Proc. Natl. Acad. Sci. USA
Vol. 80, pp. 726–730, February 1983
Biochemistry

Rapid similarity searches of nucleic acid and protein data banks

(global homology/optimal alignment)

W. J. WILBUR AND DAVID J. LIPMAN

Mathematical Research Branch, National Institute of Arthritis, Diabetes, and Digestive and Kidney Diseases, National Institutes of Health, Building 31 Room 4B-54, Bethesda, Maryland 20205

Communicated by Maxine Singer, November 8, 1982

ABSTRACT **With the development of large data banks of protein and nucleic acid sequences, the need for efficient methods of searching such banks for sequences similar to a given sequence has become evident. We present an algorithm for the global comparison of sequences based on matching *k*-tuples of sequence elements for a fixed *k*. The method results in substantial reduction in the time required to search a data bank when compared with prior techniques of similarity analysis, with minimal loss in sensitivity. The algorithm has also been adapted, in a separate implementation, to produce rigorous sequence alignments. Currently, using the DEC KL-10 system, we can compare all sequences in the entire Protein Data Bank of the National Biomedical Research Foundation with a 350-residue query sequence in less than 3 min and carry out a similar analysis with a 500-base query sequence against all eukaryotic sequences in the Los Alamos Nucleic Acid Data Base in less than 2 min.**

As the number of protein molecules and nucleic acid fragments for which the sequences have been determined has grown into the thousands (the total number of nucleotides so analyzed is now more than one million), it has become clear that a rapid method for carrying out similarity searches would be useful. Such a method should allow economical study of large data banks in search of related sequences that would then be subjected to more definitive analysis.

Currently, there are several different methods in use for analyzing the similarity of two sequences. For the purpose of global comparison (considering both complete sequences), there are the methods of Fitch (1) as implemented by Dayhoff (2), of Needleman and Wunsch (3) and Sellers (4) [see Smith *et al.* (5) for proof of the equivalence of these two algorithms], and of Sankoff (6). Given a set of scoring rules, such as +1 for a base match and −3 for a gap, a Needleman–Wunsch type algorithm considers all possible alignments, including gaps, and will find an optimal alignment under the scoring rules. All of these methods require computer time on the order of $N \times M$, where N and M are the lengths of the sequences compared. Local search methods (a search for similar fragments of two sequences) have been proposed by Korn *et al.* (7), Sellers (8), Smith and Waterman (9), and Goad and Kanehisa (10). These methods are under the same time constraints as the global methods already noted. Dayhoff (2) has implemented an algorithm that compares a 25-residue test subsequence from one peptide chain with all possible 25-residue subsequences from another, not allowing gaps. If all test subsequences are used, the time is again of the order of $N \times M$ but, in many instances, reasonable choices for test subsequences can improve the time without significant sacrifice in the accuracy of results.

All of the search techniques mentioned above become computationally intensive and quite expensive when applied to large banks of sequences. We shall describe here a global algorithm for comparing two nucleic acid or two amino acid sequences. This algorithm involves the construction of an optimal alignment that is useful in its own right. The algorithm also requires a computation time on the order of $N \times M$, where N and M are the lengths of the sequences being compared, but, for given sequences, the computation is many times faster than the above-mentioned methods. Results obtained by the method and its limitations and advantages are discussed.

METHODS

Computational Methods and Data Sources. All computing was done on the DEC KL-10 computer facility at the National Institutes of Health. The programs are written in DEC-10 Pascal.* The graphs shown were generated by using the MLAB program facility at the National Institutes of Health. All sequences were taken from the Los Alamos Sequence Data Base and the National Biomedical Research Foundation Data Bank.

