Art Collections (art)

BY DAVID RASMUSSEN LOLCK (DENMARK)

In this task we want to find a secret permutation while only being able to query at most 4 000 times for the number of inversions.

Subtask 1. $N \leq 6$

In this subtask we have well in excess of $N!$ allowed queries, so we can query every single permutation and output the permutation with 0 inversions.

Subtask 2. $N \leq 40$

In this subtask we have more than $2N^2$ allowed queries. We notice that if we swap two adjacent elements at the positions i and $i+1$ in a permutation P, the number of inversions will increase by one if $P[i]$ comes before $P[i+1]$ in the secret permutation and decrease by one if $P[i]$ comes after $P[i+1]$ in the secret permutation. This way, we can compare any two elements with 2 queries. Using this, we can implement any N^2 sorting algorithm (for example bubblesort, insertionsort, selection sort, etc) with $2 \cdot N^2$ queries.

Subtask 3. $N \leq 250$

In this subtask we are allowed $2N \cdot \lceil \log_2(N) \rceil$ queries. We can now use our comparator from the previous subtask with mergesort to solve this subtask. Note that while **std**::*sort* is guaranteed to only make $O(N \log N)$ many comparisons, the constant of this algorithm is insufficient for this subtask. In contrast, **std**::*stable_sort* implements mergesort and can be used to solve this subtask.

Subtask 4. $N \leq 444$

In this subtask the limit is reduced to $N \cdot \lceil \log_2(N) \rceil$ allowed queries, so we need a way to compare two elements using only a single query. For this, we notice that the permutation

$$[1, 2, \dots, i-1, i, i+1, \dots, j-1, j, j+1, \dots N]$$

has fewer inversions than

$$[1, 2, \dots, i-1, j, i+1, \dots, j-1, i, j+1, \dots N]$$

if and only if i comes before j in the secret permutation. Thus, we can query the permutation $[1, 2, \dots, N]$ once at the start of the program and then compare any two elements with a single additional query. Using this new comparator, mergesort solves this subtask.

Subtask 5. $N \leq 2\,000$

For some element i, we compare the number of inversions of the two permutations

$$P_1 = [i, i+1, \dots, N, 1, 2, \dots, i-1] \quad \text{and} \quad P_2 = [i+1, \dots, N, 1, 2, \dots, i-1, i].$$

Let k be the number of inversions not involving element i of P_1, ℓ_1 be the number of inversions of P_1, and ℓ_2 be the number of inversions of P_2. Note that the k inversions not involving element i are exactly the same for P_1 and P_2. The number of remaining inversions $j_1 = \ell_1 - k$ of P_1 thus all involve element i. Since i is the first element in P_1, these remaining inversions are precisely the number of elements smaller than i in the secret permutation. Similarly, the number of remaining inversions $j_2 = \ell_2 - k$ of P_2 is precisely the number of elements greater than i in the secret permutation. Hence, we also have $j_1 + j_2 = N - 1$. Combining all of these observations, we see that element i is at position

$$j_1 + 1 = \frac{2j_1}{2} + 1 = \frac{j_1 + N - 1 - j_2}{2} + 1 = \frac{j_1 + k - (j_2 + k) + N - 1}{2} + 1 = \frac{\ell_1 - \ell_2 + N - 1}{2} + 1.$$

Since we can obtain ℓ_1 and ℓ_2 with two queries, we can determine the positions of all elements individually with $2N$ queries.

Subtask 6. No further constraints.

For the full solution, we notice that with our choice of the cyclic permutations in the previous subtask, we query each permutation exactly twice and can thus remember the return values of *publish*. This reduces the amount of queries to N.