Volume 12 Number 1 1984 Nucleic Acids Research

A comprehensive sequence analysis program for the IBM personal computer

Cary Queen
Laboratory of Biochemistry, National Cancer Institute, Bethesda, MD 20205, and

Laurence Jay Korn
Department of Genetics, School of Medicine, Stanford University, Stanford, CA 94305, USA

Received 22 August 1983

ABSTRACT

We have developed a versatile program for the analysis of nucleic acid and protein sequences on the IBM Personal Computer. The program is interactive and self-instructing. It contains all the features generally found in sequence analysis programs on large computers, including extensive homology routines, as well as new procedures for the entry of sequence data. The program contains facilities to store and utilize the entire Nucleic Acid Sequence Data Bank. We have devised a new algorithm to find restriction enzyme sites, which allows our microcomputer program to find all sites on a small plasmid for 100 different enzymes in 1 to 2 minutes.

INTRODUCTION

The analysis of nucleic acid and protein sequences can be greatly facilitated by the use of a computer. We have previously presented (1) and improved (2,3) one of the more extensive computer programs for sequence analysis. This program has been used, for example, as an aid in the study of origins of DNA replication (4), satellite DNA (5), bacterial transformation sequences (6), small nuclear RNAs (7), splicing (8), transposable elements (9), SV40 variants (10), vaccinia (11) and Epstein-Barr (12) viruses, retroviruses (13-15), transcription termination sites (16), and genes for tRNA (17), ribosomal proteins (18), 5S ribosomal RNA (19), large ribosomal RNAs (20,21), globins (22,23), immunoglobulins (24,25), hormone precursors (26), dihydrofolate reductase (27), chorion proteins (28), and the tryptophan operon (29,30). However, this program was designed to be used on a large (mainframe) computer in a non-interactive environment. Hence the user must have significant experience in order to utilize the program most effectively. Moreover, use of the program on a central mainframe computer may involve continuing expense.

The advent of microcomputers has made it possible to create sequence analysis programs that can be run on a readily accessible laboratory computer. The first such programs (e.g., 31-34) were written on small, 8-bit

computers. The simplicity and very limited memory capacity of these machines necessarily limits the scope and performance of the programs, and prevents the programs from offering much instructional help to the user. Recently, powerful 16-bit microcomputers with a large amount of memory space, such as the IBM Personal Computer (IBM PC), have been introduced. The availability of these machines creates the potential to have a fully comprehensive, very fast, self-instructing sequence analysis program in the laboratory.

Based on our previous programs (1-3), we have written a sequence analysis program specifically for the IBM PC. Because this computer is rapidly becoming an industrial and laboratory standard, our choice will not significantly reduce the applicability of the program, while allowing the use of machine-specific hardware and software features to enhance program performance. In writing the program, we sought to achieve four design goals:

(1) The program should be interactive, self-instructing, and self-contained. A researcher should be able to use it without knowledge of computers or operating systems.

(2) The program should incorporate the most useful procedures of our previous programs, as well as many useful procedures devised by others. Special procedures to facilitate rapid entry and collating of sequence data are desirable.

(3) The program should be able to analyze long sequences such as bacteriophage lambda. It should have facilities to store and manipulate the National Nucleic Acid Sequence Data Bank.

(4) The program must execute quickly. Sequences of small plasmids should be analyzed almost immediately, and comparisons with the whole Data Bank done in a practicable amount of time.

In order to achieve the last of these goals, it was necessary to develop a computer algorithm that may be of general interest.

MATERIALS AND METHODS

Materials. The program was written on and for an IBM Personal Computer with the monochrome screen, two double-sided disk drives (Tandon TM 100-2), and a graphics printer (Epson MX-80). An AST Research Megaplus Board supplied the required 512K memory, clock-calendar and asynchronous port. Most efficient use of the National Nucleic Acid Sequence Data Bank requires a Winchester hard disk (e.g., Davong 10 Mb), and direct data entry from autoradiograms requires the Grafbar Sonic Digitizer (Science Accessories

Corp., Model GP-7). Approximate costs are $3000 for the computer system with disk drives and printer, $800 for the memory board, $1600 for the hard disk, and $1000 for the digitizer.

General methods. The program was largely written in IBM Pascal, with several sections written in assembly language to improve speed or allow access to hardware features. Extensive use was made of the ability of IBM Pascal to access memory directly, in order to circumvent its 64K stack size limitation. The memory-mapped architecture of the monochrome display was used to produce fast screen changes.

Restriction site location algorithm. To rapidly locate all restriction sites for many different enzymes, we devised a new algorithm. The idea of the algorithm is to prepare a listing in alphabetical order of all sets of 6 consecutive nucleotides (6-mers) that occur in the sequence. All locations for most enzyme sites may then be found by simply looking them up in the alphabetical table. In order to do this conveniently, the first step of the algorithm changes each nucleotide in the sequence to a number from 0-3 (A's to 0, C's to 1, etc.). In the second step, each set of 6 consecutive nucleotide numbers N(I-5) ... N(I) is assigned a number by the formula $X(I) = N(I) + 4 \cdot N(I-1) + 16 \cdot N(I-2) + \ldots + 1024 \cdot N(I-5)$. This can be done most rapidly using the recursive formula $X(I) = 4 \cdot (X(I-1) - 1024 \cdot N(I-6)) + N(I)$. (In effect the algorithm views each 6-mer as a number in the base 4, so alphabetical order of the 6-mers corresponds to ascending order of the numbers.) In the third step, the number of X(I)'s having each of the 4096 possible values are counted and recorded.

