CS440

Assignment 2

Due Feb 25, 2019

Computer Science Department
Colorado State University

Feb 13, 2019

Preliminaries. The input is an array A of size n that contains all the integers $1, \ldots, n$. In A, each and every integer from 1 to n appears once[1].

Input file. The input to your program is called `input.txt`, which contains a representation of A. Make sure your program reads from that file in the current directory. Each entry is put on a separate line. For instance,

line 1: 4
line 2: 1
line 3: 3
line 4: 2
line 5: 5

represents the array $\boxed{4}\boxed{1}\boxed{3}\boxed{2}\boxed{5}$ with $n = 5$.

Sorting by transpositions. A transposition $T(i, j, k)$ is identified by three indices $1 \leq i \leq j < k \leq n$ and swaps the index intervals $[i \cdots j]$ and $[j + 1 \cdots k]$ when applied to an array. For instance, let $i = 2$, $j = 2$, and $k = 5$ and apply $T(2, 2, 5)$ to the array above to obtain $\boxed{4}\boxed{3}\boxed{2}\boxed{5}\boxed{1}$.

We would like to sort an input array with minimum number of transpositions. All transpositions, independent of their indices, have unit cost. That is, we would like to find the minimum k such that transpositions $T_1, T_2, \ldots, T_k$ yield $\boxed{1}\boxed{\cdots}\boxed{n}$ when applied to the input.

Origins of the problem. The problem of sorting by transpositions arose in the context of evolutionary genomics. DNA molecules break and rearrange during the course of evolution by various events including transpositions. Generally, we are interested in finding the minimum number of such events that transform an ancestral genome to a descendant genome.

Output. Your program should print the minimum number of transpositions needed to sort the input array. The actual transpositions is not needed. Hence, your program outputs one integer in the standard output.

[1] If you are familiar with the permutations terminology, A in an n-permutation in the symmetric group S_n.

Grading. We will test your program on 10 different inputs and let it run for 1 minute each. Each test case is worth 10 points. If the output is correct, you get 10 points; otherwise, you get 0. Prepare your program to deal with large values of n.

Implementation suggestions. The problem is proven to be NP-Hard. Hence, we do not know any polynomial-time algorithm for the problem. Moreover, the search space is exponentially large[2]. You are free to use any of the optimal search algorithms that we have discussed in class, namely BFS, Uniform cost, IDS, Bidirectional, or A* and its variants. Be prepared to deal with large input in 1 minute run time on the 120-unix-lab machines such as `denver`.

Upload your answer on Canvas in one zip file or tarball. Include all the code/scripts you have written in your submission as well as (scanned) handwritten or typed answers.

[2] $|S_n| = n!$.