The Algorithm. We shall here describe how two sequences, S1 and S2, of lengths $N1$ and $N2$, respectively, are to be compared. As motivation, it is useful to think in terms of the dot matrix comparison of S1 and S2 (11) in which the beginnings of both sequences are placed to the upper left of the matrix and one sequence is positioned horizontally and the other is positioned vertically. The diagonals running downward from left to right in the dot matrix illustrate the degree of similarity that would be found by a simple sliding comparison with the different possible choices of alignment register. Frequently, one can look at the dot matrix comparison and immediately see certain diagonals that appear to have a number of points above background and, therefore, indicate a level of similarity for the two sequences in certain regions. It is generally true that these significant diagonals are still clearly visible when the dot matrix is filtered to only show matches of length k or greater, where k is a small positive integer. For this reason, our attention will be directed to such k-tuples.

The first step in the algorithm is the location of all the k-tuple matches between the sequences S1 and S2. In precise terms, a k-tuple match consists of two k-tuples—S1(i),S1(i + 1), . . .,S1(i + k − 1) from S1 and S2(j),S2(j + 1), . . .,S2 × (j + k − 1) from S2—that are identical. If there are p elements in the alphabet from which the sequences are made, then there are p^k possible different k-tuples. To locate all k-tuple matches, we follow a method described by Dumas and Ninio (12). We have chosen a simple method (there are many possible) of converting any k-tuple into an integer between 1 and p^k. Then, a one-dimensional array, C, of length p^k and consisting of pointers set initially to nil is used. In a single pass through S1, each integer position i is added to a list constructed at $C(ic)$, where ic is the coded form of the k-tuple beginning at i in S1.

The publication costs of this article were defrayed in part by page charge payment. This article must therefore be hereby marked "*advertisement*" in accordance with 18 U. S. C. §1734 solely to indicate this fact.

* The programs described in this paper available from the authors.

Then, in a single pass through S2, each integer position j marks the beginning of k-tuple t_j that codes to jc and, from the list marked by the pointer $C(jc)$, one reads off the positions of all k-tuples in S1 that match with t_j. The speed with which the k-tuple matches between S1 and S2 can be located by this method is dependent on two factors. First, if in carrying out the procedure, each matching k-tuple occurred only once in each sequence, then the time would be of the order of the longer of the two sequences. Second, if a particular k-tuple occurs R times in one sequence and T times in the other, then with all combinations, there are $R \times T$ matches of the two sequences produced by this k-tuple. For this reason, if very long sequences ($\approx$100,000) are compared, the product $R \times T$ may become large and cause the comparison time to behave as a product. For the sequence lengths encountered in practice and the k values used, $R \times T$ is generally a small number with an average not greater than 4 and this results in a computation time on the order of the length of the longer sequence.

For a particular choice of k-tuple size, the algorithm does not consider all k-tuple matches that are found between two sequences in making a comparison but rather those that occur in a certain region that we term window space. Window space is defined as a specified region around the most significant diagonals in the comparison. To be more precise, we represent a diagonal by d_m, where m is any integer between $1 - N2$ and $N1 - 1$ and d_m will stand for all pairs of elements S1(i),S2(j) with $m = i - j$. A k-tuple match is said to lie on d_m if and only if its matched pairs of sequence elements are part of d_m. Clearly, every k-tuple match lies on some diagonal. If d_m and d_n are diagonals, the *distance* between them will be taken as $|n - m|$. A diagonal will be considered *significant* if it has a number of k-tuple matches a certain number of standard deviations above the mean for all diagonals with k-tuple matches. Other definitions of the term significant diagonal are possible but this has been the most generally useful. The definition of window space requires a nonnegative integer parameter w and window space is the collection of all diagonals within a distance w of any significant diagonal. The k-tuple matches that occur in window space are just those that occur on some diagonal composing window space.

When k and w have been fixed and the significant diagonals have been determined, window space is defined. The algorithm then produces a Needleman–Wunsch (3) type alignment that is optimal under certain scoring rules, which we now specify. A score of +1 is given to each pair of sequence elements that occur adjacent to each other in a given alignment provided they are part of a k-tuple match produced by the alignment and this k-tuple match occurs in window space. A score of $-g$ is given to each gap that occurs in either sequence between positively scored matches in an alignment. Here the parameter g may be set to any desired value but it is the same for all gaps and independent of their length. Under these rules, the algorithm produces one of the alignments with the greatest possible score (there may be more than one such alignment). Briefly, the algorithm does not fill out an $N1 \times N2$ matrix as is done in the more standard Needleman–Wunsch (3) or Sellers (4) techniques but attaches scores just to the k-tuple matches that occur in window space. A more detailed discussion of the algorithm will appear elsewhere.