Now denoting the number of X(I)'s having the value V by C(V), the numbers C(V) are used to quickly prepare the alphabetical listing of 6-mers. More precisely, a list of the starting nucleotide positions I of all 6-mers is prepared according to ascending value of the corresponding X(I)'s, as follows: A region of memory large enough to contain a list of all nucleotide positions is set aside. This region is divided into 4096 blocks, one for each possible value of X(I). The key point here is that the size and hence positions of the blocks can be determined in advance from the numbers C(V). Namely, the block containing I's such that X(I) = 0 must have room for C(0) numbers and hence extends from memory positions 1 to C(0), the block with I's such that X(I) = 1 must have room for C(1) numbers and hence extends from memory position C(0) + 1 to C(0) + C(1), etc. As its fourth step, the algorithm simply goes through the original sequential list of X(I)'s (representing the 6-mers) and places each I in the memory block set

aside for nucleotide positions with that value of X(I). (Pointers are kept to indicate the extent to which each block is filled, and thus to show the precise memory position available for the next I). A record of the position of each memory block is also kept in memory.

Having established this list of nucleotide positions I according to ascending value of X(I) (i.e., according to alphabetical order of 6-mers), all occurrences of any restriction site that consists of 6 specific nucleotides can be located almost instantly. The restriction site, which is itself a 6-mer, is simply converted into a number X by the formula used above; then the memory block containing I's such that X(I) = X gives a complete list of the nucleotide positions where the site appears. The locations of a site having less than 6 nucleotides can be found in several contiguous memory blocks, corresponding to all possible 6-mer extensions of the site. And the program locates sites that are not self-complementary or that have ambiguous nucleotides by generating all possible specific forms of the site and finding their locations from the list. Minor modifications of the algorithm allow it to handle ambiguous nucleotides in the sequence itself, by keeping a separate list of all 6-mers containing them. To locate restriction sites extending over more than 6 nucleotides, the program must search through the sequence directly, but there are only a few such sites.

RESULTS AND DISCUSSION

General features of the program. The program is divided into two sections, one containing all functions for the entry of sequence data, and the other containing all procedures for analysis of the sequence data. Each section is about 180K bytes long (compiled from about 3000 lines of code) and is contained on a separate disk. Upon inserting one of the disks in the drive and typing "run", a menu of possible functions appears (Fig. 1). After the user types the first letter of the desired function, either the program asks for information relevant to that function, or a secondary menu appears. In the latter case, the user again types a letter to choose a specific function.

The program communicates with the user by means of prompts (requests that appear on the screen for information or instructions). The prompts are short and written in straightforward English, for example, "Please give the name of the sequence to be entered." Some of the prompts clearly require a yes or no answer; the user answers these by typing Y or N. Other prompts require a choice of several alternatives. The choices are then

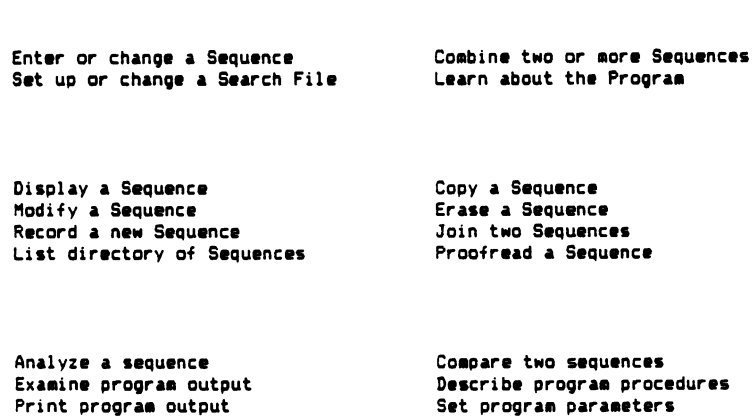


Figure 1. Menus used in the program. When appearing on the computer screen, the first letter of each choice is brightly lit. The user types one of these letters to choose a function, or types ? for information.
A. Primary menu for data entry section.
B. Secondary menu for sequence entry and change.
C. Primary menu for data analysis section.

listed briefly in the prompt, with the first letter of each choice made bright on the screen. The user types one of these letters to indicate his or her decision. The third kind of prompt requires the user to type one or more words, perhaps the name of a sequence. Having typed the answer, the user can alter it using simple word-processing facilities provided by the program, e.g. Delete and Insert keys. After the answer is in its final form, the Return key is pressed so that the computer will register it.

