COMP202 **Programming Assignment**
Complexity of Algorithms **19 March 2021**

Title: Longest Anchored Comb Problem

Submission deadline: 26 April 2021 (Monday, 12:00)

Submissions should be made via CANVAS:
`https://https://liverpool.instructure.com/`

Go to COMP202 and to Assignment:
COMP202: CA2 – PROGRAMMING ASSIGNMENT

Notes:

1. This assessment is worth 15% of your overall course grade.
2. Standard late penalties apply, as per university policy.
3. Learning outcomes covered by this CA task will also be covered within resit exam: this prevents the need for explicit reassessment of this component. The resit exam will replace the CA component in case this is failed.

Learning outcomes

The purpose of this exercise is for you to demonstrate the following learning outcomes and for me to assess your achievement of them.

1. To demonstrate how the study of algorithmics has been applied in a number of different domains.
2. To introduce formal concepts of measures of complexity and algorithms analysis.
3. To introduce fundamental methods in data structures and algorithms design.

Note: You will be provided with a collection of sample inputs together with correct answers to these inputs. You should aim at submitting your final program only if it produces correct answers to all these inputs.

Academic integrity

The work that you submit should be the product of yourself (alone), and not that with any other student or outside help. Obviously I am providing a source code framework within which you will provide your own method for solving this problem, but the code that you write within this framework

should be your own code, not that obtained in collaboration with other students or other outside assistance or sources.

Problem Description

Let us first define a notion of a *comb*. Suppose that we are given an array $A[1, 2, \ldots, n]$ containing $n \geq 2$ positive, not necessary distinct, integers. We do not assume that the input array is sorted. Any subsequence in this array of the form $A[i_1], A[i_1 + 1], A[i_2], A[i_2 + 1], \ldots, A[i_k], A[i_k + 1]$ of $2k$ elements of array A, for some integer $k \geq 1$, and for some indices $1 \leq i_1 < i_1 + 1 < i_2 < i_2 + 1 < \cdots < i_k < i_k + 1 \leq n$ such that

$$A[i_1] < A[i_1 + 1] > A[i_2] < A[i_2 + 1] > A[i_3] < \cdots > A[i_k] < A[i_k + 1],$$

is called a *comb*. The above condition on the comb can be written more formally as:

- $A[i_j] < A[i_j + 1]$, for each $j = 1, 2, \ldots, k$, and
- $A[i_j + 1] > A[i_{j+1}]$, for each $j = 1, 2, \ldots, k - 1$.

We call each pair $A[i_j], A[i_j + 1]$ for each $j = 1, 2, \ldots, k$, a *tooth* of the comb. Thus, a comb contains k teeth. Note, that a given tooth contains two *consecutive* elements from array A, and we are allowed to skip any number of elements from array A between two consecutive teeth of the comb. Note also, that if $k = 1$, we will have a comb with just one tooth. The number of teeth, k, in a comb is called its *length*.

We call a given comb $A[i_1], A[i_1 + 1], A[i_2], A[i_2 + 1], \ldots, A[i_k], A[i_k + 1]$ *anchored* if $A[i_1] = A[i_k]$, that is, when its first and last tooth have the same first elements. In particular if $k = 1$ such a comb with one tooth is also considered to be anchored.

I introduce here a problem which I will call the Longest Anchored Comb problem. You are given as input an array $A[1, 2, \ldots, n]$ containing $n \geq 2$ positive, not necessary distinct, integers, and the task is to find the longest (that is, with the largest possible k) *anchored comb* in array A and output its length k (the number of teeth), or output 0 if there is no anchored comb in array A.

For instance if $A[1, 3, 10, 15, 21]$, $n = 5$, then because this sequence is increasing, the longest comb has length $k = 1$, and such longest comb is not unique, e.g., $1, 3$; $3, 10$; $15, 21$ – are examples of 3 longest and anchored combs, each of length 1, i.e., each having one tooth. The output in this instance is therefore 1. Another example of the input array $A = [5, 3, 2, 1]$ with $n = 4$, where the sequence is decreasing means that there is no comb there, so the output to the Longest Anchored Comb Problem is 0.

