Volume 12 Number 1 1984 Nucleic Acids Research

Rapid searches for complex patterns in biological molecules

R.M.Abarbanel, P.R.Wieneke, E.Mansfield, D.A.Jaffe and D.L.Brutlag

IntelliGenetics, Inc., 124 University Avenue, Suite 300, Palo Alto, CA 94301, USA

Received 17 August 1983

ABSTRACT

The intrinsic redundancy of genetic information makes searching for patterns in biological sequences a difficult task. We have designed an interactive self-documenting computer program called *QUEST* that allows rapid searching of large DNA and protein data banks for highly redundant consensus sequences or character patterns. *QUEST* uses a concise language for specifying character patterns containing several levels of ambiguity and pattern arrangement. Examples of the use of this program for sequence data are given. Details of the algorithm and pattern optimization are explained.

INTRODUCTION

Biological Sequence Redundancy

Pattern matching in biological sequence information is often complicated by functional redundancy: many different sequences can perform the same biological role. The redundancy of protein sequence became clear when proteins of identical function were sequenced from a variety of organisms. The elucidation of the genetic code demonstrated redundancy at the level of nucleotide coding sequences. As many as six different triplets can encode the same amino acid. A third level of biological sequence redundancy occurs at the level of protein-DNA interaction. Operator sequences for repressor binding and promoter sequences for the initiation of transcription show highly ambiguous consensus sequences. Even in the simplest case, the recognition of short discrete base sequences by a restriction endonuclease, sequences are often not specified exactly. Finally, the folding of RNAs shows considerable functional redundancy. Very often the mere presence of a base pair is essential for determining RNA secondary structure and function, while the exact sequence is irrelevant.

Each of these levels of functional redundancy (protein-DNA interaction, genetic code, RNA structure and protein structure) have pronounced effects on the flow of genetic information from DNA to phenotype. Nature has often made use of this flexibility of expression to impose control at each of the critical steps in this flow of genetic information. Unfortunately this ambiguity causes great difficulty for molecular biologists wishing to extract the maximum amount of biological information from sequences. The comparison of numerous sequences filling similar biological roles often lead to

highly ambiguous consensus sequences for these functional regions. What is needed is a method of finding such patterns with amounts of redundancy varying from the well defined genetic code to the least well defined aspects of RNA and protein folding.

Pattern Matching Solutions

A large number of methods for both exact and inexact pattern matching in large databases have been developed in the field of computer and information science, however, none of these methods is adequate when faced with the levels of redundancy described above. The algorithm of Knuth, Morris, and Pratt [1], for instance, allows exact pattern matching of a string of characters with a large database. A more flexible pattern matching algorithm is based on Knuth's "trie" search algorithm [2, 3]. As implemented in the XSEARCH program by Kanerva and Daniels this method generates a finite state machine (FSM) that is used to scan a very large database for either a single pattern or any boolean relationship between several patterns. For example, to represent the nucleic acid triplet "ANG", XSEARCH uses four patterns: "AAG", "ATG", "AGG", and "ACG". XSEARCH is also conveniently designed to scan a very large number of files sequentially and to report any target found in any of them. The FSM makes for an extremely fast matching of exact patterns that are simultaneously sought in the input stream to the program.

One of the most flexible representations of ambiguous patterns is provided within UNIX™ [4] utility programs for matching of regular expressions against lines in files. The UNIX concept of regular expressions is a set of symbols used to represent allowed character variations in patterns. (An adaptation of these symbols is provided in Table 1.) These UNIX utilities allow potentially very complex patterns to be defined with regular expressions. The application of this UNIX pattern matching language to a biological problem was first reported by Abarbanel and co-workers [5]. They were making secondary structure assignments in α/β proteins based on rules whose premises were the matches of sequences with complex patterns.

We have developed a tool called ***QUEST*** to allow the flexible exploration of sequences in a data bank. ***QUEST*** combines the flexibility of the UNIX pattern matching utilities and the speed of a finite state machine. In addition, ***QUEST*** allows patterns to be defined in terms of other patterns. Using these patterns and patterns of patterns one can express most of the naturally occurring redundancy in biological information. ***QUEST*** is also a powerful general database searching tool that can be used for finding references from bibliographic files, strains of bacteria or plasmids with particular properties from a strain collection, names and addresses from telephone lists, etc.

METHODS

The search algorithms used in ***QUEST*** were chosen to optimize two opposing factors: first, to meet diverse requirements imposed by the variety of different character patterns to which it would be applied, and, second, the knowledge that it would need to be fast enough to be useful for searching large data bases. ***QUEST*** needs to handle regular text by line, page or file, and sequences of amino acids and nucleic acids, or their associated comments

Table 1.

Patterns are defined using regular alphabetic and numeric characters and special symbols. The simplest pattern is a string of characters that matches only itself. Special characters are used to indicate beginnings and ends of lines, regions of variability, and repeated regions. Some words which act as conjunctions between symbols or expressions of symbols are reserved for this role. These are detailed in Table 2.

