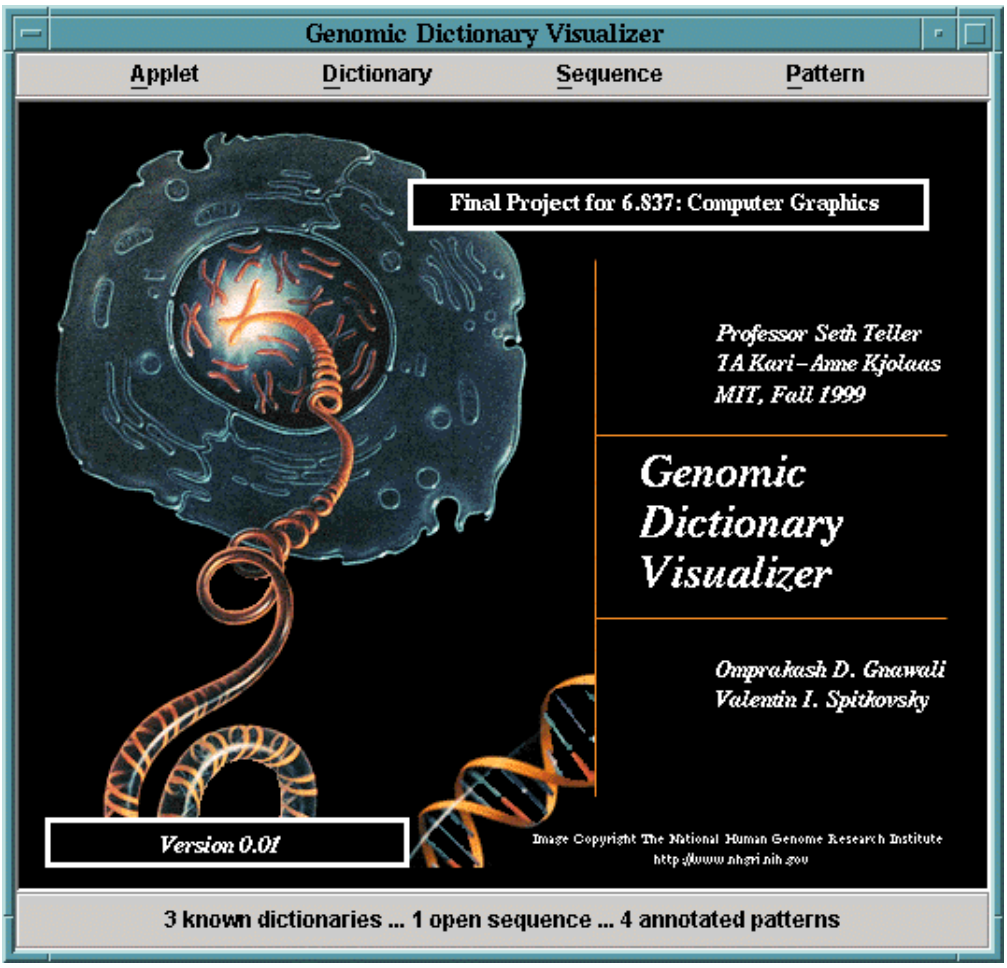
6.837: Computer Graphics (Omprakash D. Gnawali & Valentin I. Spitkovsky)

A Visualization for the Dictionary Approach to Gene Annotation

Report

Abstract



Five intensive weeks of brainstorming and coding culminated in a stand-alone Java applet: the Genomic Dictionary Visualizer (GDV). The GDV tool should prove to be useful to researchers involved in computational biology. It facilitates quick and simple visualization of interesting regions of arbitrary genetic material. While a lot of features are missing, GDV's fundamental contribution is its completeness: all of the basic functionality is there and the framework for adding features exists. To the authors' knowledge, GDV is the first and only tool for visualizing genetic sequences.

Top.