In response to each possible prompt, there are two other options that may be used. The user can type ? to receive information on the screen about the available choices, their effects, and their relations to other parts of the program. Some 150 of these help messages are incorporated in the program, totaling about 800 lines of text. Together they constitute complete instruction in the program's use. The messages are stored in memory rather than disk so that they can be displayed on the screen without delay. After displaying the message, the program then repeats the original prompt. The user can also simply press the Return key in reply to any prompt. For those prompts requesting non-critical information, the program then chooses the alternative most often used. (This alternative is noted in the help message.) But if the program cannot proceed without the information, it returns to the menu. Suitable use of the Return key can make movement

through the program very fast.

In designing the program, we have attempted to make it error-resistant. Incorrect input from the user will not cause the program to crash or become confused. For example, if the user answers a prompt by typing a letter that is not meaningful, the computer produces a short warning beep, displays on the screen the message, "This is not an allowable response; try again or type ?", and repeats the prompt. If the user asks to see a sequence that is not contained on the disk of recorded sequences, the computer beeps, displays the message, "There is no sequence by that name on the disk; try again", and asks for the name of another sequence. If the user provides a number that is not in an allowable range, or provides a letter when a number is needed, the computer will display a message and provide another opportunity to enter the number. Other errors are handled similarly.

The need to learn about a computer operating system has discouraged some biologists from using either mainframe or small computers. To avoid this problem, we have made our program useable without any knowledge of the IBM PC DOS operating system. Functions that would normally be carried out by an operating system, such as erasing or copying files containing sequences, have been incorporated in the program itself. The only operation that must still be performed by DOS is formatting disks, and this simple operation is explained in a program help message.

It is important that a large interactive computer program contain features that prevent the user from becoming "lost" or "stuck" in any part of the program. For this purpose, our program indicates in one of the corners of the screen, in highlighted display, the name of the function currently in use. After performing a function, the program generally will automatically return to a menu, from which other functions can be selected. When the program does not return to the menu directly (in order to give the user a chance to repeat the function), pressing the Return key in response to the next prompt will make it do so. In fact, the program is so designed that pressing Return in response to several consecutive prompts will always lead to a menu. Pressing Return when at a secondary menu leads to the higher level menu, and pressing Return at that menu exits the program.

Data entry features. Upon starting the data entry section of the program, a menu with four choices appears (Fig. 1A). If the user chooses the learn function by typing L, a brief tutorial on how to use the program is presented. As described above, more detailed information may be obtained throughout the program by typing ?'s.

If the user chooses the enter function by typing E, the secondary menu for sequence entry appears (Fig. 1B). The most fundamental function on this menu is the record function, which allows the user to enter and store on a disk nucleic acid and protein sequences. Sequences up to 60,000 nucleotides in length may be entered. After choosing this function, the user is asked to name the sequence, to classify it as DNA, RNA or protein, and to make an optional comment that will be stored with the sequence. For nucleic acid sequences, the program then asks whether the sequence will be entered from the keyboard or with a digitizing device. If the keyboard option is chosen, the program invokes a specialized screen editor designed to facilitate rapid sequence entry and correction (Fig. 2).

Specifically, the program divides the screen into a sequence area and a command area. Commands or sequence residues are entered at the position of a blinking cursor, which may be moved around by using four arrow keys. To enter the sequence, the user moves the cursor to the first empty position (#1 when beginning) and simply types the letters A,C,G,T and U representing nucleotides. The program automatically numbers the sequence, skips a space every 10 letters, shifts to a new line every 60 letters, and changes

```
PBR322                          Length   215        SEQUENCE: Record

       10         20         30         40         50         60
TTCTCATGTT TGACAGCTTA TCATCGATAA GCTTTAATGC GGTAGTTTAT CACAGTTAAA

       70         80         90        100        110        120
TTGCTAACGC AGTCAGGCAC CGTGTATGAA ATCTAACAAT GCGCTCATCG TCATCCTCGG

      130        140        150        160        170        180
CACCGTCACC CTGGATGCTG TAGGCATAGG CTTGGTTATG CCGGTACTGC CGGGCCTCTT

      190        200        210        220        230        240
GCGGGATATC GTCCATTCCG ACAGCATCGC CAGTC_

      250        260        270        280        290        300


      310        320        330        340        350        360


Locate Number:                          Delete From:
Locate String:                               To   :
Return (or ?):                          Insert At  :
```

Figure 2. Example of the program screen editor for sequence entry. The figure is an exact facsimile of the computer screen after 215 nucleotides of pBR322 have been entered, except that the topmost line is highlighted on the screen and the cursor at position 216 is blinking.

to a new page (with higher numbers) when needed. To change a letter, the user just types a new letter over it. To remove a letter, the user moves the cursor there and presses the Delete key. Letters can be inserted after the Insert key is pressed. The part of the sequence shown on the screen can be changed at any time by using the PageUp and PageDown keys.

The command area of the editor is used to locate features of a sequence while editing and to delete or insert long regions. For example, when a string of nucleotides is typed next to the Locate String command, the part of the sequence containing the first occurrence of the string is shown on the screen, with the string itself brightened. Various safety features are built into the editor. For instance, if the user types a letter other than one for the usual nucleotides, the computer produces a brief warning beep (although it accepts the letter). Upon completing the sequence, the user presses the Return key, the computer records the sequence, and the sequence entry menu reappears. Similar screen editors allow the rapid entry of protein sequences in either the 3-letter or 1-letter format.

