Short Take-Home Exam 1

(Deadline: 7th of April 2023, Friday, 23:59)

1 Problem

In this exam, you will write a solver for a puzzle that we named "Rotator". 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 three integers i, j and s, such that $1 \leq i < j \leq n$ and $1 \leq s \leq j - i$. Then, rotate (i.e., circularly shift) the elements between a_i and a_j (inclusively) to the left s times. In other words, you are allowed to change the permutation

$$a_1, \ldots, a_n$$

to the permutation

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

by making the move (i, j, s).

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

```
A_1 A_2 ... A_N
```

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

3 Output Format

```
M
I_1 J_1 S_1
...
I_M J_M S_M
```

$\texttt{M}$ = 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.
$\texttt{S}_i$ = The third parameter of the ith move.

4 Limits

$1 \leq \texttt{N} \leq 15$
$1 \leq \texttt{M} \leq 7$
Time Limit: 1.5 seconds
Memory Limit: 256 MB

5 Clarifications

- Your solution is expected as a C++ program source named `sthe1.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. You can also copy code from your previous THE submissions for this course. Copying from elsewhere will 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 with `g++` using the options: `-std=c++2a -O3 -lm -Wall -Wextra -Wpedantic`
- Late submissions are not allowed.

6 Example

Sample Input 1

```
4 1 5 3 2
```

Sample Output 1

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

Sample Input 2

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

Sample Output 2

```
5     # Start:  8 1 7 5 6 4 2 9 3
1 2 1 # Result: 1 8 7 5 6 4 2 9 3
2 6 2 # Result: 1 5 6 4 8 7 2 9 3
4 7 2 # Result: 1 5 6 7 2 4 8 9 3
2 6 3 # Result: 1 2 4 5 6 7 8 9 3
3 9 6 # 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. (That is, we suggest using iterative deepening A*.)
- We studied a similar problem and its heuristic previously. That should give you some hints.
- In case you cannot come up with any good heuristic, a well-implemented DFID without any heuristics is able to get half of the points.
- The standard library function `std::rotate` in the `<algorithm>` header may come in handy.