Introduction

Originally begun in 1990, the U.S. Human Genome Project is coordinated by the U.S. Department of Energy and the National Institutes of Health. Conceived as a 15-year endeavor, rapid technological advances have accelerated the project to an expected completion date of 2003. Project goals are to 1) identify all the estimated 80,000-100,000 genes in human DNA; 2) determine the sequences of the 3 billion chemical bases that make up human DNA; 3) store this information in databases; 4) develop tools for data analysis; and 5) address the ethical, legal, and social issues (ELSI) that may arise from the project. Once the genes are identified and their nucleic sequences are determined, researchers will have to tackle new and more difficult problems to understand what it is that these genes do. To get a feel for

just how difficult this problem is, consider the modern central biological dogma:

1. genes (DNA) are transcribed into RNA by the enzyme RNA polymerase
2. RNA transcripts are subjected to post-transcriptional modification
3. mRNA molecules are translated by ribosomes
4. newly synthesized proteins are often modified after translation
5. the protein carries out its function

Not all DNA is expressed as protein. Sections of non-expression are termed introns. Introns are excised by spliceosomes during the pre-mRNA phase, and are not found in the final mRNA product. Exons are all that remain of the original DNA message upon completion of transcription. Clearly, even knowing all the nucleotides that make up a gene is not enough to predict a protein that it codes for.

Gene Recognition

In light of these complications, gene recognition emerged as a subfield of computational biology. Researchers in this area attempt to design algorithms and heuristics which would eventually compile to efficient computer programs for variations of the following problem: given a large sequence of genetic material (read: a string of hundreds of thousands of nucleotides)

1. Locate the promoter regions (these occur just before the genes).
2. Identify the genes.
3. Identify the exons and the introns within each gene.
4. Predict the initial amino acid sequences.
5. Predict the proteins made by each gene.

Gene Annotation

Of particular interest in the gene recognition problem is the subproblem of gene annotation. Suppose that we had access to a reliable promoter detector. Recall that promoters are short regions of DNA that occur close to the beginning of the gene. They are the chemical signals which alert the biological machinery that there is a gene to be expressed and specify how much of the protein should be made. Having identified the promoters in our large chunk of DNA, we could then split that DNA into shorter segments, each beginning with the promoter region and ending just before the next promoter region. We would then be confident that the resulting segments of DNA contain exactly one gene. The next task would be to annotate that gene: to specify which segments are introns (to be cut out) and at which point the cell begins and ends its translation of the concatenation of the remaining segments (exons). Of tremendous help in this task are several biological facts:

1. Translation begins after an ATG.
2. Translation ends at the first stop codon (one of {TAA, TAG, TGA}).
3. Introns start with a GT.
4. Introns end with an AG.

This limits the number of possible annotations or ‘‘parses’’ of a gene substantially, but does not reduce the solution space to a single nice crisp answer. As a result, there is a need for good tools for evaluating or scoring a parse. Armed with an arsenal of such tools, a gene annotation algorithm could intelligently decide between parses and predict the ‘‘best’’ one. Even if the level of gene annotation were not good

enough for the gene recognition problem, it would still be extremely useful. One of the main challenges facing biologists today is the accurate annotation of newly sequenced genomic data. If the annotating algorithm could latch on to more than just a handful of the most basic signals, it would save a lot of hours of human work. Available for this task are extremely large databases of proteins, expressed sequence tags (roughly, snippets of coding regions of DNA), and much smaller databases of annotated genes.

Dictionary

One tool already developed is a fast and fully automated dictionary for looking up genomic information. Qualitatively, given a brand new sequence, the dictionary answers the question ‘‘Have we ever seen anything like it before?’’ Quantitatively, the answer to that question is a list of all matches (longer than a cutoff value) between the substrings of the input sequence and the substrings of all the sequences on which the dictionary was based.

Implementation

All of the dictionary code is in the C++ programming language, written and tested under the Linux operating system. It was designed and optimized to utilize the resources of the host computer as efficiently as possible. The alphabet used by the dictionary is entirely user-defined. Combined with the fact that there are no restrictions on lengths, numbers, and types of sequences in the dictionary, the uses for it are virtually endless.