A nucleic acid sequence may also be entered directly from an autoradiogram of a sequencing gel by the use of a sonic digitizing device, an idea due to R. Staden. This compact electronic device can precisely determine the position of a special pen, which emits an ultrasonic sound, by measuring the amount of time that the sound waves require to reach it. Upon choosing to enter a sequence this way, the program first asks the user to touch the corners of the gel lanes with the sonic pen. The user then touches successive bands on the autoradiogram with the pen, which can be done very quickly, and the program registers nucleotides according to which lanes contain the bands. To help avoid errors, the computer produces a tone as each band is touched, using a different pitch for each of the four possible nucleotides. The nucleotides entered this way appear on the screen as if typed from the keyboard, and the keyboard itself may be used to enter special letters for ambiguous nucleotides, or to make corrections as described above.

In addition to recording a new sequence, the menu for sequence entry contains seven other functions (Fig. 1B). The modify function allows one to change a sequence that has already been recorded, by means of the same screen editor used to enter sequences. The display function is used to examine a recorded sequence, using a limited screen editor that does not allow changes to be made. The display and modify functions also allow the user to respectively examine and change the optional comment

recorded with the sequence. The list directory function provides an alphabetical listing of all sequences recorded on the disk (Fig. 3). The directory gives the name of each sequence, its kind (DNA, RNA, or protein), length in residues, date of recording, and date of last modification. If the directory contains more sequences than will fit on the screen at once, the user can browse through it using the PageUp and PageDown keys. The directory is automatically compiled and maintained by the program.

Two other functions allow the user to copy and erase individual sequences. The program also provides a mechanism to erase or copy many sequences at once, and to view a directory containing only a specified group of sequences. In reply to a prompt asking for the name of a sequence, the user can give a name containing an * in it. Then the program will note internally all names that can be formed from the given name by replacing the * with any series of letters. The sequences with these names will all be erased, copied or listed in a directory, depending on the program function. For example, phage* includes all names that begin with phage; *promoter includes all names that end with promoter; and phage*promoter includes all names that begin with phage, end with promoter, and have any characters in the middle (e.g., phage lambda PL promoter). This mechanism, which is also used in the analysis section of the program, gives the user a way to treat groups of related sequences conveniently.

When entering sequences into the computer, mistakes in reading or typing are a major concern. It has been found that the fastest and most reliable method to ensure accuracy of the entered sequence is actually to enter the sequence twice, and then have the computer find any discrepancies (M. Dayhoff, personal communication). For this purpose we have included a proofreading function in the program, which uses a modification of a published homology algorithm (1) to compare two copies of a sequence and find any differences. A final function on the menu finds overlapping

NAME	KIND	LENGTH	DATE	UPDATE
Kappa gene gel 1	D	125	03-05-83	
Kappa gene gel 2	D	112	03-12-83	
Kappa gene gel 3	D	130	03-20-83	
Kappa gene project	D	345	03-12-83	03-21-83
Mouse kappa message	R	946	01-05-83	
Mouse kappa protein	P	217	01-05-83	
pBR322	D	4362	01-01-83	

Figure 3. Sample directory of sequences. The figure appears as on the computer screen (D = DNA, R = RNA, P = protein).

regions in two sequences and joins the sequences at the overlap. This function is useful when compiling a sequence from individual sequencing gels. We are also planning to incorporate into the program a procedure, similar to one described (35), that will automatically arrange and merge the many gel readings produced when sequencing by the "shotgun" method.

Returning now to the primary data entry menu (Fig. 1A), another function combines any number of different sequences into a longer sequence by simply linking them. Only specified parts of the sequences need be combined, and the orientation of any sequence may be reversed before combining it. Also, a combined nucleic acid sequence may be translated and stored as a protein. This procedure is useful in constructing a recombinant plasmid sequence from its constituent parts, or in joining the exons of a gene to represent the mRNA. As special cases, it includes the ability to store a specified part of a single sequence as a new sequence, to reverse the orientation of a sequence, or to translate and record a nucleic acid sequence as a new protein sequence.

Finally, there is a function to set up or change search files. A search file is a list of nucleotide or amino acid strings, such as restriction enzyme sites or sites of regulatory significance. As described below, the analysis section of the program will search a sequence for the strings in specified search files. The strings may contain special letters to represent alternative nucleotides, for example P for purine and Q for pyrimidine. Upon choosing this function, the user is presented with another menu, similar to the sequence entry menu, which lets him record, modify, display, copy, erase or list a directory of search files. Entering and modifying search files is made easier by another special purpose editor included in the program.

Analysis features. The program currently has 12 procedures for the analysis of individual sequences, as well as the ability to compare two sequences for homologous regions. Most procedures apply to both nucleic acid and protein sequences. The program has the capacity to analyze sequences up to 60,000 nucleotides in length, thus handling viral genomes such as lambda and adenovirus.