To illustrate the function of the algorithm, Fig. 1 shows the alignment produced for two nucleotide fragments when $k = 3$, $g = 7$, and w is chosen large enough so that window space includes all 3-tuple matches. Those base matches that occur as part of a 3-tuple match are marked by stars and all other matches are marked by colons. Since there are 55 matches marked by stars and two gaps (−7 each), the score for this alignment is 41. The algorithm ensures that there is no alignment of these two fragments that produces a higher score than 41.

The Search. When $k > 1$ and w is of modest size, the alignment algorithm attains such speed as to provide a useful and economical method of searching large data banks of sequences. Given a query sequence and a choice of parameters, k, w, and g, optimal alignment scores are produced for comparison of the query sequence with each sequence in the data bank. This produces a list of raw scores, one for each bank sequence. In general and for random sequence comparisons, the score is dependent on the lengths of the query and bank sequences. The raw score (R_{score}) is converted to what we term the factored score by the normalizing transformation

$$F_{score} = (R_{score} - M)/SD, \quad [1]$$

where M is the mean of the distribution. This effectively removes dependence on the length of the query sequence. By repeated querying of the bank with randomly generated sequences of different lengths and the use of curve fitting, an empirical correction of the factored score is made dependent on the length of the bank sequence that produced it. The corrected scores are then ranked from the highest to the lowest to produce an ordering of the bank sequences beginning with those most similar (highest corrected score) to the query sequence.

It would clearly be desirable to assign P values to the corrected scores produced in a search; however, the assignment of a meaningful P value to a similarity score produced by a Needleman–Wunsch type method is a difficult theoretical problem. Because of the global optimization involved, reasonably simple combinatorial methods such as those used in the local method of Goad and Kanehisa (10) are not applicable. Analytical results obtained by Steele (13) are inadequate to assign P values and deal only with the case when the gap penalty is zero.

An approach that has been used in an attempt to define the

```
X 30         40         50         60         70         80         90
AACGTCAAGGCCGCCTGGGGTAAGGTCGGCGCGCACGCTGGCGAGTATGGTGCGGAGGCCCTGGAGAGG
  ***      :   *****  *****    : :  : :  :        *****  **********  ***
TGCTGTCTCTTGCCTGTGGGGAAAGGTGAACTCCGATGAAGT------TGGTGGTGAGGCCCTGGGCAGG
X          40         50         60         70                 80         90

  100        110        120        130        140        150        X
ATGTTCCTGTCCTTCCCCACCACCAAGACCTACTTCCCGCACTTC---GACCTGAGCCACGGCTCTG
 :: :  :   :: ***    *** :::  *****       ***    *****   :   : *****
CTGCTGGTTGTCTACCCTTGGACCCAGCGGTACTTTGATAGCTTTGGAGACCTATCCTCTGCCTCTGCTA
      100        110        120        130        140        150        160X
```

FIG. 1. Example of the actual program output for implementation of the alignment algorithm with $k = 3$, $g = 7$, and w large enough to include all diagonals in window space (the same alignment is produced with $w = 20$). (The numbering shown is based on the larger sequences from which the DNA fragments were taken.) X, point at which the original DNA has been cut, here only at the ends; *, match that is part of a 3-tuple match; :, all other matches. Gaps that occur are automatically placed adjacent to the 3-tuple match occurring to their right (toward the 3′ ends of the sequences).