SYMBOLS USED IN PATTERN DEFINITIONS

Symbol	Meaning
.	Any character.
↑	Beginning of line, sequence, paragraph, page or file.
$	End of line, sequence, paragraph, page or file.
*	ZERO or more repeats of the proceeding symbol or expression.
+	A symbol followed by + means ONE or more repeats of that symbol or expression.
?	A symbol followed by ? means ZERO or ONE of that symbol or expression.
-	"Through", used in [...] to indicate a range of values, e.g. [A-CG-K] means [ABCGHIJK].
[ ]	Logical inclusive OR of symbols in brackets, as in [ABC].
[~]	Negation of logical inclusive OR of symbols in brackets, e.g. [~A-C] means any character other than A, B, and C.
{m} {m,} {m,u}	Integer values in {} indicate the number of times that a preceding delimited expression is to be applied; m is the minimum number and u is the maximum. A missing u means infinity. If only m is present, then the pattern is applied exactly m times. The symbol '+' is equivalent to {1,} and '*' is equivalent to {0,}, and '?' is equivalent to {0,1}.
( )	Used for grouping. The '*' and '+' operators may apply to single characters, or to groups of characters enclosed in brackets '[]', or to expressions enclosed in parentheses.

The basic search engine is a finite state machine which is compiled within *QUEST* from patterns represented by strings of character symbols. These symbols are denoted as regular expressions with syntax adapted from the string patterns available in the UNIX operating system and searching utilities (See Table 1.).

The finite state machine implemented in *QUEST* is a dynamically allocated two-dimensional array within which each location contains the number of a state within the current key compilation. The first dimension is addressed by the integer value of the input characters, the ASCII code on most systems, and the second dimension by the current state. The result of each table look-up is the next state. There is also an array, parallel to the STATE dimension of the search matrix, which indicates whether or not an accepting state (complete key) has been reached. Thus the search is table driven

Table 2.

The conjunctions AND, OR, THEN, & (for concatenation), and NOT can be used to combine patterns of simpler patterns. The conjunction names are reserved for these meaning. Consequently to match the names of the conjunctions themselves, they must be quoted. For example, to match the word THEN in the text of this article, the pattern would have to include the quotes: "THEN". Several example patterns that use the special symbols and conjunctions are found in Table 3.

CONJUNCTIONS	
THEN	Is used between patterns to make a match only when the second pattern follows the first.
&	Acts similarly, but requires that the second pattern directly follow the first pattern. '&' may be used to split very long patterns. It is a concatenation operator.
OR	Connected patterns will match when either the first or second pattern matches.
AND	Connected patterns will match when both patterns match.

by the values in the array and the incoming data.

As a general search technique, finite state machines are very fast, although they require a great deal of memory to represent more complicated keys. As one might expect using a two-dimensional array to represent the search key, the FSM grows geometrically, or possibly exponentially in certain worst cases, with the length of the key patterns specified by the user. Unfortunately, these "worst" cases are often generated by the highly ambiguous keys which occur when searching biological data.

An added level of control is provided that can be thought of as a parse tree in which the leaves are each a compiled FSM. Operators at this level are termed "conjunctions" and are detailed in Table 2. Traversing nodes marked by conjunctions translates to making decisions about the state of the search in terms of where it should continue within the tree. In the case of the "&" (concatenation) and "AND", the programs pointer into the text being searched may also need to be reset.

Splitting up keys can help conserve memory in two ways. First, most array implementations have an easier time allocating several small arrays than one large array, even though the total word size is the same. Secondly, keys which are ambiguous, particularly those which can vary in length, may grow very rapidly with the number of possibilities following each point of ambiguity. This is particularly a problem with keys which use many recurring characters, as with searches in DNA sequences. If one can break up such a key, the number of consequent possibilities is diminished considerably. By way of example, the key "aa..aaa..a" requires one array dimensioned 41 by 128 whereas the key "aa..a & aa..a" would use two 12 by 128 arrays.

In general, the less a FSM is broken up, the faster it executes. This is particularly true if the search requires more than one pass over the data. This memory-size verses execution-time trade off is the basis for most of QUEST's optimization attempts, something the sophisticated user should keep

Table 3.

Examples of patterns or "keys". Simple exact matching patterns and several example of more complex patterns are shown. Also illustrated are patterns defined in terms of other patterns. The key PROMOTER matches 28 times in the NIH GenBank ™ (May 1983) sequences.

EXAMPLES OF QUEST PATTERNS

KEY NAME	PATTERN	What this pattern matches:
SIMPLE	GAGA	Only GAGA.
SIMPLE-VAR	GA.A	GA followed by any character followed by A. For example, GAYA, GATA, GAAA, or GAGA.
GAS	GA*S	G followed by ZERO or more A's followed by one S. For example, GS, GAS, GAAAAAAAS.
G-A-T	GA?T	G followed by ZERO or ONE A followed by T, that is, GT or GAT.
GA-X-A	GA.*A	GA followed by ZERO or MORE characters of any type, followed by ONE A. For example, GAA, GATTTA, or GAEKQRTA.
NUC2	[AT][GC]	A or T followed by G or C, that is, AG, AC, TG or TC.
NOT-EARLY-ALPHA	[~A-M]	Any character not between A and M inclusive.
Mult-TA	T+A+	ONE or MORE T's followed by ONE or MORE A's, for example, TA, TTTA, TAAA, or TTAAAA.
CHGPAIR1	K...D	K separated by 3 arbitrary characters from a D. K = Lysine, D = Aspartic acid, a charged pair.
CHGPAIR-A	[KR]...[DE]	K or R separated by 3 arbitrary characters from a D or E. E = Glutamic acid.
MULT-C-PAIRS	(K...D){3,7}	3 to 7 (K...D) type patterns in a row. This might also have been written using the CHGPAIR1 definition as CHGPAIR1{3,7}.
AUTH-LINE	↑AUTHOR	A line beginning with the word AUTHOR. This may be of use in comments in sequence files.
END-METHOD	METHOD$	A line ending with METHOD.
AccI	GTJKAC	A key to be interpreted as Stanford DNA Code for the AccI restriction enzyme -- when expanded, this key will be G[TU][CAJ][TUGK]AC
CUT-BOTH	AccI AND HindII	Both AccI and HindII sites are present.
CUT-ORDER	AccI THEN HindII	AccI site followed by anything followed by HindII site.

in mind for extensive searches.