Upon starting the analysis section of the program, another menu appears (Fig. 1C). If the user chooses the function to describe program procedures, the 12 different procedures are listed on the screen (Table 1). The user is then asked to type the number of a procedure, and a detailed explanation of that procedure also appears on the screen. The explanation includes a

Table 1. Analysis procedures of the Program

1. List and number the sequence in various formats
2. Determine nucleotide or amino acid frequencies
3. Determine codon frequencies
4. Translate a nucleic acid in one frame or reverse translate a protein
5. Translate a nucleic acid in all reading frames
6. Locate sites in a sequence, find fragment lengths, and display in tables
7. Locate sites in a sequence and display graphically
8. Locate sites in a sequence and display along sequence
9. Find repeated regions in a sequence
10. Find inverted repeats in a nucleic acid sequence
11. Find sequence regions rich in specified residues
12. Determine protein secondary structure

description of what the procedure does, what its output format looks like, and what program parameters control it (see below). Descriptions of any number of procedures may be obtained before exiting this function.

The precise operation of the analysis procedures is regulated by 26 program parameters that may be assigned values by the user. For example, various parameters control the number of residues printed per line, the stringency of homology searches, and whether a 1-, 2-, or 3-letter amino acid code is used. To set these parameters, the user invokes the indicated menu function. A numbered list of all the parameters then appears on the screen, together with their current values and acceptable ranges. The program then prompts the user for the number of a parameter to change. After entering it, the user is prompted for the new value of that parameter, which is registered on the screen. The user can type ? instead of a new value; then a detailed explanation of the parameter, and a list of the procedures it controls, will appear. An attempt to enter a value for a parameter outside its allowed range will be rejected by the computer, followed by a new opportunity to enter its value. After as many values as desired have been entered, the whole set of parameters is stored on disk, from where it can be recalled when performing an analysis. Up to 10 distinct sets of values for the parameters can be stored and recalled, giving the user flexibility to perform different analyses under different conditions.

To actually analyze one or more sequences, the corresponding menu function is chosen. The program then prompts for the name of a sequence to

analyze. After verifying that the entered sequence is on the disk, the program asks the user to specify which set of stored parameter values should apply to the analysis of that sequence. By pressing the Return key in reply to this prompt, the user can tell the program to apply its built-in parameter values (those most generally useful) rather than a stored set. Next the program gives the user the option to restrict the analysis to part of the sequence, to reverse the orientation of a nucleic acid sequence before analysis, or to specify that a sequence is linear or circular. Then the program asks the user to indicate which search files should be used by the string search procedures. Pressing the Return key at that point specifies a particular search file that normally contains all restriction enzyme sites. Finally, the program asks for the numbers of the procedures to be applied to the sequence. Having obtained all the information needed to analyze one sequence, the program asks for the name of another sequence to analyze at the same time, and so on, until the user specifies that the analysis actually begin. The * notation described above can be used to direct that a group of related sequences be analyzed.

Turning now to the analysis procedures themselves (Table 1), procedure 1 lists and numbers a nucleic acid or protein sequence. Various parameters control the number of residues listed per line, the frequency of numbering, and whether a space is skipped after each 10 nucleotides and each amino acid. Other parameters can be used to list one or both strands of a DNA sequence, and to specify that a 1-,2- or 3-letter code be used for amino acids. The same format will then be used in the output of all other procedures that involve listing the sequence, such as translating it.

Procedure 2 determines the frequencies of the different nucleotides in a nucleic acid sequence and of the amino acids in a protein sequence. The frequencies are given both as an absolute count and as a percentage of the total number of residues. For nucleic acids, the A+T and C+G frequencies are also given, as well as the frequencies of the 16 possible dinucleotides. For proteins the numbers of acidic, basic, aromatic, and hydrophobic amino acids are given, as well as the calculated molecular weight.

Procedure 3 determines the codon usage frequencies in any one reading frame of a nucleic acid sequence. Like the other procedures, it may be restricted to a specified part of the sequence. Both absolute counts and percentages are given. The procedure also gives the frequencies of the amino acids represented by the codons and the molecular weight of the encoded protein.

Procedure 4 translates a nucleic acid in one reading frame or reverse translates a protein (Fig. 4A). Using parameters, the user can specify that certain amino acids in the translated sequence be replaced by * signs or + signs, thus highlighting those amino acids. Alternatively, the printing of all but specified amino acids can be suppressed. For example, one might have only the "amino acids" MET and END printed, thus making it easy to scan for open reading frames.

Procedure 5 translates a nucleic acid sequence in all three reading frames at once (Fig. 4B). The output from this procedure is easiest to read when a parameter is used to specify use of the 1-letter amino acid code or a special 2-letter code suggested to us, which consists primarily of the first 2 letters of the 3-letter code. The same methods of highlighting amino acids described above can be used here.

A

```
Gly His Val Ser Leu Met Ser Arg Gln Pro Gly Leu His Trp Ser Tyr End
GGN CAQ GUN UCN CUN AUG UCN CGN CAP CCN GGN CUN CAQ UGG UCN UAQ UAP
            AGQ UUP     AGQ AGP             UUP         AGQ     UGA
```

B

```
          .         .        30        .         .        60
TTCTCATGTTTGACAGCTTATCATCGATAAGCTTTAATGCGGTAGTTTATCACAGTTAAA
Ph Se Cy Le Th Al Ty Hi Ar ** Al Le ++ Ar ** Ph Il Th Va Ly
 Se Hi Va ** Gn Le Il Il Ap Ly Le ** Cy Gl Se Le Se Gn Le An
  Le ++ Ph Ap Se Le Se Se Il Se Ph An Al Va Va Ty Hi Se ** I
```

C

```
11/14   157   TATGCCGGTACTGC   170
0.79    148   ATACGG  ATGTCG   137
-14.0               **   *
```

Figure 4. Sample output from the program. The figure is taken directly from more extensive output, and appears as on the screen or after printing.