Exon Prediction & Gene Recognition

Exons, the coding regions of DNA, are under high selective pressure to not mutate very much. As a result, large pieces of coding DNA are conserved between various proteins and even species. Therefore, one could build a dictionary of the known exons or proteins. Faced with a new sequence, using the dictionary, one could quickly find all the segments in it that match reasonably long pieces of known exons or proteins. These portions of the new sequence could be safely labeled as ‘‘exon,’’ reducing the parse search space and yielding hints to the origin of the sequence. Introns, the non-coding regions between exons, are under much less of a selective pressure to conserve themselves: most changes in introns go unnoticed. Therefore, finding a reasonably long match between an exon and an intron is highly unlikely.

Repeat Masking

One could easily build a dictionary from repeat databases. Repeats are long sequences of DNA that occur very frequently in introns and very rarely in exons. A dictionary could then be used to rapidly find repeat segments in genes. This technique provides an alternative to alignment based repeat maskers such as RepeatMasker. The method is especially useful for exon prediction and gene recognition and annotation, where it is advantageous not only to mask complete repeats, but to mask segments (perhaps from repeats) that do not occur in exons.

Finding Related Sequences

The dictionary approach offers an alternative algorithm for finding similarities between sequences. The most widely used program developed for this purpose is BLAST, which is an alignment tool; it is often manually applied for the purposes of gene annotation. Alignment based algorithms do not require the

matches to be exact. On the other hand, they do run into problems because they rely on almost arbitrary scoring functions in their dynamic programming routines. Also, the notion of alignment altogether eliminates many non-consecutive matches. Consider the strings AAAAABBBBBCCCCC and CCCCCBBBBBAAAAA. An alignment algorithm would find at most one set of segments that match between these two strings. A dictionary approach would find all three.

Demand for Visualization

The dictionary generates a lot of information, which can be passed on to heuristics that implement the artificial intelligence and statistical analysis which, using the raw dictionary hit data, solve high-level gene recognition subproblems. The data shuttled between these abstraction layers is very rich in information. Some of this information may be lost or masked by the high-level programs -- it would be very useful if the researchers could have access to this knowledge base. While it is possible to leave a trace of the hits found by the dictionary in the form of a text file, such communication of information is inefficient, from the human point of view: the amount of data is huge and it is virtually impossible to extract any insights for more than a few isolated regions.

A visualization tool which could depict the similarities between sequences, point out known biological signals, allow context switching between sequences on-the-fly, and just simply display rudimentary statistics is greatly needed. This demand is fueled by researchers specializing in gene recognition: if they could visualize the information their algorithms are getting, they could improve their own understanding of how those algorithms behave. That would be useful for drawing insights and for debugging. More importantly, this visualization tool would come in handy for many biologists. Having gotten a sequence hot off the press, he or she may want to know what it is he or she is looking at. Combining the GDV with some known dictionaries, the biologist could very quickly pull up the sequence on the screen and highlight the areas of activity. This would yield some clues as to what the researcher is dealing with. Parts of the genetic sequence may match some known proteins, and looking at the names of the matching sequences may even hint at the function of genes within the data.

As far as the authors know, such visualization tools either do not exist or aren’t widely available. Yet from a computational point of view, the functionality sought is far from overwhelming. While the project may not have much to do with graphics per se, it is a perfect example of scientific visualization, a viable option for the final project. The tool in question has everything to do with the problem of effective communication of a wealth of knowledge.

Top

Goals

Our intent was to rectify the problem of a missing means of communication of biological sequence data to the scientific community. We set out to solve the problem through a visualization by utilizing a very high-bandwidth communication channel we humans have: our eye-sight. The challenge we faced was finding a fast and elegant way to deliver the important information in an understandable way, while keeping the visualization intuitive and easy to use. We have been able to split our task into several logical components or goals:

- Abstract representation of a single sequence:

Global view of the sequence, local views of the sequence -- ability to zoom in on actual characters, annotation of interesting user-defined patterns -- for instance the ATG and the stop codons (in the case of DNA), annotation of interesting user-defined regions -- perhaps the user already knows that some portions of the sequence are exons or introns (in the case of DNA), etc.