Certain operators such as "NOT" are not explicitly provided in the regular expression syntax; these are however, provided as part of the "additional functionality" mentioned above. Also included is the ability to specify arbitrary repeats of patterns or expressions of patterns. These repeat symbols

provide a shorthand for what would require much longer patterns using the simple regular expression syntax. In addition, the conjunctions are more mnemonic and easier to learn than their equivalent notation without the conjunctions.

Once a key has been specified, a series of source level transformations are done. The primary task of the optimizer is to make sure that keys made up of regular expressions linked by conjunctions are translated into the proper FSM's. For example, the key "abc & (de OR fg)" should become "abcde OR abcfg". Another function is to break up long regular expressions, especially those which are "punctuated" by a groups of "." characters. Finally, ***QUEST*** tries to simplify patterns linked by "&", "OR" or "AND" into single patterns (that is, if the resultant patterns won't take up too much memory). This is to gain a faster execution time. Thus, the compound key, "TGA OR ATG" becomes the single regular expression, "(TGA)|(ATG)" and is compiled into a FSM as such.

When keys are defined in terms of other keys, this sort of optimization allows the final fully defined key to be treated as if the entire key were entered at once. Several examples of such hierarchical defined keys are provided in the Discussion and Tables 3 and 5.

Much of ***QUEST***'s power comes from knowing something about the data, and typical biological uses of the program. When searching a sequence database ***QUEST*** can take advantage of the known format to skip over inappropriate text. For example, if one is searching the comments at the head of each sequence, the program simply skips over the sequence itself. Likewise, the comments are skipped while ***QUEST*** is looking at sequences.

A related feature is ***QUEST***'s ability to search circular sequences properly. Because sequences are stored in a linear representation, most other search facilities have no way of dealing with key occurrences across the "ends" of circular sequences.

Realistically, biologists are as likely to want to search for "ATG .{35,250} TAA", or even "ATG (...){35,250} (TAG OR TAA OR TGA)". Unfortunately, the "." symbol is implemented rather inefficiently within the FSM. Long strings of dots or the ".?" combination must be trapped by ***QUEST*** and handled by special code during the running of a search rather than at the time the FSM's and parse tree are constructed.

DISCUSSION and RESULTS

Pattern Language

QUEST is both a pattern matching program and a database access tool that can search large databases for character patterns which are highly ambiguous. Complex consensus sequences such as those described for promoter-operator interaction can be quickly located even in very large databases. As described in **METHODS**, we have used a simple language similar to that of the UNIX pattern matching facility for expressing character patterns in a concise form. Table 1 shows the flexibility afforded in the expression of character strings. This language allows one to describe all of the known restriction enzyme recognition sequences and consensus sequences for TATA-boxes etc.

Table 4.

Patterns for the Genetic Code. The wobble of the third base is expressed as the symbol "." which will match any single character; or as sets, for example in the pattern for Serine where the third base may be either T or C. The letters Q, Y, P and R are also included in case the sequence database contains those letters to represent pyrimidine and purine ambiguities. One could also replace T in the above keys with [TU] to allow the keys to search RNA sequences.

AMINO ACID CODON KEYS

1. Ala - GC.	13. Met - ATG
2. Arg - (CG. OR AG[AG])	14. Phe - TT[CTQY]
3. Asn - AA[CTQY]	15. Pro - CC.
4. Asp - GA[CTQY]	16. Ser - (TC. OR AG[CTQY])
5. Cys - TG[CTQY]	17. Thr - AC.
6. Glu - GA[AGPR]	18. Trp - TGG
7. Gln - CA[AGPR]	19. Tyr - TA[CTQY]
8. Gly - GG.	20. Val - GT.
9. His - CA[CTQY]	21. Stop - (TA[GARP] OR TGA)
10. Ile - AT[TCA]	23. Ochre - TAA
11. Leu - (CT. OR TT[AGPR])	23. Amber - TAG
12. Lys - AA[AGPR]	24. Umber - TGA

On top of the character level representation, we have added a higher level syntax for combining several individual patterns together to allow the expression of more complex relationships between character expressions. Conjunctions such as PATTERN1 **and** PATTERN2, or PATTERN1 **then** PATTERN2, or PATTERN1 **and not** PATTERN2, etc. provide the flexibility needed for the expression of the ambiguity found in the genetic code and in promoter sequences. Table 4 shows some examples of using the **or** conjunction for expressing the codons for all of the amino acids.

A major advantage of having a second level of syntax is that it allows one to build up patterns (or search KEYs as they are called) based on other previously defined keys. This allows one to build up libraries of sequence patterns (such as amino acid codons or restriction sites) and then to define much more complex relationships using these keys as starting elements. Banks of search KEYs can be saved for future use and libraries of them can be assembled.

In order to provide the power needed to search large databases such as the European Molecular Biology Laboratory [6] or the NIH GenBank [7] DNA databanks we have used this pattern matching language to generate a finite state machine which can search large databases in a single pass without doing backtracking.