significance of a Needleman–Wunsch type alignment is Monte Carlo simulation. There are two problems with its use. (*i*) In general, the accuracy of such a method depends on the ability to simulate actual biological sequences in a meaningful way making use of all the important statistical properties of such sequences. To what extent the generally used random shuffle is adequate for this purpose is not known. (*ii*) A useful Monte Carlo simulation would require a relatively large number of trials to assign a *P* value to the highest scores in a search and this is inconsistent with an economical and generally useful search technique. The enumerated difficulties have led us to the following method of statistical analysis. We assume that the length correction has been made. Then, for any given query sequence, it is reasonable to assume that most of the sequences in the data bank are randomly related to it. Thus, with the possible removal of a few outstanding scores, a random distribution of scores is obtained from any given search. The corrected score (C_{score}) in this distribution is then converted to a normalized score by the transformation

$$z = (C_{score} - M)/SD. \quad [2]$$

We then let Mv be the smallest z among all the transformed scores and perform the transformation

$$z' = \ln(z - Mv + 1). \quad [3]$$

It is found that the z' values are approximately normally distributed for both protein and nucleic acid data banks. Fig. 2 illustrates the similarity between the distributions obtained from random and real sequences, confirming our hypothesis that even for real sequences, the distribution is basically random. In practice, we use the distribution of transformed scores (by Eqs. 2 and 3) to assign to each bank sequence a significance value that is the number of standard deviations its transformed score is above the mean in the distribution. If the distribution were normal, such a significance value could be readily converted to a *P* value; however, the approximate normality of the distribution only allows us at this point to interpret it as an empirical guide to the significance of similarity between the query and bank sequences. The statistical analysis we have given is quite satisfactory for sequences of approximately average composition relative to the data bank; however, large deviations in a query sequence from such a composition may lead to the following: (*i*) a lower average score against the bank and (*ii*) a relatively high score for those sequences in the bank having a similarly biased composition. This will result in misleading significance levels.

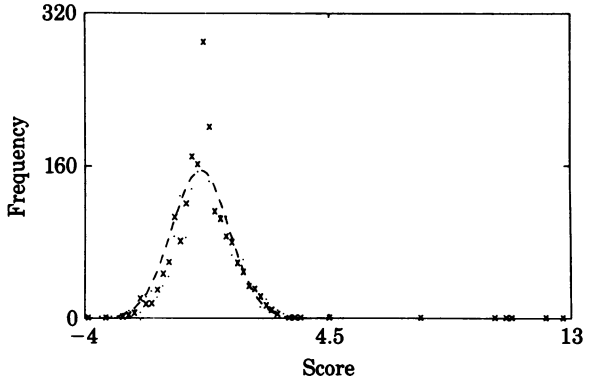

FIG. 2. Similarity between distributions obtained from random and real sequences. X, Frequency distribution of transformed scores obtained in a global search of the National Biomedical Research Foundation Protein Data Bank using human growth hormone as query sequence (scores in the tail to the right represent sequences having significant similarity to the query sequence); ·, distribution obtained after a random shuffle of the human growth hormone; – –, normal curve having the same area, mean, and SD as the search-generated frequency distributions.

RESULTS AND DISCUSSION

The algorithm we have described produces alignments closely related to the alignments produced by the Needleman–Wunsch method as extended by Smith *et al.* (5). In fact, with $k = 1$ and w chosen large enough so that window space includes all diagonals, our algorithm is equivalent to a Needleman–Wunsch algorithm with the scoring parameters set the same. When $k > 1$ and w is relatively small, the algorithm still produces alignments that provide a good approximation to a Needleman–Wunsch alignment.

For the purpose of comparison, 28 pairs of nucleic acid sequences having a range of similarities from the random to the closely related were selected from the Los Alamos Nucleic Acid Sequence Data Bank. Alignments were produced by our method with $k = 4$ (as is currently implemented in the nucleic acid search program), a window of 10, and a gap penalty of six. Gaps between k-tuples were arbitrarily placed at the 3′ end of the intervals between the k-tuples. Alignments were also produced by the Needleman–Wunsch method as implemented in the Los Alamos sequence analysis package with matches, mismatches, and gaps receiving scores of 1, −2, and −3, respectively. For each pair of sequences, the actual number of matching bases as a percentage of the length of the shorter sequence was calculated for the two methods of alignment. The results are shown in Fig. 3. Notably, all points lie on or above the diagonal because the Needleman–Wunsch algorithm can optimize the placement of gaps between matching k-tuples and can optimize to a finer level of detail. Nevertheless, the agreement between the two methods is excellent, even with $k = 4$.