- Abstract representation of a query result:

  Initially, this data is a huge stream of text. We planned to set up filters, to process the information only at the desired resolution. The user may be interested in viewing only a very small subset of the result, and the abstract visualization could run very fast, as there won't be much real geometry to render. The abstract representation had to provide insights into the hidden properties of the hits.

  We intended to explore varieties of abstract representations to find a subset suitable for our task. Some basic thoughts: graphing the number of hits through locations of the input sequence, somehow incorporating the lengths of the hits into the chart, highlighting the sources of the hits if multiple dictionaries are queried, using color.

- Abstract representation of just two sequences:

  The plan was to enable the user to identify a particular match, take a look at the sequence from which the match came, and compare that sequence to the input sequence at a very high level of detail. An interesting representation could include a two-dimensional image with the sequences' positions on each of the axes and the color (or surface value) at each location evaluating to some informative function. A simple function could be a binary 0/1 value based on whether or not the characters at the two positions match; then the easily identified diagonal streaks would represent subsequence matches. A more complicated function could be something like the number of matches of some length that go through both of these locations.

- Interaction:

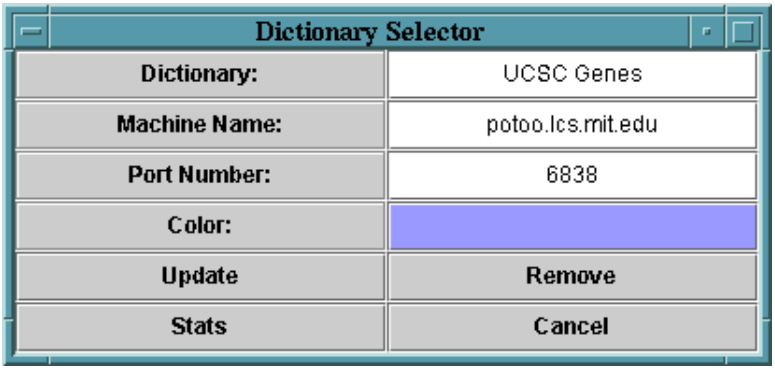
  While the visualization tool would be incredibly useful even in its static form, the users could benefit tremendously if empowered to interact with it dynamically. Letting the user play with the parameters -- perhaps turning a knob hooked up the the hit cutoff level -- and immediately see the effects of the actions could bring about further insights. Being able to see clusters of hits spring up at various areas as the hit cutoff is altered may alert the user and direct the user's attention towards the interesting regions. Something like this could even be animated.

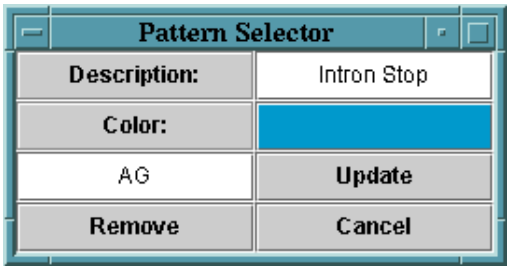
Our most important goal, however, was to end up with a finished product. In our view, it was more important to produce an intuitive, usable, stable, stand-alone tool, however simple, rather than a highly complex contraption with broken features.

Because there was a potential for visualizing a lot of data, we stressed that the tool must be fast; at the same time, the usefulness of the tool depends on its accessibility. With these concerns in mind, we choose to implement the visualization tool as a Java applet communicating with the dictionaries over TCP connections; all of the computationally intensive crunching was to be done on the host computer running the optimized C++ binaries. The applet itself was to be kept as light as possible, to maintain adequate speed. The graphics need not be glamorous: they need to be informative, efficient, and fast.

