Long Take-Home Exam 1

(Deadline: 5th of April 2023, Wednesday, 23:59)

1 Problem

In this exam, you will write a solver for a puzzle that we named "Reversal". As an instance of the puzzle, you receive an arbitrary permutation of the integers from 1 to n. The goal is to end up with the monotonically increasing permutation $1, \ldots, n$.

To achieve this goal, you are allowed to make a move by changing the permutation in the following way: Let $a_1, \ldots, a_n$ be the permutation at hand. Choose two integers i and j, such that $1 \leq i < j \leq n$. Then, reverse the range of elements between a_i and a_j (inclusively). In other words, you are allowed to change the permutation

$$a_1, \ldots, a_n$$

to the permutation

$$a_1, \ldots, a_{i-1}, a_j, a_{j-1}, \ldots, a_{i+1}, a_i, a_{j+1}, \ldots, a_n$$

by making the move (i, j).

Your task is to write a program that finds the minimum number of moves that requires to solve a given instance of the puzzle and lists the moves in order.

2 Input Format

```
A1 A2 ... AN
```

$\texttt{A}_i$ = The ith element of the initial permutation.
$\texttt{N}$ = The length of the initial permutation.

3 Output Format

```
M
I1 J1
...
IM JM
```

$\texttt{M}$ = Minimum number of moves required to solve the puzzle.
$\texttt{I}_i$ = The first parameter of the ith move.
$\texttt{J}_i$ = The second parameter of the ith move.

4 Limits

$1 \leq \texttt{N} \leq 15$
$1 \leq \texttt{M} \leq 9$
Time Limit: 1 second
Memory Limit: 256 MB

5 Clarifications

- Your solution is expected as a C++ program source named `1the1.cpp` that reads from the standard input and writes to the standard output.

- It is OK to copy code from the sample codes we shared in our course website in ODTÜClass for this THE. Copying from any other source would be considered cheating.
- You are supposed to submit your code via the VPL item in ODTÜClass. Use the "evaluate" feature to run the auto-grader on your submission.
- The grade from the auto-grader is not final. We can later do further evaluations of your code and adjust your grade. Solutions that do not attempt a "reasonable solution" to the given task may lose points.
- We will compile your code on g++ with options: `-std=c++2a -O3 -lm -Wall -Wextra -Wpedantic`
- Late submissions are not allowed.

6 Example

Sample Input 1

```
6 4 1 3 2 5
```

Sample Output 1

```
4   # Start:  6 4 1 3 2 5.
1 3 # Result: 1 4 6 3 2 5.
2 5 # Result: 1 2 3 6 4 5.
4 5 # Result: 1 2 3 4 6 5.
5 6 # Result: 1 2 3 4 5 6.
```

Sample Input 2

```
9 7 2 4 6 3 8 5 1
```

Sample Output 2

```
6   # Start:  9 7 2 4 6 3 8 5 1
1 9 # Result: 1 5 8 3 6 4 2 7 9
5 7 # Result: 1 5 8 3 2 4 6 7 9
2 5 # Result: 1 2 3 8 5 4 6 7 9
5 6 # Result: 1 2 3 8 4 5 6 7 9
4 8 # Result: 1 2 3 7 6 5 4 8 9
4 7 # Result: 1 2 3 4 5 6 7 8 9
```

7 Hints

- We suggest using DFID (depth-first iterative deepening) coupled with a reasonably accurate admissible heuristic to solve this problem. (Such an algorithm is also known as IDA* = Iterative Deepening A*.)
- Recall the following heuristic from the lecture:
  - When viewing any permutation, imagine an anchored 0 at the beginning and an anchored $(n+1)$ at the end.
  - Then, the final permutation we want to reach is $\mathbf{0}, 1, \ldots, n, \mathbf{(n+1)}$.
  - Notice that all adjacent pairs of numbers in the goal permutation are consecutive (numerically).

- Given any permutation p, let $t(p)$ be the number of adjacent pairs that are non-consecutive. Clearly, $t(p) = 0$ if and only if p is the goal permutation.
- Every move in the puzzle replaces two of the adjacent pairs with two new ones. (One pair at the left end and one pair at the right end of the reversed range.) All other adjacent pairs are still intact. Note that the adjacent pairs strictly inside the reversed range are still adjacent, only their order are reversed.
- Consequently, each move can decrease the number of non-consecutive adjacent pairs by at most 2.
- It follows $h(p) = \lceil t(p)/2 \rceil$ is an admissible heuristic.