A. Output from procedure 4. A random sequence of 17 amino acids was reverse translated (P = purine, Q = pyrimidine, N = nucleotide)

B. Output from procedure 5. A part of pBR322 is shown translated in all reading frames. Various parameters specified the frequency of numbering, the use of a 2-letter amino acid code, and the translation of End codons into ** and ATG codons into ++ .

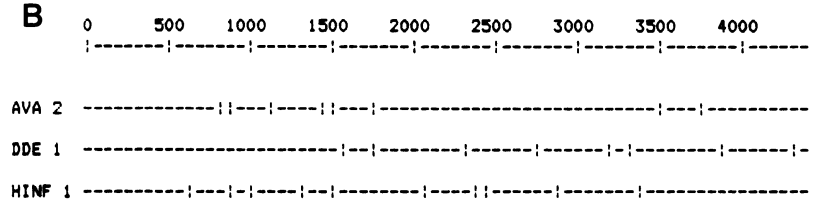
C. One inverted repeat found by procedure 10. The numbers around the inverted repeats are the nucleotide numbers where they begin and end. The ratio of correct matches to length is given as a fraction and decimal; the free energy of the stem-and-loop (38) is listed below.

Procedures 6-8 all locate strings in nucleic acid or protein sequences, but show the results in three different formats (Fig. 5). These procedures can locate nucleotide strings, such as restriction enzyme sites, in nucleic acid sequences and can find amino acid strings, such as modification sites, in protein sequences. Moreover, they can locate amino acid strings in a nucleic acid, i.e., find all sites in the nucleic acid that encode them. Finally, given a protein sequence they can find all places where the corresponding gene might contain specified nucleotide strings such as restriction sites. This last function may be useful in attempting to clone a gene after the encoded protein has been sequenced. By setting a parameter, the user can allow these procedures to accept a specified number of mismatches when searching for strings. This feature might be used in looking for

A

	#	SITES	FRAGMENTS		FRAGMENT ENDS	
TAQ 1 (TCGA)	7					
		24	1444	(33.1)	2574	4018
		339	1307	(30.0)	1267	2574
		651	475	(10.9)	651	1126
		1126	368	(8.4)	4018	24
		1267	315	(7.2)	24	339
		2574	312	(7.2)	339	651
		4018	141	(3.2)	1126	1267

B



C

```
      550        560        570        580        590        600
CTGTTGGGCG CCATCTCCTT GCATGCACCA TTCCTTGCGG CGGCGGTGCT CAACGGCCTC
     NH         S          F  F       H          H M
     AH         P          N  N       G          A N
     RA         H          U  U       I          E L
     11         1          1  1       1          3 1
```

Figure 5. Sample output from the string search procedures of the program. The figure is taken directly from more extensive output.

A. Output from procedure 6. Sites and fragments for Taq 1 in pBR322.

B. Output from procedure 7. Sites for 3 enzymes in pBR322 graphed.

C. Output from procedure 8. Sites for all enzymes between positions 540 and 600 of pBR322 (position 1 is at the unique Eco R1 site).

regulatory sites.

Procedure 6 lists the locations of the strings in tables (Fig. 5A), in the same format as our former program (2). For example, all the sites for each restriction enzyme can be tabulated. The tables also contain the lengths of the fragments produced by cutting at these sites, listed in descending order as on a gel. Next to each fragment, its percentage of sequence length and its end points are given. The program also provides a way to list the fragments produced by cutting with several enzymes at once, as in a multiple digest.

Procedure 7 shows the locations of each string graphically (Fig. 5B). The procedure first draws, subdivides, and numbers a line to represent the sequence, with a suitable size for the divisions chosen automatically. It then draws another line for each string and represents the occurrences of that string by vertical bars at the appropriate positions. For ease of reference, the numbered line is repeated after every six lines.

Procedure 8 shows the locations of all the strings in a search file simultaneously. It does so by displaying the sequence and listing the name of each string underneath the first residue of each site where it occurs (Fig. 5C). Actually, only an abbreviation for each name consisting of its first 3 and last 1 letters is used. If two restriction enzyme sites occur at the same place, the name of the more specific one is listed (e.g., GGATCC is more specific than GGPQCC, where P represents any purine and Q any pyrimidine). This procedure is useful in those genetic engineering applications which require some enzyme site in a given region of DNA.

