

Assignment 6 - Rearranging cars

1 Problem specification

There is a parking lot with N spaces and $N-1$ cars in it. Your task is to write an algorithm to rearrange the cars in a given way. Only one car can be moved at a time to the empty slot.

The parking lot is described by an array of numbers. Let's identify cars with numbers from 1 to $N-1$, and the number 0 means an empty parking space.

The input to your function is two arrays, each with a permutation of the numbers 0 to N (you don't have to validate it). Your function must generate a series of moves and print them.

For example, with $N=4$ and the inputs $[1, 2, 0, 3]$ and $[3, 1, 2, 0]$, the following output could be generated:

- move from 0 to 2
- move from 3 to 0
- move from 1 to 3
- move from 2 to 1
- move from 3 to 2

2 Solution

2.1 Terminology

1. Here we will consider the empty place as a 0 car.
2. Let's formalize the problem. We are given two arrays of cars:

first state of the parking - $(a_1, a_2, a_3, \dots, a_n)$

next state of the parking - $(b_1, b_2, b_3, \dots, b_n)$

and we need to make the first array equal to the second by rearranging it. So it is the same thing as rearranging the permutation $\begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ b_1 & b_2 & b_3 & \dots & b_n \end{pmatrix}$ into identity permutation.

3. As we know each permutation can be written as $\begin{pmatrix} 0 & 1 & 2 & \dots & n-1 \\ c_1 & c_2 & c_3 & \dots & c_n \end{pmatrix}$ and can be considered as a composition of **cycles**:
 $(c_{i1} \ c_{i2} \ \dots \ c_{ik}) \circ \dots \circ (c_{j1} \ c_{j2} \ \dots \ c_{jl})$
4. Let's call **chain** a cycle which includes 0-element
5. As we also know, a transposition is therefore a permutation of two elements. So let's call **zero-transposition** a transposition which includes 0-element
6. **real-cycle** - cycle that contains more than 1 element and don't include 0-element
7. **fake-cycle** - 1-element cycle
8. So now we need to rearrange given permutation into identity permutation using zero-transpositions.

2.2 Number of moves

Goal: Minimise a number of moves

MAIN Theorem 1 (Minimal number of moves). *Minimal number of moves always equals to $N + K - 1 - Q$, where*

N - a size of a permutation (a number of slots in the parking lot)

K - a number of real-cycles

Q - a number of fake-cycles

Proposition 1 (chain). *Each chain can be rearranged by $t - 1$ steps, where t is it's length. And can't be rearranged less than by $t - 1$ steps.*

It's important that by one step we can put only one non-zero number on it's position. So it can't be done by less than $t - 1$ zero-transpositions. And there is an algorithm which shows how to accomplish this by exactly $t - 1$ transposition: we need to find a number that must be in a place where now is 0 and swap it with zero. Then we need to repeat this action $t - 2$ more times. Thus each iteration puts one number in a right position. (I won't write here a strict prove of it because it results from chain's definition)

Proposition 2 (real-cycle). *Each real-cycle can be rearranged by $t + 1$ steps, where t is it's length*

The main difference between chain and real-cycle is that we can't start rearranging cycle immediately: we can put a number in real cycle in the right place by 1 step. So, anyway, we need an additional step to implement 0 into the real-cycle and make a chain, which length is $t + 1$.

Now we're ready to prove the MAIN Theorem.

Number $N - 1 - Q$ states for cars which are not in the right places (here we don't consider 0-car). As it was stated: by one step we can put only one non-zero number on it's position. So we need all these steps. K states for a number of additional moves (described in Proposition 2). So the answer is $N - 1 + k - Q$.

2.3 Algorithm

See arrangement.h, arrangement.cpp