It is evident that the parameter w is an important factor in determining the size of window space in a comparison and thus in influencing the time required in a calculation. It is important to know how sensitive scores are to w, as this gives some indication of the gain that may be expected by optimizing in a larger window space. To test the effect of different w values, 100 pairs of randomly generated nucleic acid sequences, each

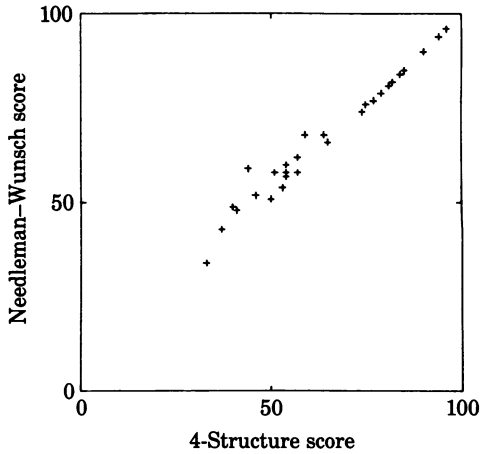

FIG. 3. Twenty-eight pairs of nucleic acid sequences were analyzed to produce an alignment by their 4-tuples using the global algorithm and by the Needleman–Wunsch algorithm as implemented in the Los Alamos sequence analysis package (for parameter choices used, see text). The actual number of matches produced in the alignment as a percentage of the length of the shorter sequence for the two methods is plotted as a point for each comparison.

sequence 500 bases long, were compared by the algorithm with $k = 4$ and $g = 7$. The comparison was done with a w of 5 and the mean raw score was 33.25 with a standard deviation of 3.75. With a larger window, raw scores cannot decrease. A repeat comparison with $w = 20$ produced an average increase in the raw score of 0.69 with a standard deviation of 1.13. When $w = 40$ was used, the average increase in raw score over the case $w = 20$ was 0.04 with a standard deviation of 0.24. This indicates that minor changes will occur in moving from a w of 5 to one of 20 but, above $w = 20$, very little improvement in score and, consequently, in alignment can be expected.

The dependence of the algorithmic computation time on the parameters k and w and on the lengths of the sequences is especially important when considering searches of large data banks of sequences. These dependencies are illustrated in Figs. 4 and 5. The average computation time for comparison of simulated nucleic acid sequences, each 500 bases long, is shown as a function of k-tuple size for three different choices of w in Fig. 4. The smaller window sizes significantly reduce the dependence of computation time on k. In addition, it is evident that a w of 20 is a relatively optimum value for, as noted above, little improvement in alignments can be expected by increasing w and little improvement in speed can be made by decreasing it. The dependence of computation time on length is shown in Fig. 5. Each point on the graph represents an average time to compare two randomly generated nucleic acid sequences of the stated length with $k = 4$, $w = 20$, and $g = 7$. It is clear that the time is of the order of N^2, where N is sequence length. Such a time dependence is the general rule for Needleman–Wunsch type alignment algorithms. For comparison, running our algorithm on a DEC KL-10 system with $k = 4$ and $w = 20$ will produce an alignment of two nucleic acid sequences of length 500 in 0.4 sec while running a full Needleman–Wunsch or Sellers alignment on a comparable system requires approximately 1 min. Direct comparison of core requirements for the two methods is not meaningful because, in general, core requirements can be decreased in exchange for increases in computation time. What could manifest itself as a core differential does, however, manifest itself as a part of the time differential seen for the two methods.

