Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Department Informatik
Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger
Tomáš Gavenčiak

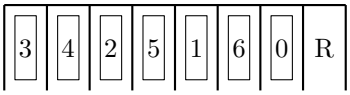
27th October 2016

Datenstrukturen & Algorithmen Exercise Sheet P6 AS 16

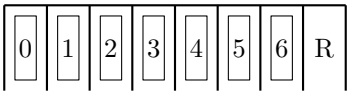
Hand-in: Before Thursday, 3rd November 2016 10:00 via the online judge (source code only).

Exercise P6.1 *Chinese wall parking.*

Along the Chinese Great Wall, there is a very long parking place and, as always, it is full of visitors' cars except for one reserved space at the right end, like in the following picture.



The n cars are numbered from 0 to $n-1$ and the visitors parked them just arbitrarily, as you can see. However, the Chinese regulations now require that the cars must be ordered by their number from left to right and with the reserved space empty, as in the following picture.



You are given the task to re-park the cars into the right order. You will be given $n \geq 1$, the number of cars, and their numbers as they are initially parked left-to-right as an array $a_0 \dots a_{n-1}$. The car numbers are $0 \dots (n-1)$, each number used exactly once, so car i should always go into position i (for i from 0 to $n-1$).

You should output the *minimal* number of car moves required to order the cars as prescribed by the law. One *car move* is to move any single car from its current position to an empty space (including the one reserved space). Note that there is always exactly one free space in the parking place when you finish a move, and it is not allowed to leave a car on the road in the middle of the move. It does not matter how far you move the car in a single move.

The following is one of the possible shortest sequences of car moves with the intermediate states, having 8 moves. The character `_` denotes the empty space.

```
3 4 2 5 1 6 0 _   move car 3
_ 4 2 5 1 6 0 3   move car 0
0 4 2 5 1 6 _ 3   move car 6
0 4 2 5 1 _ 6 3   move car 5
0 4 2 _ 1 5 6 3   move car 3
0 4 2 3 1 5 6 _   move car 1
0 4 2 3 _ 5 6 1   move car 4
0 _ 2 3 4 5 6 1   move car 1
```

```
0 1 2 3 4 5 6 _   done
```

Note that you are not required to actually construct the sequence of moves, only determine the minimal number of moves required. Also note that there are many possible optimal sequences.

Input The input consists of two lines. The first line contains just the integer n. The second line contains n integers a_0 to a_{n-1} (the car numbers) separated by spaces.

Output The output should contain one number, the minimal number of car moves required to order the cars.

Grading You will get 1 bonus point for every 100 judge points, rounded down. Your may get up to 200 judge points. The program should be reasonably efficient and work in $O(n)$ time or similar to get full points.

Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=18985&problem=DA_P6.1`, enroll password is "`quicksort`".

Examples

Input (the example above):

```
7
3 4 2 5 1 6 0
```

Output:

```
8
```

Input (cars already ordered):

```
5
0 1 2 3 4
```

Output:

```
0
```

Notes For this exercise, we provide a program template as an Eclipse project archive on the lecture website, which will load the input for you. The archive also contains more tests for you convenience – you can copy&paste the data into your running program.