Let us consider another input $A[1, 3, 2, 11, 12, 10, 11, 2, 23]$ with $n = 9$. Here subsequence $A[1], A[2], A[3], A[4], A[6], A[7], A[8], A[9]$, which is

$$1, 3, 2, 11, 10, 11, 2, 23,$$

is a comb of length $k = 4$ (with 4 teeth), but it is not anchored because the first elements of its first and last teeth are not equal: $A[1] = 1$ and $A[8] = 2$. However, subsequence $A[3], A[4], A[6], A[7], A[8], A[9]$ which is $2, 11, 10, 11, 2, 23$ is an anchored comb of length $k = 3$ (with 3 teeth), because the first elements of its first and last teeth are equal: $A[3] = 2$ and $A[8] = 2$. This is also the longest anchored comb in this instance, so the output to the Longest Anchored Comb problem is 3.

Some further examples of inputs.

Suppose, for instance, that $n = 10$ and that the input sequence is:

1 3 2 4 3 5 4 6 1 3,

that is, $A[1] = 1, A[2] = 3, A[3] = 2, A[4] = 4, A[5] = 3, A[6] = 5, A[7] = 4, A[8] = 6, A[9] = 1, A[10] = 3$. Then, for instance, $A[2] = 3, A[4] = 4, A[5] = 3, A[6] = 5$ is an anchored comb of length 2, but the longest anchored comb is the whole array and has length 5.

Suppose, for instance, that $n = 10$ and that the input sequence is:

1 5 2 6 3 7 8 4 10 1,

that is, $A[1] = 1, A[2] = 5, A[3] = 2, A[4] = 6, A[5] = 3, A[6] = 7, A[7] = 8, A[8] = 4, A[9] = 10, A[10] = 1$. Then, for instance, $A[1] = 1, A[2] = 5, A[3] = 2, A[4] = 6, A[5] = 3, A[6] = 7, A[8] = 4, A[9] = 10$ is a (non-anchored) comb of length 4, the longest anchored comb in this instance has length 1, for instance $A[1] = 1, A[2] = 5$, or $A[3] = 2, A[4] = 6$, and so the output to the Longest Anchored Comb problem is 1.

For further examples of inputs together with answers, see the text file dataTwo.txt that I provide (see explanation of the data format below). The file dataTwo.txt contains also solutions.

You should write a procedure that for any given input sequence of n positive integers (multiple identical numbers allowed) finds the length of the longest anchored comb (or 0 if there is none). Your procedure should only output the value of the longest anchored comb or 0.

Additionally, you should include a brief idea of your solution in the *commented* text in your code, describing how you derive your recursive solution first and ideas of its sequential implementation. You should also include a short analysis and justification of the running time of your procedure in terms of n. These descriptions are part of the assessment of your solution.

Hints

You are supposed to solve the Longest Anchored Comb problem by dynamic programming. A possible initial solution idea from one of the exercises from the exercise list on dynamic programming could inspire your solution here. In your solution (described as part of the assessment) you should come up with appropriate recurrence solution to the problem first, which then you should translate into a sequential solution that fills in a dynamic programming table in an appropriate way in your implementation. As a more specific hint, try to first solve the longest comb problem without the assumption that it is anchored.

Programming Language

You will be using Java as the programming language.

Program framework description

IMPORTANT: Before submitting, you must rename your file `Main123456789.java` where `123456789` is replaced with all digits of your Student ID. You also must rename the main public class `Main123456789{ }` in your file by also replacing `123456789` by all digits of your Student ID.

I provide a template program called `Main123456789.java` that you will use (without altering anything but the place to put your code) to include the code of your procedure to solve the Longest Anchored Comb problem. Note that you may add additional procedures outside of the procedure LongestComb if needed.

To use this template, after you write your code inside of procedure called LongestComb, you must include in the *current directory* the input text files dataOne.txt and dataTwo.txt. Note, however, that if you need any additional procedures, you may include them outside of the text of the procedure LongestComb.

To compile your program in command line (under Linux/Unix) use something like this (this may differ within your system):

```
javac Main123456789.java
```

Then, you can run your program from command line like this

```
java Main123456789 -opt1
```

which will run the program with dataOne.txt as input file and will output answers (that is the values of the longest anchored comb or 0) to all instances in order they appear in dataOne.txt. You may use your own dataOne.txt in the format (see below) to experiment with your program. Input file dataOne.txt may contain any number of correct input sequences.