Procedures 9 and 10 find homologies in a sequence (Fig. 4C): procedure 9 locates repeated regions while procedure 10 finds inverted repeats (stem-and-loop structures). These routines use the same algorithm that we described (1) and that others have adopted (36,37). Both mismatches and loopouts are allowed in the homologies. With each homology, the ratio of correct matches to total length is reported, and for stem-and-loops, the free energy is calculated according to the rules of Tinoco (38). Various parameters specify the extent of mismatching allowed, the maximum length of a loopout, and the minimum length required for an homology. Varying these parameters allows control over the quality and number of homologies found. For procedure 10, the minimum and maximum size of the hairpin loop can also be set. Direct repeats will be located in both nucleic acid and protein sequences; inverted repeats of course only in nucleic acids.

Procedure 11 finds regions of a sequence "rich" in particular nucleo-

tides or amino acids. More precisely, the program will put a row of * signs under every region of the sequence where at least M-out of N residues are in a specified set. The values of M and N can be established using parameters. This procedure could be used to locate regions of a nucleic acid rich in A's and T's, or regions of a protein rich in hydrophobic amino acids.

Procedure 12, currently the last procedure of the program, uses the rules of Chou and Fasman (39) to predict protein secondary structure. The protein is listed and an A is put under each amino acid expected to be part of an alpha helix, a B under each amino acid expected to be part of a beta sheet, and a T under each amino acid expected to be part of a turn.

Another menu function is used to compare two sequences for homologous regions. The homology algorithm is the same as the one used to find direct repeats in one sequence and allows a specified degree of mismatching. As usual, only parts of the sequences need be compared, and their orientations can be reversed. The * notation may be used to compare one sequence with a series of others. The program can compare two nucleic acid sequences or two protein sequences. Morever, it can compare a nucleic acid sequence directly with a protein sequence. That is, it can find all segments of the nucleic acid that encode any part of the protein, with a certain number of mistakes allowed. This ability is useful in aligning the sequence of a cDNA with the protein sequence it encodes.

The user can direct the output from an analysis or comparison to either a printer or a disk. If the output is stored on a disk, it can later be viewed on the screen using the examine function. This function divides the stored output into numbered, screen-size pages, and provides an index that lists which pages contain the various procedures and sequences. The user can give the numbers of the pages wanted to appear on the screen, or can scan through the output pages using the PageUp and PageDown keys. The output is stored in the form of a normal DOS textfile, so that it can also be examined or modified with word-processing programs, and can be used as the input to user-designed programs. Output stored on disk can also be printed, using the menu's print function. When controlling the computer printer, the program automatically adjusts the letter size to make the specified number of residues fit on a line. There is also an option to print with heavier letters for purposes of photographic reproduction.

A major concern in developing our program was that it run fast enough on a plasmid-sized sequence to avoid significant delay. In procedures 6-8, a straightforward search for the whole set of restriction enzyme sites on

the plasmid pBR322 (4362 base pairs) turned out to require about 12 minutes. To reduce this time to an acceptable value, it was necessary to devise a new algorithm that would look for all enzyme sites essentially simultaneously, as described in Methods. The time now required for typical procedures operating on the test plasmid pBR322 is shown in Table 2; a longer sequence will require proportionally more time. Only operations to find all direct repeats at any relative positions, or all inverted repeats with any loop size, can require a lot of time, as is also true on mainframe computers.

National Nucleic Acid Sequence Data Bank. The National Institutes of Health and other government agencies have funded a project to collect all published nucleic acid sequences into a National Data Bank. Although the data bank is available online through Bolt Beranek and Newman Inc., no restrictions have been placed on its redistribution. We have therefore developed a program (unpublished data) that arranges the data bank into a form compatible with the program described here. Because only 2 bits of data are required to define the 4 nucleotide possibilities, we can pack 3 nucleotides into an 8-bit byte and still leave room for special characters. It is therefore possible to store about a million nucleotides from the Data Bank (or from user sequences) on one 360 kilobyte disk.

We have placed the sequences in the current Data Bank on four disks, with one disk each for eucaryotes, viruses of eucaryotes, procaryotes, and viruses of procaryotes. The program handles the Data Bank sequences in just the same manner as sequences entered by the user: it can display, analyze and compare them. The * notation described above is especially useful in searching the Data Bank for sequences from one organism or of one type, by listing an appropriate directory of sequences. The * mechanism is also useful in analyzing a group of related sequences from the Data Bank. We have included

Table 2. Sample performance of the program on pBR322

Procedure	Time (in sec.)
2. Count nucleotides and dinucleotides	6
5. Translate in all reading frames	73
7. Locate all restriction sites and display graphically	85
10. Find all inverted repeats with at least 6 matches and a hairpin loop less than 20 nucleotides in length	137

in the program a new and fast algorithm, to be described elsewhere, for comparing a user sequence with some or all of the Data Bank sequences in a feasible amount of time.

Although the Data Bank is still small enough to fit on four floppy disks, this arrangement is not ideally convenient because the user must sometimes change disks. We have therefore written a version of the program that utilizes a Winchester hard disk with a capacity of 10 megabytes, enough capacity for the whole Data Bank even after substantial growth. This version of the program also allows users to store their own sequences on the hard disk, while preventing different users from damaging each other's data. Finally, both sections of the program itself can be stored on the hard disk, making the whole program optimally convenient to use.

ACKNOWLEDGEMENTS

We would like to thank M. Wegman and B. Paterson for useful conversations.

