

Biosequence Algorithms, Spring 2005

Exercise 6, Monday March 21, 2005, at 14.15–16 in MT2

1. We say that a string C is a *merge* (limitys) of strings A and B , if it is obtained by merging the characters of A and B in their original relative order together. For example, “KAAKELI” is a merge of strings “ALI” and “KAKE”. Present an algorithm that takes three strings $A[1 \dots n]$, $B[1 \dots m]$ and $C[1 \dots n + m]$, and tests in $O(nm)$ time whether C is a merge of A and B .
2. Let $S_1[1 \dots n]$ and $S_2[1 \dots m]$ be two strings with $n \leq m$. Their similarity (defined by a given scoring matrix s) can be computed by maintaining only two columns of the dynamic programming table. Present the algorithm. Could the similarity be computed efficiently using even less space?
3. (Gusfield, Ex. 11.13) In a dynamic programming table for *edit distance*, are the entries along a row necessarily nondecreasing? What about down a column or a diagonal of the table? How about in a dynamic programming table for string *similarity*? (Assume negative scores for mismatches and spaces, and positive scores for matches.)
4. Consider locating approximate occurrences of the pattern $P = \text{“aino”}$ in the text “vaitonainen”. Let us score matches by 1, and mismatches and insertions/deletions by -1 . Present the dynamic programming table. Explain how the approximate occurrences of P are found, if we require their similarity with the pattern to be at least 2.
5. (\sim Gusfield, Ex. 11.11) The number of optimal alignments between two strings can grow exponentially with respect to the length of the strings. Thus it is not possible to enumerate all of them in worst-case polynomial time. Present a dynamic programming algorithm that computes the *number* of optimal alignments between given strings $S_1[1 \dots n]$ and $S_2[1 \dots m]$ in $O(nm)$ time (either for edit distance or similarity).
6. The course feedback form is available at the course homepage. Fill in and return the form.