You are given a string S of n characters and another string T of m characters such that $m \leq n$. A *subsequence* S' of S is any (not necessarily contiguous) sequence of characters within S. For example, a subsequence of $S = abcde$ is $S' = abd$. A *supersequence* of S' is any sequence of characters that contains S' as a subsequence. For example, $S = bacedf$ is a supersequence of $S' = bcd$. Similarly, a *superstring* is a contiguous supersequence.

You want to find the length of the longest subsequence of S that appears as a prefix of T. For example, if $S = abcdefgh$ and $T = bcdghf$, then your algorithm should return 5.

(a) Let T' be a prefix of T. Show that:

- If T' is a subsequence of S, then any substring of T' is also a subsequence of S.
- If T' is not a subsequence of S, then any superstring of T' is not a subsequence of S.

(b) For a given string A of n characters and another string B of m characters (with $m \leq n$), assume that there is an $O(f(n))$ algorithm that decides if B is a subsequence of A. Using this algorithm, describe an $O(f(n) \log m)$-time algorithm to compute the length of the longest subsequence of S that appears as a prefix of T.

Rubric.

- Your argument should prove both results. You can use any result you previously proved in your argument.
- You should justify why your algorithm is correct and why they run in the allocated time complexities (or faster!).
- This task will form part of the portfolio.
- Ensure that your argument is clear and keep reworking your solutions until your lab demonstrator is happy with your work.

Solution.

(a) If T' is a subsequence of S, the characters of T' appear in S in the same order as they appear in T'. Any substring of T' is a subset of these characters in the same order they appear in, and since they already appear in the correct order in S, any subset of them will also appear in the correct order in S. Therefore, any substring of T' is also a subsequence of S.

If T' is not a subsequence of S, the characters of T' appear in S in a different order as they appear in T'. Any superstring of T' is a superset of these characters with additional characters, the order is not changed, so any superset of them will remain in the incorrect order. Therefore, any superstring of T' is not a subsequence of S.

(b) We can perform a binary search to identify the length of the longest subsequence. We first start by taking some middle value i, say $i = \left\lfloor \frac{m}{2} \right\rfloor$. We then observe if the first i characters of B form a subsequence of A. If this is true, then we know that our length must be greater than or equal to this value of i, and so we reduce our searching space. If this is not the case, we know it must be less than i, so we reduce the searching space again - using the binary search method. Eventually, we will yield a value of i that stops the search, and this value is the value we return. Our binary search has a time complexity of $O(\log m)$, and so our algorithm will be $O\left(f(n) \log m\right)$.