Sequence Database Files and File Hierarchy

In response to the *QUEST* prompt for the names of the files to search one may respond with the either the name of a single file, a specification that can stand for a group of files (i.e. *.NIH would stand for all files with the extension NIH), or one can indirectly specify the files by giving the name of a file which contains the names of files to search. These indirect file specifications can contain further indirect file specifications allowing a complete hierarchical description of files. The ability to have

indirect files containing the names of files for *QUEST* to search provides the user a way of organizing a large hierarchical database such as found in taxonomic classifications. The entire nucleic acid database may be organized in lists of files by genus, phylum, etc.

The SCOPE of a Search

To facilitate the searching of biological databases and large text files, *QUEST* allows control over the scope of the search in the data files. For instance, one may limit the search to only the sequences (SEQUENCE scope) or only the annotations (COMMENTS scope) in protein and DNA sequence databases. On the other hand, one can limit the search to individual LINEs, PAGEs or entire FILEs for general text files. The scope of the search influences the interpretation of boolean relations in the KEYs. For instance if one were searching for "SV40 AND PROMOTER" in LINE scope, one would be asking for a single line containing both words "SV40" and "PROMOTER" in any order. However, in PAGE scope, these two words need only occur on the same page. In COMMENTS scope *QUEST* will find any DNA sequence which contains those two words anywhere in the annotations to a single sequence.

The LINE, PAGE, and FILE scopes can be used on any type of text file. For example if one had a strain collection with many properties of each strain listed one strain to a page, then searching in page scope would allow one to find a strain given a boolean relation between any set of descriptive words included among the strain properties. Similarly, bibliographic files with literature references could be searched for any combination of properties simultaneously depending only on the information stored in the file.

The SEQUENCE and COMMENTS scopes require a more specific format for the data files, namely the format of sequence files used by the IntelliGenetics' *SEQ* program [8]. This format contains numerous sequences per file, each preceded by comments or annotations. The lines containing the comments begin with a semicolon and comments are followed by a line containing the sequence name (an arbitrary character string that uniquely labels the sequence in the database). The format of the sequence itself is simpler in that *QUEST* ignores all blanks and carriage returns so the lines can be of any length. This format is the same as that used by *QUEST* for files of KEYs. Keys may have COMMENTS preceding the NAME. The key definition then follows until the end of page or file.

Output Control

QUEST allows a great degree of control over the output of the program. One can search for a pattern of nucleotides and ask *QUEST* to report only the names of the FILES that contain the search KEY, or to report both the file names and the sequence names. Further, one can request that *QUEST* print the actual pattern of bases found in the database together with the surrounding sequence. *QUEST* will also print out the complete annotation for each sequence containing the pattern.

From a database management point of view, the most important output capability of *QUEST* is to COLLECT all of the sequences containing the search KEYs from a large database into another, probably smaller file for later analyses by other programs as well as for searching for yet other keys by

QUEST. *QUEST* also remembers the names of the files that contained the pattern during each previous search, allowing one to perform additional and/or more stringent searches on this subset of files.

Examples

The flexibility of *QUEST* for finding search KEYs in large databases and the control of the output is best shown by a few examples of the actual interaction. All of the examples except the first have been edited to show only the most relevant points.

Ambiguous patterns for amino acid codons. The first example shows how one can represent amino acid codons in order to form ambiguous nucleotide patterns that represent a given amino acid sequence. These patterns are then used to search a nucleotide sequence database for any occurrence of a region that could code for the given amino acid sequence without having to translate all of the coding regions.

```
                    > O <
                    O| |O  IntelliGenetics
                    > O <

          QUEST - Biological Database Management System

                TOPS20 Version 3.0 - July 1983

              (C) 1983 by IntelliGenetics, Inc.

           Enter the options you would like, or type

               "Q" to quit,
               "HELP" for the help menu, or
               "?" for the command list.

QUEST [Sequence]: load opioid.key      THE FIRST COMMAND IS TO LOAD SEARCH
                                       KEYS FROM A PREVIOUS QUEST RUN
 Loading from opioid.key

  1.  Arg -  (CG.  OR  AG[AG])            AMINO ACID CODONS FROM TABLE 4
  2.  Gly -  GG.
  3.  Leu -  (CT.  OR  TT[AG])
  4.  Lys -  AA[AG]
  5.  Met -  ATG
  6.  Phe -  TT[CT]
  7.  Tyr -  TA[CT]
  8.  MINIMAL -  Tyr Gly Gly Phe        MINIMAL PEPTIDE WITH OPIOID ACTIVITY
  9.  DELTA -  (MINIMAL Leu) OR (MINIMAL Met)
  10. DYNORPHIN -  (DELTA Arg Arg) OR (DELTA Arg Lys)
```

