq 2$ we have $C(l)=l^{l-2}$. This can be computed quickly using modular exponentiation.

Now that we can solve our question for a single cycle, all that remains is to count the ways in which we can interleave the solutions to all cycles of the given permutation. This is very simple: the answer is the appropriate multinomial coefficient.

Imagine that we have a permutation with cycle lengths $l_1, \ldots, l_c$ and that we already chose one specific solution for each cycle. The number of ways to interleave these solutions is always the same:

$$\binom{n-c}{l_1-1,\ l_2-1,\ \ldots,\ l_c-1} = \frac{(n-c)!}{(l_1-1)!\ldots(l_c-1)!}\ .$$

The modulus is a prime larger than n, so the factorials in the denominator can be replaced by their multiplicative inverses $inv[(l_k-1)!]$ in the numerator. Thus, the complete answer can be computed as follows:

$$(n-c)!\prod_{k=1}^{c} inv[(l_k-1)!]\cdot l_k^{l_k-1}\ .$$

Note that $C(l)$ is also Cayley's formula giving the number of labeled trees on n vertices. As an exercise, try finding a combinatorial bijection between labeled trees and our sequences of swaps.