Now, if you run the program with

```
java Main123456789 -opt2
```

this will run the program with dataTwo.txt as input file. In this case, the output will be the indices of inputs from dataTwo.txt that were solved incorrectly by your program together with percentage of correctly solved instances. If all instances are solved correctly, the output should be 100%. File dataTwo.txt contains the same instances as dataOne.txt, but, in addition dataTwo.txt also contains the correct, that is values of the longest anchored combs or 0, answers to these instances. You may use dataTwo.txt to test the correctness of your program.
Description of the input data format:

Input data text file dataOne.txt has the following format (this example has 2 inputs, each input ends with A; note the number 0 is the part of the input format, but not part of the input sequences):

$$
\begin{array}{c}
0 \\
a_1 \\
a_2 \\
\ldots \\
\ldots \\
a_n \\
A \\
0 \\
a_1 \\
a_2 \\
\ldots \\
\ldots \\
a_n \\
A
\end{array}
$$

In general file dataOne.txt can have an arbitrary number of distinct inputs of arbitrary, varying lengths. File dataOne.txt contains 38 different instances of the problem. Observe that n is not explicitly given as part of the instance. Also 0 which starts each instance does not have any particular purpose; it is just format of the input data to mark beginning of an instance.

Input data text file dataTwo.txt has the following format (this example

has again 2 inputs, each input ends with A):

$$
\begin{array}{c}
0 \\
ans_1 \\
a_1 \\
a_2 \\
... \\
... \\
a_n \\
A \\
0 \\
ans_2 \\
a_1 \\
a_2 \\
... \\
... \\
a_n \\
A
\end{array}
$$

There ans_1 (ans_2, respectively) is the value of the longest anchored comb for the first (second, respectively) instance or 0 if there is no anchored comb in those instances. File dataTwo.txt contains the same 38 instances of the problem as in file dataOne.txt but in addition has the answers. This data can be used to test the correctness of your procedure.

Again, in general file dataTwo.txt can have an arbitrary number of distinct input sequences of arbitrary, varying lengths. It is provided by me with correct answers to these instances.

The solutions shown in dataTwo.txt are (at least) the claimed solutions for each sample input, computed by my program. Recall that your solution should print out the value of the longest anchored comb in the given sequence for the given instance, or 0 in case if this instance contains no anchored comb. Note, that your program does not need to output this longest anchored comb.

Program submission

You must submit a **single** Java source code in a single file that must be called `Main123456789.java` (not the byte code), where `123456789` is replaced with all digits of your Student ID, via CANVAS:
`https://https://liverpool.instructure.com/`

Go to COMP202 and to Assignment: COMP202: CA2 – PROGRAMMING ASSIGNMENT

IMPORTANT: Before submitting, you must rename your file `Main123456789.java` where `123456789` is replaced with all digits of your Student ID. You also

must rename the main public class `Main123456789{ }` in your file by also replacing `123456789` by all digits of your Student ID.

Your source file `Main123456789.java` must have the (unaltered) text of the template provided by me, containing the text of your procedure inside the LongestComb method. You are allowed to include additional procedures outside the LongestComb method if needed. In addition, within commented parts of method LongestComb, you should describe your recursive solution and how you implement it sequentially. Moreover, you should also describe a short running time analysis of your sequential implementation in terms of n and big-O notation.

You are responsible that your source code program `Main123456789.java` can be compiled correctly with Java compiler and executed on (any) one of the computers in the Computer Science Department's that runs Java installation under Linux, where I will test your programs. You may also remotely connect to any Linux computer in the Department to compile/test your program. Programs that will not correctly compile on Departmental Linux Java installation will automatically receive mark 0 for the correctness part.

Assessment

Marking on this assessment will be performed on the following basis:

- Accuracy of solution (e.g., does it give correct answers for all of the test cases, and is it a general solution method for instances of this problem?): 60%
- Clarity of solution (Is your program easy to follow? Is it commented to help me see your solution method?): 10%
- Correctness of time complexity (in big-O notation of the problem size n) description of your procedure and description of your solution. (Have you provided an analysis of the (asymptotic) running time of your method, and is that analysis correct? Is the description of your solution (recursion and sequential implementation) correct and clearly written?: 20%
- Optimality of solution (Do you have the "best", i.e. quickest, solution possible in terms of the asymptotic runtime?): 10%