Simple and Efficient Pattern Matching Algorithms for Biological Sequences

Abstract:

Storing and processing of large DNA sequences has always been a major problem due to increasing volume of DNA sequence data. In computational biology one often needs to look up the occurrence of some pattern P in a text T. In different phases pattern matching is a very practical operation. For example, pattern matching enables users to find the locations of particular DNA subsequences in a database. To perform faster searches, pattern matching algorithms are needed. The present paper introduces three pattern matching algorithms raise performance by utilizing word processing (in place of the character processing presented in previous works) and also by searching the least frequent word of the pattern in the sequence.

Introduction:

In the pattern matching problem, sequence is scanned to detect the positions of a P in the T. It is imperative that this kind of problem be addressed because of its applications. The pattern matching problem arises in the different scopes of computational bioinformatics, which include the basic local alignment search, biomarker discovery, sequence alignment, and homologous series detection. It is essential to identify the positions of various patterns in databases .In medicine the knowledge of DNA sequences may be applied when exploring possible disease or diagnoses. DNA sequence analysis may also be used to compare a gene to similar genes from the same or different organisms, as well as to predict its function. The functionality of a newly discovered DNA sequence can be predicted by comparing it to known DNA sequences. The development of algorithms is still required. This is necessary because many current algorithms may not be well scalable for databases or large DNA sequences due to high-computational costs. The proposed algorithms are divided into two phases: preprocessing and matching. In the preprocessing phase, the potential intervals of the text to be matched with the pattern are recognized .These intervals are called windows. During the matching phase, the windows are carefully scanned in order to be matched with the pattern. The less windows found in the preprocessing phase, the less time is spent in the matching phase checking the windows. The present work's primary aim is to decrease the number of windows. Some algorithms scan the text separately to discover the first and last character of the pattern for diagnosing the windows. The first suggested algorithm in the current analysis, finds the windows by considering both the first and last character of the pattern at the same time. Almost all processors nowadays have a computational length of 32 or 64 bits in each execution period. The second algorithm to conduct word-based comparisons. The word processing is performed by utilizing the processing power of the processor.so the number of detected windows decreases and speeds up the comparisons. The third algorithm searches the text for a low-frequency word of the pattern. This technique advances the algorithm's efficiency by decreasing the number of windows.