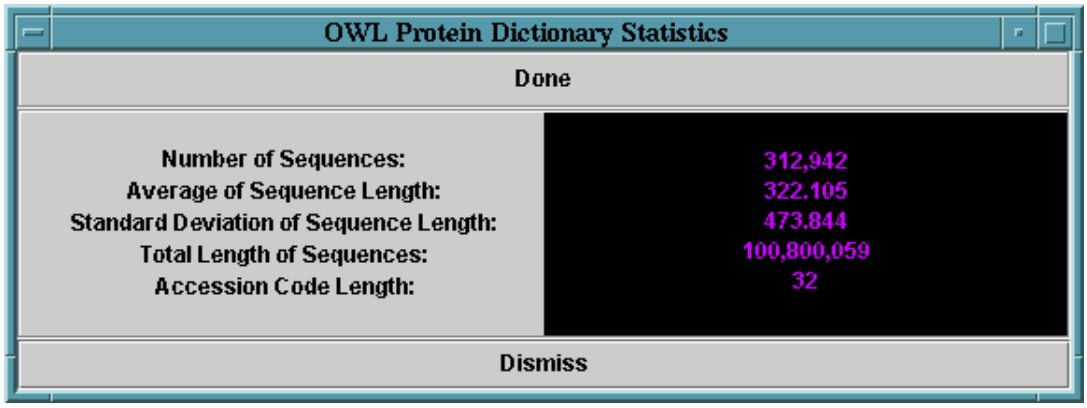
Achievements

Despite the challenge to produce a project in a matter of some four or five weeks, we managed to meet our most crucial goal: to produce a product capable of stand-alone usage. The Java applet we ended up with is as bare as could be, yet it is complete. It contains most of the basic functionality we set out to give it.