These keys show the ability to record patterns used in prevous searches and also the ability to define more complex keys in terms of patterms previously defined. For example the **MINIMAL** key, "Tyr Gly Gly Phe", is interpreted by *QUEST* to be TA[CT]GG.GG.TT[CT] where the square brackets mean one and only one of the enclosed letters is allowed at that position; the "." stands for any character.

```
QUEST [Sequence]: setting                    THESE TWO COMMANDS TELL
SETTING [Sequence]: output hits              QUEST TO PRINT ONLY
                                             TARGETS FOUND.

QUEST [Sequence]: run minimal
Key = TA[CT] GG. GG. TT[CT]                  QUEST PARSES MINIMAL KEY

File(s) to Search (<CR> when done): HUMAN.NIH
File(s) to Search (<CR> when done):     WE SEARCH ONLY FOR HUMAN OPIOIDS.
        PROMPT = YES                    QUEST DESCRIBES THE PARAMETER
        OUTPUT = HITS-ONLY              SETTINGS BEFORE BEGINNING SEARCH.
        SCOPE = SEQUENCE

searching <SEQUENCES>HUMAN.NIH.8303
    HUM58SRNA
    HUM5SRRNA  ...
    HUMCMOS
    HUMENKEPH

HUMENKEPH
| Hit starts at position 523 of HUMENKEPH and ends at 534
V
      532 534
TATGGGGGCT TC

(Q)uit, (N)ext sequence, or (C)ontinue? (<CR> = C)

HUMENKEPH
| Hit starts at position 544 of HUMENKEPH and ends at 555
V
      553 555
TATGGAGGCT TC

(Q)uit, (N)ext sequence, or (C)ontinue? (<CR> = C) Q
QUEST [Sequence]:  q
```

We terminated the above search after finding only the first two occurences of opioid peptides, however a complete search of the human sequences shows that only known opioid proteins were found.

Gathering sequences having "HORMONE" in comments. In the next two examples we first find all of the FILES in the IntelliGenetics version of the NIH GenBank which contain the word HORMONE. After finding the relevant files, we then ask *QUEST* to pick out every sequence containing HORMONE in its comments and to place them into a new file called HORMONE.SEQ:

```
QUEST [Sequence]: setting
SETTING [Sequence]: scope comments          THIS COMMAND TELLS QUEST
                                            TO SEARCH ONLY THE ANNOTATIONS
                                            AND NOT THE SEQUENCES.

SETTING [Comments]: run
Specify key (type "^" to abort):
Key:  hormone                    THE SEARCH KEY WE WANT IS THE WORD "HORMONE".

File(s) to Search (<CR> when done): *.NIH
File(s) to Search (<CR> when done):  WE WILL SEARCH THE ENTIRE GENBANK DATABASE.
     OUTPUT = FILES-ONLY     SCOPE = COMMENTS
```

```
searching <SEQUENCES>ADENOVIRUS.NIH.8303
searching <SEQUENCES>AKV-VIRUS.NIH.8303
      .
searching <SEQUENCES>BOVINE.NIH.8303     *Found*
      .
      .
searching <SEQUENCES>HUMAN.NIH.8303      *Found*
      .
searching <SEQUENCES>MOUSE.NIH.8303      *Found*
      .
searching <SEQUENCES>YEAST.NIH.8303
```

At this point *QUEST* has found all files that contain the word hormone. During this search we could have asked *QUEST* to pull out the relevant sequences from the database and store them in a new file, however, for frequent keywords it is often more efficient to merely find the names of the files containing the keyword first. This also allows one to fine tune the search by adding additional search criteria if the first search proves to be too broad.

```
QUEST [Comments]: setting (confirm with carriage return)
SETTING [Comments]: collect (into optional file name) HORMONE.SEQ
    Opening HORMONE.SEQ.
SETTING [Comments]: run

Specify key (type "↑" to abort):
<CR> = "HORMONE"
Key:                                    QUEST REMEMBERS LAST SEARCH KEY.
Key = HORMONE

Type one of the following:
    <CR>          = Current File list
    !             = List of previous hits
    ↑             = Abort
    ?             = Help on file specification
    <file spec.>  = Start new file list

File(s) to search: !                    SEARCHES FILES PREVIOUSLY FOUND TO
                                        CONTAIN "HORMONE".

searching <SEQUENCES>BOVINE.NIH.8303
    BOVAVPNPII                  *Found*
    BOVGH                       *Found*
    BOVPTH                      *Found*
    BOVPTH2                     *Found*
    BOVTHYMOA1                  *Found*
  5 hits found in <SEQUENCES>BOVINE.NIH.8303

searching <SEQUENCES>FISH.NIH.8303
    FSHSOMAT22                  *Found*
  1 hit found in <SEQUENCES>FISH.NIH.8303

searching <SEQUENCES>HUMAN.NIH.8303
    HUMSOMAT                    *Found*
  1 hit found in <SEQUENCES>HUMAN.NIH.8303

searching <SEQUENCES>MOUSE.NIH.8303
    MUSBMSHEND                  *Found*
  1 hit found in <SEQUENCES>MOUSE.NIH.8303
```

```
searching <SEQUENCES>RAT.NIH.8303
    RATCALCITN                  *Found*
    RATGH1                      *Found*
     .
     .
    RATSOMAT                    *Found*
  14 hits found in <SEQUENCES>RAT.NIH.8303

QUEST [Comments]: quit
    Closing HORMONE.SEQ              QUEST SAVES ALL SEQUENCES IN NEW FILE.
```

During this second search, the annotations of each sequence have been examined for the key HORMONE and when found the name of the sequences are printed. Each of these sequences have also been collected into the file HORMONE.SEQ for either additional searches or other analyses.