*Footnote

Correspondence and requests for reprints should be addressed to Dr. Laurence Jay Korn. The program described here and compatible copies of the National Sequence Data Bank are available from SciSoft Inc., No. 189, 1259 El Camino Real, Menlo Park, California 94025.

REFERENCES

1. Korn, L.J., Queen, C. and Wegman, M.N. (1977). Proc. Natl. Acad. Sci. USA 74, 4401-4405.
2. Queen, C. and Korn, L.J. (1980). Methods Enzymol. 65, 595-609.
3. Queen, C., Wegman, M.N. and Korn, L.J. (1982). Nucleic Acids Res. 10, 449-456.
4. Selzer, G., Som, T., Itoh, T. and Tomizawa, J. (1983). Cell 32, 119-129.
5. Hsich, T. and Brutlag, D. (1979). J. Mol. Biol. 135, 465-481.
6. Danner, D.B., Deich, R.A., Sisco, K. and Smith, H.O. (1980). Gene 11, 311-317.
7. Epstein, P., Reddy, R. and Busch, H. (1981). Proc. Natl. Acad. Sci. USA 78, 1562-1566.
8. Mount, S.M. and Steitz, J.A. (1981). Nucleic Acids Res. 9, 6351-6368.
9. Halling, S.M. and Kleckner, N. (1982). Cell 28, 155-163.
10. McCutchan, T., Singer, M. and Rosenberg, M. (1979). J. Biol. Chem. 254, 3592-3597.
11. Baroudy, B.M., Venkates, S. and Moss, B. (1982). Cell 28, 315-324.
12. Cheung, A. and Kieff, E. (1982). J. Virology 44, 286-294.
13. Shimotohno, K., Muzutani, S. and Temin, H. (1980). Nature 285, 550-554.
14. Shoemaker, C., Hoffman, J., Goff, S.P. and Baltimore, D. (1981). J. Virology 40, 164-172.
15. Rushlozo, K.E., Lautenberger, J.A., Papas, T.S., Baluda, M.A., Perbal, B., Chirikjian, J.G. and Reddy, E.P. (1982). Science 216, 1421-1423.

16. Kupper, H., Sekiya, T., Rosenberg M., Egan, J. and Landy, A. (1978). Nature 272, 423-428.
17. Hovemann, B., Sharp, S., Yamada, H. and Soll, D. (1980). Cell 19, 889-895.
18. Olins, P.O. and Nomura, M. (1981). Cell 26, 205-211.
19. Korn, L.J. and Brown, D.D. (1978). Cell 15, 1145-1156.
20. Young, R.A., Macklis, R. and Steitz, J.A. (1979). J. Biol. Chem. 254, 3264-3271.
21. Long, E.O., Rebbert, M.L. and Dawid, I.B. (1981). Proc. Natl. Acad. Sci. USA 78, 1513-1517.
22. Konkel, D.A., Tilghman, S.M. and Leder, P. (1978). Cell 15, 1125-1132.
23. Schon, E.A., Cleary, M.L., Haynes, J.R. and Lingrel, J.B. (1981). Cell 27, 359-369.
24. Hieter, P.A., Max, E., Seidman, J.G., Maizel, J.V. and Leder, P. (1980). Cell 22, 197-207.
25. Hieter, P.A., Maizel, J.V. and Leder, P. (1982). J. Biol. Chem. 257, 1516-1522.
26. Nakanishi, S., Inoue, A., Kita, T., Nakamura, M., Chang, A.C.Y., Cohen, S.N. and Numa, S. (1979). Nature 278, 423-427.
27. Nunberg, J.H., Kaufman, R.S., Chang, A.C.Y., Cohen, S.N. and Schimke, R.T. (1980). Cell 19, 355-364.
28. Jones, C.W. and Kafatos, F.C. (1980). Cell 22, 855-867.
29. Nichols, B.P., Van Cleemput, M. and Yanofsky, C. (1981). J. Mol. Biol. 146, 45-54.
30. Wu, A.M., Chapman, A.B., Platt, T., Guarente, L.P. and Beckwith, J. (1980). Cell 19, 829-836.
31. Conrad, B. and Mount, D.W. (1982). Nucleic Acids Res. 10, 31-38.
32. Larson, R. and Messing, J. (1982). Nucleic Acids Res. 10, 39-49.
33. Pustell, J. and Kafatos, F.C. (1982). Nucleic Acids Res. 10, 51-59.
34. Fristensky, B., Lis, J. and Wu, R. (1982). Nucleic Acids Res. 10, 6451-6463.
35. Staden, R. (1982). Nucleic Acids Res. 10, 2951-2961.
36. Brutlag, D.L., Clayton, J., Friedland, P. and Kedes, L.H. (1982). Nucleic Acids Res. 10, 279-294.
37. Harr, R., Hagblom, P. and Gustafsson, P. (1982). Nucleic Acids Res. 10, 365-374.
38. Tinoco, I., Borer, P.N., Dengler, B., Levine, M.D., Uhlenbeck, O.C., Crothers, D.M. and Gralla, J. (1973). Nature New Biology 246, 40-41.
39. Chou, P.Y. and Fasman, G.D. (1974). Biochem. 13, 222-245.