7 Algorithms 1 (fms27)

Given a sequence `s` of items from a domain U with a total order $<$, we derive a *subsequence* of `s` by dropping zero or more items from `s`. A *descending* sequence is one in which every item except the first is strictly smaller than its predecessor.

[*Note:* In this question, we use 0-indexing and Python slice notation to indicate parts of a sequence, and so should you in your answer. For example, if `s` is `[A, B, C, D]`, then `s[0]` is `A`, `s[1:3]` is `[B, C]`, `s[:3]` is `[A, B, C]`, `s[3:]` is `D`, `s[4:]` is `[]` and `s[-1]` is `D`.]

We seek an efficient algorithm that, given a sequence `s` of length n, returns a descending subsequence `t` of maximal length.

(a) Discuss the following statement and prove that it is incorrect. [2 marks]

> This problem exhibits *optimal substructure*, meaning that the optimal solution may be expressed in terms of optimal solutions to subproblems, and may therefore be solved by dynamic programming. If we split sequence `s` into two parts, any optimal solution (a maximally long descending subsequence of `s`) must consist of an optimal solution to the first part concatenated with an optimal solution to the second part.

(b) Prove that the following statement is also incorrect. [3 marks]

> Any optimal solution for `s[:k]` must start with an optimal solution for `s[:k-1]`, possibly with `s[k-1]` appended to it if this combination still is a descending sequence.

(c) Give a correct recursive description and formula for the optimal solution in terms of optimal solutions to subproblems, with clear justification. The recursive description and formula must solve the given problem directly, as opposed to reducing it to another problem. [*Note:* The optimal solution is a subsequence, not the length of a subsequence.] [5 marks]

(d) Clearly explain how to solve the problem through the alternative approach of reducing it to one of the well-known dynamic programming problems studied in the lecture course, giving a correct recursive description and formula for the optimal solution to the well-known problem. [5 marks]

(e) Give clear pseudocode to solve the problem using the approach in Part (d), using recursive memoized top-down dynamic programming. [5 marks]