Long open coding region. The flexibility of the *QUEST* pattern language is shown in the next example in which the sequences in the HORMONE.SEQ file can be searched for long open coding region:

```
QUEST [Sequence]: load (optional file name) ocr.key

 Loading from ocr.key

  1.  START   - ATG
  2.  STOP    - (TAA OR TAG OR TGA)
  3.  NO-STOP - ([CAG].. OR TT. OR TC. OR TA[CT] OR TG[CGT])
  3.  OCR     - START & ( (NO-STOP){30,} ) & STOP
```

These search KEYs define start and stop codons that could present in a DNA sequence. Further it shows how KEYs can be defined in terms of other predefined keys. The OCR key specifies a region beginning with a start codon, followed by at least thirty triplets before encountering one of the STOP patterns. When used to search the HORMONE.SEQ file this KEY was able to detect all of the known coding regions in those sequences. A partial output is shown below:

```
    BOVPTH2

BOVPTH2
| Hit starts at base 93 of BOVPTH2 and ends at 440
V
       102         112         122         132         142         152         162
ATGATGTCTG CAAAAGACAT GGTTAAGGTA ATGATTGTCA TGCTTGCCAT CTGTTTTCTT GCAAGATCAG
...
       382         392         402         412         422         432       440
ATCAGAAAAG TCTTGGAGAA GCAGACAAAG CTGATGTGGA TGTATTAATT AAAGCTAAAC CCCAGTGA
```

The sequence searching capabilities of *QUEST* can also be exploited for searching sequence files and analyzing the data simultaneously. For example, one can readily search a list of files containing plasmid vectors for those vectors containing an **EcoRI** and a **PstI** sites and NOT a **BglII** site.

Protein secondary structure. Previous examples have shown how *QUEST* might be used to searches databases or selected sets of sequences from nucleic acid or protein databases. *QUEST* may also be used to identify sites in single sequences or related groups of sequences, to aid in making inferences about possible structures of the molecules involved.

Patterns may be used to locate regions of amino acid sequences that are likely to be α-helices or β-strands. Cohen et. al. [5] have described the use of amino acid patterns for the assignment of secondary structure in $\alpha/\beta_{parallel}$ proteins. Taylor and Thornton [9] have located super-secondary structures using patterns that can be expressed using the symbols and conjunctions of ***QUEST***.

In order to use ***QUEST*** for these purposes, patterns are developed to help locate turns in protein sequences. The segments between these turns are then studied for the presence or absence of patterns likely to occur in α-helices or β-strands. The coincidence of such patterns suggests the possible secondary structure assignments for a given segment. ***QUEST*** serves well to find the elementary character patterns typical of helices, sheets, and turns, though many separate runs on the same sequence may be necessary.

Table 5 describes a number of patterns that have been used in secondary structure assignment. Patterns matched for residues 62 to 107 of Flavodoxin are illustrated in Figure 1. Flavodoxin is an α/β protein with a single 5 stranded parallel β-sheet and 4 helices. The actual secondary structure in this region is α 66-74, β 80-87, and α 93-106.

The **TURN** pattern is found at residues 62, 78, 86, 91, and 104, (marked as t3 in Figure 1 since **turn3** matches at all of these residues) due to the presence of a high density of hydrophilic residues matching patterns **turn1**, **turn2**, **turn3**, or **turn4**. The turns found at 87 and 92 are too close for any reasonable secondary structure to lie between.

The **ALPHA** pattern is found at residues 66, 67, 69, 72 to 74, and 94 to 95. Note that the current version of ***QUEST*** cannot report matches for **ALPHA** since it involves counting patterns having the same approximate location in a sequence. This type of pattern is easy to observe in a map or figure, and will be computable in ***QUEST's*** future versions.

The **BETA** pattern is found at 66, 67, 81, 82, and 106. These sites suggest that the first segment might be either α or β. This conflict is resolved as noted below. The second segment has only **BETA** matches and the third only **ALPHA**.

Alpha helices, in some proteins, are strongly suggested by the simultaneous presence of 1) (patterns **HYDROPHOBIC-PATCH-1** and **HYDROPHOBIC-PATCH-2**) hydrophobic residues at positions i, i + 4, i + 7, ... which would lie next to each other if that region were a helix, and 2) (pattern **HYDROPHILIC-STRIPE**) a corresponding hydrophilic region on the opposite face of that helix, and 3) (pattern **CHARGE-PAIR**) a pair of oppositely charged amino acids whose positions on a helix at residues i and i + 3 or i + 4 would lend stability to a helical conformation. The **ALPHA** sites marked in Figure 1 demonstrate the usefulness of simultaneous pattern marking, and reasoning on the basis of finding patterns near each other in a sequence.

In the full assignment algorithm described by Cohen et. al., the apparent conflict in the turn bounded segment 62 to 78 is resolved in favor of the helix. Their algorithm first labels the original sequence with the patterns in Table 5. Other patterns are used to locate strands which might also be edges of β-sheets. Segments are then scored for helix or sheet according to solvent accessibility that

Table 5.
Patterns used in the assignment of secondary structure and turns in $\alpha/\beta_{parallel}$ domains.

