The Theory of Computation
Bernard M.E. Moret
Addison-Wesley-Longman, 1998

Solution Manual
Chapter 7, Problem 7.23

This problem is clearly in NP (we can guess not only the common subsequence, but also, for each string, the position of the subsequence's symbols within the string). We prove that the problem is NP-complete by transforming *Independent Set* to it. Given a graph, $G = (V, E)$, and a constant, $K \leq |V|$, we construct an instance of our problem as follows. There will be $|E| + 1$ strings, one for each edge to describe the graph and one to force a solution. The forcing string is just the list of the vertices, $v_1 v_2 \ldots v_n$. For each edge, $\{v_i, v_j\}$ (assume $i < j$), the corresponding string is

$$v_1 v_2 \ldots v_{i-1} v_{i+1} \ldots v_{j-1} v_j v_{j+1} \ldots v_n v_1 v_2 \ldots v_{i-1} v_i v_{i+1} \ldots v_{j-1} v_{j+1} \ldots v_n$$

K is the same for both problems.

If we have a solution to *Independent Set*, the vertices composing the solution form a solution subsequence: they clearly form a subsequence of the forcing string. They form a subsequence of each edge description: given the string describing edge $\{v_i, v_j\}$, one of three cases can occur: (i) neither v_i nor v_j is in the solution, in which case we can pick all solution vertices in the first part of the string; (ii) v_i is in the solution (and then v_j is obviously not in it), in which case we can pick all solution vertices up to v_i from the first part of the string and all remaining vertices from the second part (where all vertices except v_j are listed—and we do not need v_j); or (iii) v_j is in the solution (but not v_i), which is treated similarly to (ii).

The converse proof depends on the crucial observation that a common subsequence of length at least K cannot contain both vertices of an edge. This follows from the fact that the forcing string guarantees that vertices must appear in increasing subscript order and that the vertices of an edge, $\{v_i, v_j\}$ for $i < j$, can only be picked in reverse order in the string associated with that edge. Thus no two vertices in the subsequence are connected with edges: in other words, the vertices of the subsequence form an independent set of size at least K.

The transformation can also be based on vertices rather than edges: we keep the same forcing string and the same bound, but produce one string per vertex rather than one per edge. The string associated with vertex v_i has two parts: the first part lists in order all vertices except those connected to v_i by an edge; the second part lists all vertices except v_i. If there is an edge $\{v_i, v_j\}$, then the common substring cannot include both v_i and v_j: if $i < j$, then the string associated with v_j has v_i appearing *after* (and thus out of order) v_j while, conversely, $i > j$ has v_i appearing *before* (and thus out of order) v_j in the string associated with v_i. (In fact, the whole common substring will appear in the first part of the string if and only if the corresponding vertex is part of the independent set, in the second part of the string if and only if the corresponding vertex is not part of the independent set.)