Imperial College London – Department of Computing

MSc in Computing Science

580: Algorithms

Assessed Coursework 2

1. In this question you are asked to create solutions to the problem of determining whether all the numbers in a given sequence are *distinct*. Each part of the question sets different requirements on the solution. The general problem is as follows:

   Given a sequence $A = [A_1, \ldots, A_N]$ of integers, the procedure DISTINCT should return TRUE if A contains no duplicates and FALSE otherwise.

   (a) Write a version of the DISTINCT procedure that uses $O(1)$ space and $O(N^2)$ time. The space constraint refers to extra memory allocated by the DISTINCT procedure itself. This *excludes* the memory used by the input sequence.

   Discuss the space and time complexity of your solution. Briefly outline one alternative solution that would satisfy the same requirements, and how you would expect its performance to compare to your DISTINCT procedure.

   (b) Can you trade off some space in order to obtain a faster solution to the problem? Your next task is to write a version of the DISTINCT procedure that runs in $O(N)$ time, but is allowed to use $O(N)$ space. You can assume the pre-existence of any of the data structures covered in the course. You can also assume that the elements of A are an *average case* input for any data structure used.

   Discuss the space and time complexity of your solution. Include a discussion of the way your procedure uses any data structure, the relevant operations of the data structure, and the effect this has on the running time of DISTINCT.

2. Given a sequence $A = [A_1, \ldots, A_N]$ of N integers, the procedure LONGEST should return the length of the longest strictly increasing sequence within A. This sequence does not have to be contiguous, but the ordering of A should be preserved, and each element must be strictly less than the next. So, given $A = [56, -12, 4, 34, -3, 5, 35]$, the longest increasing sequence is either $[-12, 4, 34, 35]$ or $[-12, -3, 5, 35]$ or $[-12, 4, 5, 35]$ (there might be more than one longest sequence), and the length is 4.

   Write a procedure for LONGEST that runs in $O(N^2)$ time.

   To succeed in this task you will need to decompose the problem into subproblems. Start by considering the following. If you know the length of the longest increasing sequence within A that finishes with A_i, for all $i < j$, what is the length of the longest sequence that finishes with A_j?

Submission

Submit By: 1900, Monday 5th March 2018

Submit your *typed* answers to CATE in a file named `cw2.pdf` by the deadline above. Scanned copies of hand-written answers will not be accepted. Procedures can be written in either pseudocode or Java. If you are using LaTeX, then two suggested ways of typesetting procedures are to use a `verbatim` environment:

```
\begin{verbatim}
  Anything typed here will
    be output exactly as it
    is written
  in your source file
\end{verbatim}
```

or an `algorithmic` environment which creates this sort of output:

```
procedure Swap($A$, $i$, $j$)
    if $i \leq j$ then
        $temp = a_i$
        $a_i = a_j$
        $a_j = temp$
    end if
end procedure
```

See https://en.wikibooks.org/wiki/LaTeX/Algorithms for details.