Protein Structure Patterns

```
Strong TURN Patterns:

    tphilic     [PGQNSTEDRKH]
    yturnphilic [YPGQNDERKSTH]
    turn1       yturnphilic tphilic{3}
    turn2       tphilic yturnphilic tphilic{2}
    turn3       tphilic{2} yturnphilic tphilic
    turn4       tphilic{3} yturnphilic

    TURN        turn1 OR turn2 OR turn3 OR turn4
                4 HYDROPHILIC AMINO ACIDS IN A ROW.
                TYROSINE MAY BE PRESENT IN AT MOST 1 OF THE 4 POSITIONS.

ALPHA Helix Patterns:

    CHARGE-PAIR [ED]...[KR] OR [KR]...[ED] OR [ED]..[KR] OR [KR]..[ED]
                A PAIR OF OPPOSITELY CHARGED RESIDUES WHICH WOULD BE
                NEAR EACH OTHER ON AN α-HELIX.

    alpha-phobic   [AVILMFWKYC]
                A HYDROPHOBIC RESIDUE LIKELY TO APPEAR IN AN α-HELIX
                (EXCLUDING PROLINE), OR LYSINE, TYROSINE, OR CYSTIENE,
                WHICH ARE NON-HYDROPHOBIC RESIDUES ALSO LIKELY TO
                APPEAR ON A BURIED SURFACE OF A HELIX.

    helix1      alpha-phobic..alpha-phobic alpha-phobic..alpha-phobic
    helix2      alpha-phobic...alpha-phobic alpha-phobic..alpha-phobic
    helix3      alpha-phobic alpha-phobic...alpha-phobic..alpha-phobic
    helix4      alpha-phobic alpha-phobic..alpha-phobic...alpha-phobic
    helix5      alpha-phobic alpha-phobic..alpha-phobic alpha-phobic

    HYDROPHOBIC-PATCH-1  helix1 OR helix2 OR helix3 OR helix4 OR helix5
                A PATCH OF "ALPHA-PHOBIC" RESIDUES ON THE SURFACE
                OF AN α-HELIX.


    P-alpha-phobic  [PAVILMFWKYC]
                LIKE ALPHA-PHOBIC, BUT ALLOWING PROLINE.


    alpha1      ([GSTKAVILFWPYMC].. (P-alpha-phobic){2}.. (P-alpha-phobic))
    alpha2      ((P-alpha-phobic)..[GSTKAVILFWPYMC] (P-alpha-phobic)..
                    (P-alpha-phobic))
    alpha3      ((P-alpha-phobic).. (P-alpha-phobic)[GSTKAVILFWPYMC]..
                    (P-alpha-phobic))
    alpha4      ((P-alpha-phobic).. (P-alpha-phobic){2} ..[GSTKAVILFWPYMC])

    HYDROPHOBIC-PATCH-2  alpha1 OR alpha2 OR alpha3 OR alpha4
                A PATCH OF "P-ALPHA-PHOBIC" RESIDUES ON THE
                SURFACE OF AN α-HELIX, ALLOWING AT MOST
                1 OF GLYCINE, SERINE OR THREONINE IN THE PATCH.

    pstripe1    [PGQNDERKSTH]...[PGQNDERKSTH]...[PGQNDERKSTH]
    pstripe2    [PGQNDERKSTH]..[PGQNDERKSTH]...[PGQNDERKSTH]
    pstripe3    [PGQNDERKSTH]...[PGQNDERKSTH]..[PGQNDERKSTH]
```

```
    HYDROPHILIC-STRIPE  pstripe1 OR pstripe2 OR pstripe3
                3 HYDROPHILIC RESIDUES ARRANGED IN AN APPROXIMATE
                LINE ON A HELIX.

    ALPHA       marked where 3 of 4 of the patterns
                CHARGE-PAIR, HYDROPHOBIC-PATCH-1, HYDROPHOBIC-PATCH-2,
                and HYDROPHILIC-STRIPE occur within 1 residue of
                each other.  All are suggestive of α-helix
                stabilizing formations.

BETA Strand Patterns:

    beta-phobic         [FPMLIVAWC]

    K-or-beta-phobic    [KFPMLIVAWYC]

    central-beta        (beta-phobic){3} OR
                        beta-phobic . beta-phobic{2}
                        beta-phobic{2} . beta-phobic
                3 OF 4 RESIDUES ARE HYDROPHOBIC. TYROSINE AND
                CYSTEINE, THOUGH NOT STRICTLY HYDROPHOBIC, MAY
                HAVE THIS ROLE IN β-SHEETS.

    not-beta-core [~FPMLIVAWYC]

    start-beta    (not-beta-core){2}

    end-beta      (not-beta-core K-or-beta-phobic)

    midbeta1    (K-or-beta-phobic){2}..
    midbeta2    ..(K-or-beta-phobic){2}
    midbeta3    (K-or-beta-phobic)..(K-or-beta-phobic)
    midbeta4    .(K-or-beta-phobic){2}.
    midbeta5    (K-or-beta-phobic).(K-or-beta-phobic).
    midbeta6    .(K-or-beta-phobic).(K-or-beta-phobic)

    beta-region start-beta (midbeta1 OR midbeta2 OR midbeta3 OR
                           midbeta4 OR midbeta5 OR midbeta6) end-beta
                NOTE THAT THE PARENTHESIZED EXPRESSION HERE MIGHT BE
                REPLACED WITH A DENSITY FUNCTION SUCH THAT 2 OF 4
                RESIDUES MUST BE "K-OR-BETA-PHOBIC".

    BETA        (central-beta OR beta-region) AND NOT CHARGE-PAIR
                EITHER OF THE β PATTERNS ABOVE, EXCEPT WHERE THERE
                IS ALSO A CHARGE PAIR WHICH WOULD DESTABILIZE A SHEET
                IN THE HYDROPHOBIC CORE OF A PROTEIN.
```

would be present if that segment were arranged as an ideal of either secondary structure. The α-accessibility score in this region is nearly twice the β score.