The search method we have presented is best illustrated by

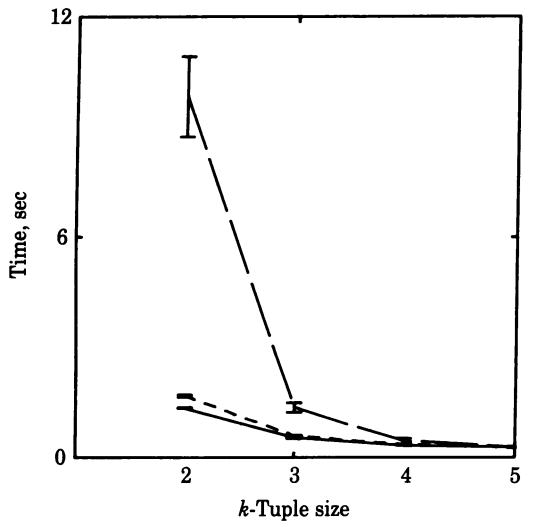

FIG. 4. Dependence of computation time on values of k and w. Each computation time shown is the mean ± 1 SD taken over a set of 50 comparisons of randomly generated nucleic acid sequences of length 500. Each curve has points for values of k from 2 to 5. The three curves represent three different values of the window parameter w: —, $w = 5$; – –, $w = 20$; — —, $w = 100$.

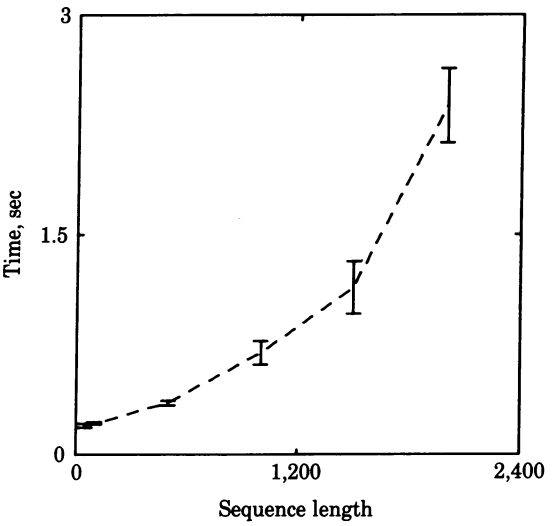

FIG. 5. Dependence of computation time on length of sequence. Each computation time shown is the mean ± 1 SD taken over a set of 50 comparisons of randomly generated nucleic acid sequences. The different lengths used for the points plotted were 50, 100, 500, 1,000, 1,500, and 2,000 bases. The parameters were fixed at $k = 4$, $w = 20$, and $g = 7$.

the results of test cases. For this purpose, we have searched the National Biomedical Research Foundation Protein Data Bank using the catalytic chain of cAMP-dependent protein kinase from bovine cardiac muscle. [Barker and Dayhoff (14) have reported a significant similarity between this sequence and the transforming protein sequences translated from the Rous avian and Moloney murine sarcoma virus *src* genes.] We have also searched the same data bank using the human somatotropin precursor sequence to see whether the relatively distant relationship between the somatotropin and prolactin sequences would be detected. The results of searches for these test cases are given in Table 1 (for the somatotropin query, the first five sequences were the five other growth hormones in the bank). These examples show that the search technique we have described is effective and illustrate the consequences of the different parameter choices made. Other test searches we have made include the following. When the mouse β major globin protein was used as a query, the search found all β globin and β-globin-like sequences, followed immediately by all α-globins and myoglobins, and finally by leghemoglobins. When the large

Table 1. Results of searches with various query sequences and parameters: Relative rank in protein bank

Query	Comparison	Gap penalty 1	2	3
Bovine cAMP-dependent protein kinase	Moloney murine virus transforming protein	2	1	2
	Rous sarcoma virus transforming protein	2	6	>40
Human somatotropin precursor*	Human prolactin precursor	6	6	6
	Pig prolactin	8	7	7
	Rat prolactin precursor	7	8	23
	Sheep and bovine prolactin	10	11	19

In some instances, the listed sequence tied with several other sequences in the bank at the indicated ranking. The value $w = 10$ was used.

