Efficient algorithms for finding a longest common increasing subsequence.  


Summary: We study the problem of finding a Longest Common Increasing Subsequence (LCIS) of multiple sequences of numbers. The LCIS problem is a fundamental issue in various application areas, including the whole genome alignment. In this paper we give an efficient algorithm to find the LCIS of two sequences in $O(\min(r \log \ell, n\ell + r) \log \log n + \text{Sort}(n))$ time where $n$ is the length of each sequence and $r$ is the number of ordered pairs of positions at which the two sequences match, $\ell$ is the length of the LCIS, and $\text{Sort}(n)$ is the time to sort $n$ numbers. For $m$ sequences where $m \geq 3$, we find the LCIS in $O(\min(mr^2, r \log \log m \ell + m \cdot \text{Sort}(n)))$ time where $r$ is the total number of $m$-tuples of positions at which the $m$ sequences match. The previous results find the LCIS of two sequences in $O(n^2)$ and $O(n\ell \log \log n + \text{Sort}(n))$ time. Our algorithm is faster when $r$ is relatively small, e.g., for $r < \min(n^2 / (\log \ell \log \log n), n\ell / \log \ell)$.

MSC:

68W05 Nonnumerical algorithms  
68W40 Analysis of algorithms  
92D20 Protein sequences, DNA sequences

Keywords:  
Design and analysis of algorithms; Longest common increasing subsequence

Software:

MUMMER

References:


This reference list is based on information provided by the publisher or from digital mathematics libraries. Its items are heuristically matched to zbMATH identifiers and may contain data conversion errors. It attempts to reflect the references listed in the original paper as accurately as possible without claiming the completeness or perfect precision of the matching.