Under Development

At this time, *QUEST* is optimized for rapid searching of entire databases for complex patterns. Under development are software tools which will allow the simultaneous searching for multiple patterns in sequences. This will allow more complex patterns to be described that include the coincidence or near-coincidence of simpler patterns. This will be particularly useful in the protein secondary structure assignment problem where the ability of a local region to fold into a structure

Residue		Turn or Secondary Structure	Patterns matched: TURN	ALPHA				BETA
			t3	CP	P1	P2	ST	α	Cβ
58-59		turn							
62	E		t3						
63	E					ST			
64	S						ST		
65	E						ST		
66	F	alpha			P1	P2		α	Cβ
67	E	.					ST	α	Cβ
68	P	.					ST		
69	F	.			P1			α	
70	I	.				P2			
71	E	.					ST		
72	E	.		CP			ST	α	
73	I	.			P1	P2		α	
74	S	alpha					ST	α	
75	T								
76	K				P1	P2			
77	I	turn			P1	P2			
78	S	turn	t3			P2			
79	G	turn							
80	K	beta			P1	P2			
81	K	.			P1	P2			Cβ
82	V	.							Cβ
83	A	.							
84	L	.							
85	F	.							
86	G	.	t3				ST		
87	S	beta					ST		
88	Y								
89	G	turn					ST		
90	W	turn							
91	G	turn	t3				ST		
92	D	turn							
93	G	alpha					ST		
94	K	.		CP			ST	α	
95	W	.			P1			α	
96	N	.							
97	R	.		CP			ST		
98	D	.		CP			ST		
99	F	.							
100	E	.					ST		
101	E	.							
102	R	.							
103	N	.				P2			
104	N	.	t3						
105	G	.							
106	Y	alpha							Cβ
107	G	turn							

Figure 1.
Secondary structure patterns matched in a portion of the sequence of Flavodoxin. t3 = turn3, CP = CHARGE-PAIR, P1 = HYDROPHOBIC-PATCH-1, P2 = HYDROPHOBIC-PATCH-2, ST = HYDROPHILIC-STRIPE, α = ALPHA, Cβ = central-beta. See Table for definitions of these patterns.

may depend on a number of features, some local to and some distant from the site of the structure.

Another feature soon to be implemented in *QUEST* is a type of density function on patterns. A key may be specified that requires a region in a sequence to have some minimum number of matches with a particular pattern or pattern expression. The pattern matching tools in *QUEST* will also be linked to other software to provide a sequence-structure laboratory where patterns may be proposed

and tested against known sequences and structures in order to develop new algorithms for structure assignment. Graphic display of matches will be an integral part of this system.

CONCLUSIONS

We have described the use of ***QUEST*** in searching for patterns in sequences and/or comments in DNA and protein data files. The general pattern matching capabilities make the program very useful for other types of text. Since the scope of a search may be set to be any one of line, page, file, sequence or comments, a text file might be set up to have related data all within one paragraph or page. QUEST then could apply a pattern to the entire page and thus retrieve all of the related information at once.

An example using page scope is searching of a file containing people's names, addresses, and phone numbers, with each individual on a separate page. Searches on any combinations of names, zip codes, area codes, street names, or other fields from the data would retrieve the entire individual's data. This grouping of data might well work for enzyme characteristics and supply information, or for strain lists with descriptions, patent numbers and inventory or cost control information. In this way, a simple paged file may act as an elementary relational database with ***QUEST*** as its retrieval method.

The ***QUEST*** program remains under developement. Facilities for handling simultaneous pattern matching of multiple possibly overlapping targets, has been specified and is now being programmed. Graphic output of match results is also being developed for bit-map type displays. The algorithms used in the ***QUEST*** program will be made available to other software to provide an environment for structure prediction and reasoning. Patterns will be matched against one or more related sequences and consensus patterns will be developed by viewing the results of matches side by side.

Implementation

QUEST was originally developed under the TOPS-20 operating system on a DEC 2060 computer at the USC Engineering Computer Laboratory. The program was written in MAINSAIL ™ [10] and thus easily transported to MAINSAIL environments on VAX/VMS and Sun Microsystems/UNIX Version 4.1c BSD.

REFERENCES

1. Knuth, D. E., Morris, J. H., and Pratt, V.R. (1977) *SIAM J. of Computing* 6:323-350.
2. Knuth, D.E. (1973) *The Art of Computer Programming.* Volume 3: *Sorting and Searching.* Addison-Wesley, Reading, MA.
3. Aho, A. V., and Corasick, M. J. (1975) *Commun. ACM* 18:333-340.
4. Trade Mark, Bell Telephone Laboratories.
5. Cohen, F. E., Abarbanel, R. M., Kuntz, I. D., Fletterick, R. J. (1983) *Biochemistry*, in print.
6. European Molecualr Biology Laboratories (EMBL) Nucleotide Sequence Data Library. Heidleberg, Germany.

7. GenBank™ is a trademark for the Genetic Sequence Data Bank established by Bolt Beranek and Newman Inc. and Los Alamos National Laboratory under contract with the National Institutes of Health.
8. Brutlag, D. L., Clayton, J., Friedland, P. and Kedes, L. H. (1982) *Nucleic Acids Res.* 10:279-294.
9. Taylor, W.R., and Thorton, J. M. (1983) *Nature* 301:540-542.
10. Wilcox, C. R., Jirak, G. A., and Dageforde, M. L. (1979) *Mainsail Implementation Overview*, Xidak Inc.