* The first five sequences in all ratings are the other growth hormones in the bank.

intron of the mouse β^{maj} globin gene was used as a query sequence, the top three sequences for eukaryotes were the analogous introns from other β-globin or β-globin-like genes. Thus, the technique appears able to detect relatively weak similarities. As with many other algorithms, there is some ambiguity in the parameter settings for best results, as shown by the results of Table 1. [For cogent remarks on this subject, see Smith *et al.* (5).]

Because our algorithm optimizes within the constraints of k and w, one cannot obtain the resolution that can be expected from the full Needleman–Wunsch or Sellers type algorithm. Although this problem can be progressively alleviated by decreasing k and increasing w, the results shown in Fig. 3 indicate that significant savings in time can be obtained at very little cost in the quality of the alignments. In fact, it may be fairly asked whether the more optimal alignment of a few relatively isolated sequence elements (not parts of k-tuple matches) that can be obtained by the full Needleman–Wunsch alignment over our method really gives a more accurate picture of biological truth. To this question, we do not know the answer.

The great advantage of the method we have presented is its speed. Currently, using the DEC KL-10 system, we are able to search the National Biomedical Foundation Protein Data Bank comparing all entries with a 350-residue query sequence in less than 3 min. On the same system, all eukaryotic sequences in the Los Alamos Nucleic Acid Data Base can be compared with a query sequence 500 bases long in less than 2 min. The significance of the results of a search could be assessed by more definitive calculations. Of greater importance, all results must be assessed in terms of biological context until a closer correlation between biology and the models by which we attempt to understand biology has been developed.

The algorithm described here has been adapted to produce local best alignments after the manner of Smith and Waterman (9). Searches based on local best alignments have proven more useful than global searches in dealing with the inhomogeneity of nucleic acids.

We would like to thank Dr. T. Smith for supplying us with a specially edited tape of the eukaryotic nucleic acid sequences and Dr. H. Saroff for a stripped version of the Protein Data Bank. We would also like to thank Dr. P. Haeberli for a helpful discussion regarding the efficient location of k-tuple matches.

1. Fitch, W. M. (1966) *J. Mol. Biol.* **16**, 9–16.
2. Dayhoff, M. O. (1979) in *Atlas of Protein Sequence and Structure*, ed. Dayhoff, M. O. (National Biomedical Research Foundation, Washington, DC), Vol. 5, Suppl. 3, pp. 1–8.
3. Needleman, S. B. & Wunsch, C. D. (1970) *J. Mol. Biol.* **48**, 443–453.
4. Sellers, P. H. (1974) *SIAM J. Appl. Math.* **26**, 787–793.
5. Smith, T. F., Waterman, M. S. & Fitch, W. M. (1981) *J. Mol. Evol.* **18**, 38–46.
6. Sankoff, D. (1972) *Proc. Natl. Acad. Sci. USA* **69**, 4–6.
7. Korn, L. J., Queen, C. L. & Wegman, M. N. (1977) *Proc. Natl. Acad. Sci. USA* **74**, 4401–4405.
8. Sellers, P. H. (1979) *Proc. Natl. Acad. Sci. USA* **76**, 3041.
9. Smith, T. F. & Waterman, M. S. (1981) *J. Mol. Biol.* **147**, 195–197.
10. Goad, W. B. & Kanehisa, M. I. (1982) *Nucleic Acids Res.* **10**, 247–263.
11. Maizel, J. & Lenk, R. (1981) *Proc. Natl. Acad. Sci. USA* **78**, 7665–7669.
12. Dumas, J. P. & Nimo, J. (1982) *Nucleic Acids Res.* **10**, 197–206.
13. Steele, J. M. (1982) *SIAM J. Appl. Math.* **42**, 731–737.
14. Barker, W. C. & Dayhoff, M. O. (1982) *Proc. Natl. Acad. Sci. USA* **79**, 2836–2839.