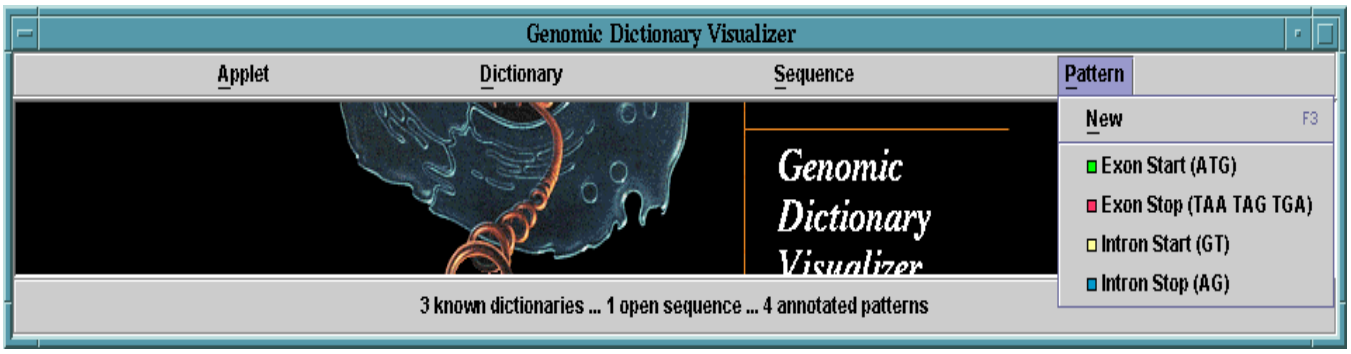


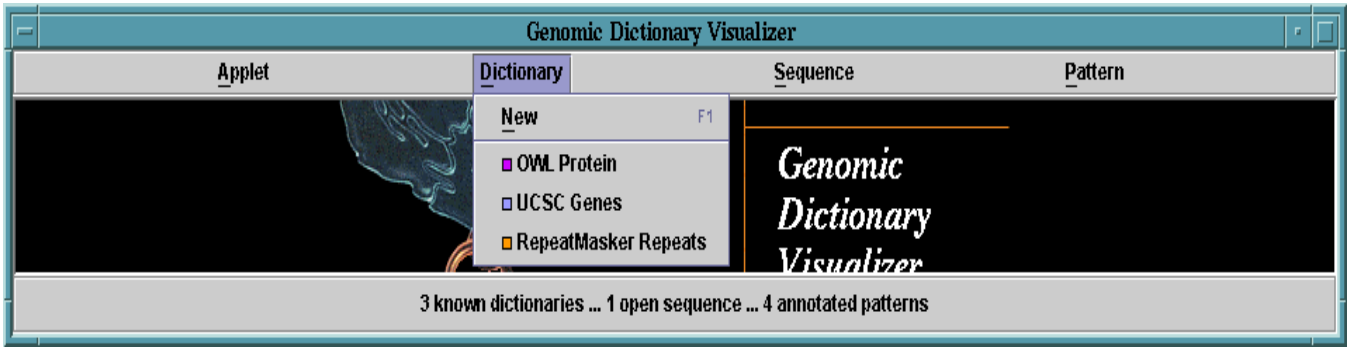


There is an easy way to quickly gather statistics about a known dictionary:

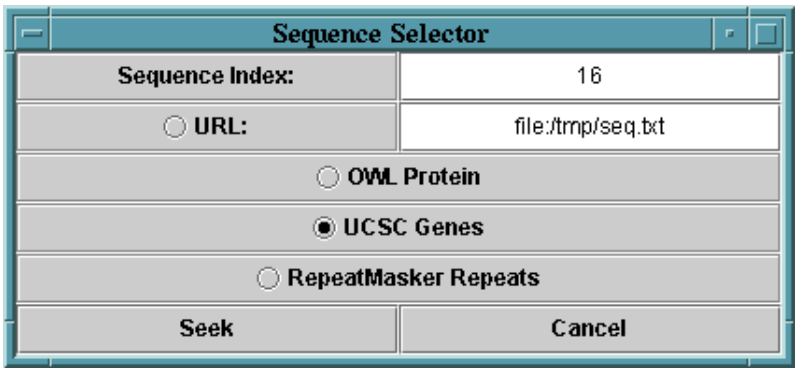


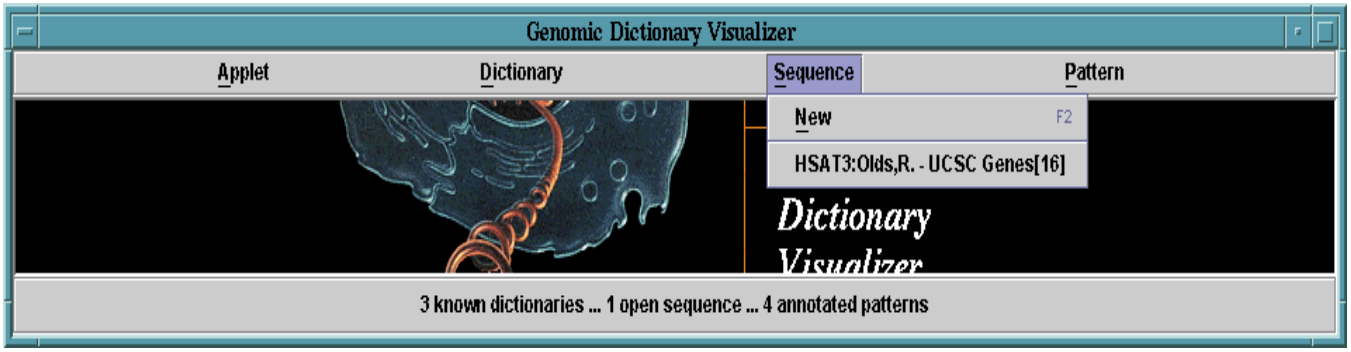
The applet maintains an dynamic base of known dictionaries and patterns to highlight:





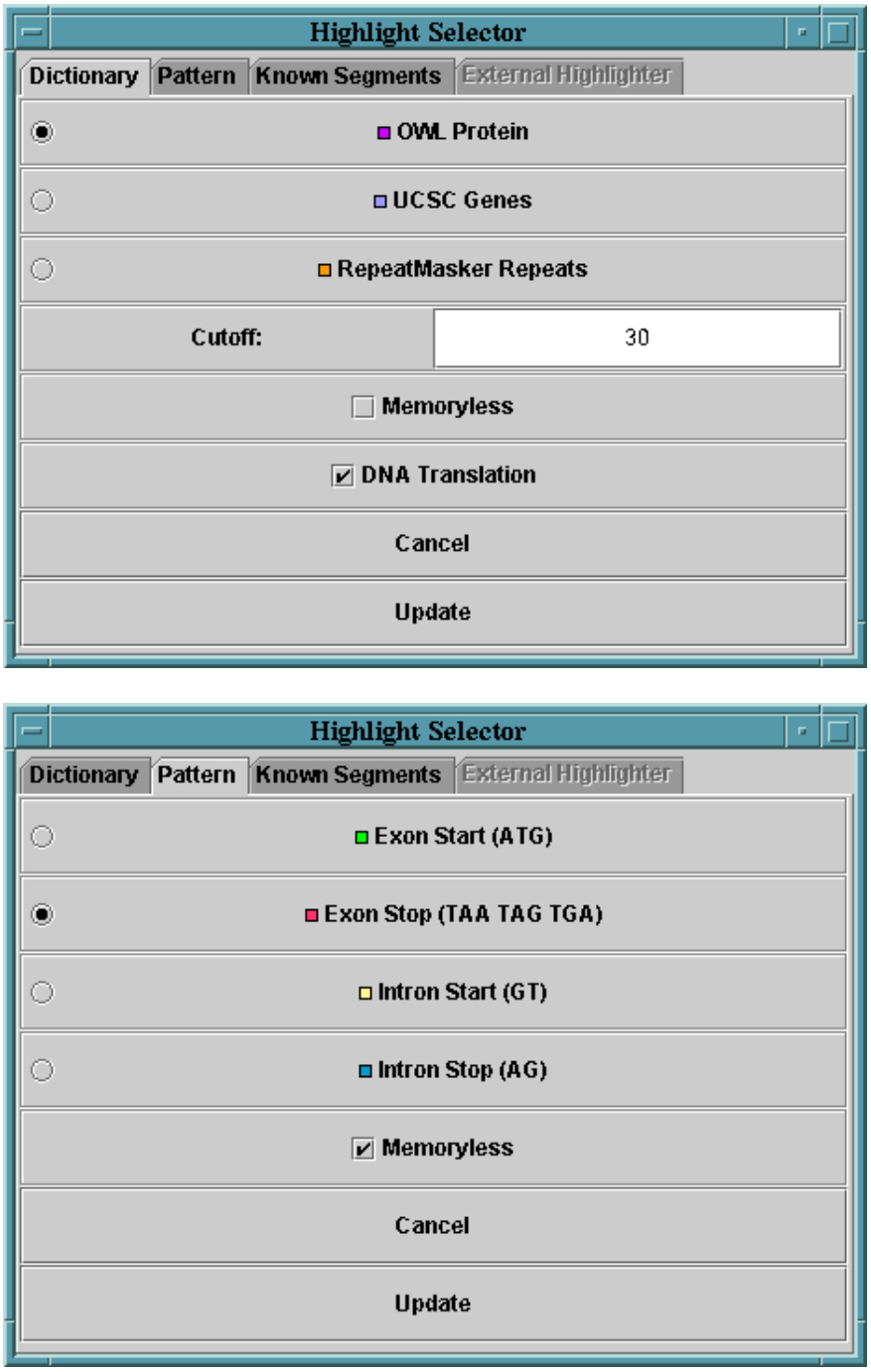
Each dictionary and pattern is associated with a color which is later used to highlight the appropriate regions in loaded sequences. The user can open multiple sequences at once, pulling them out of either known dictionaries or files / urls:

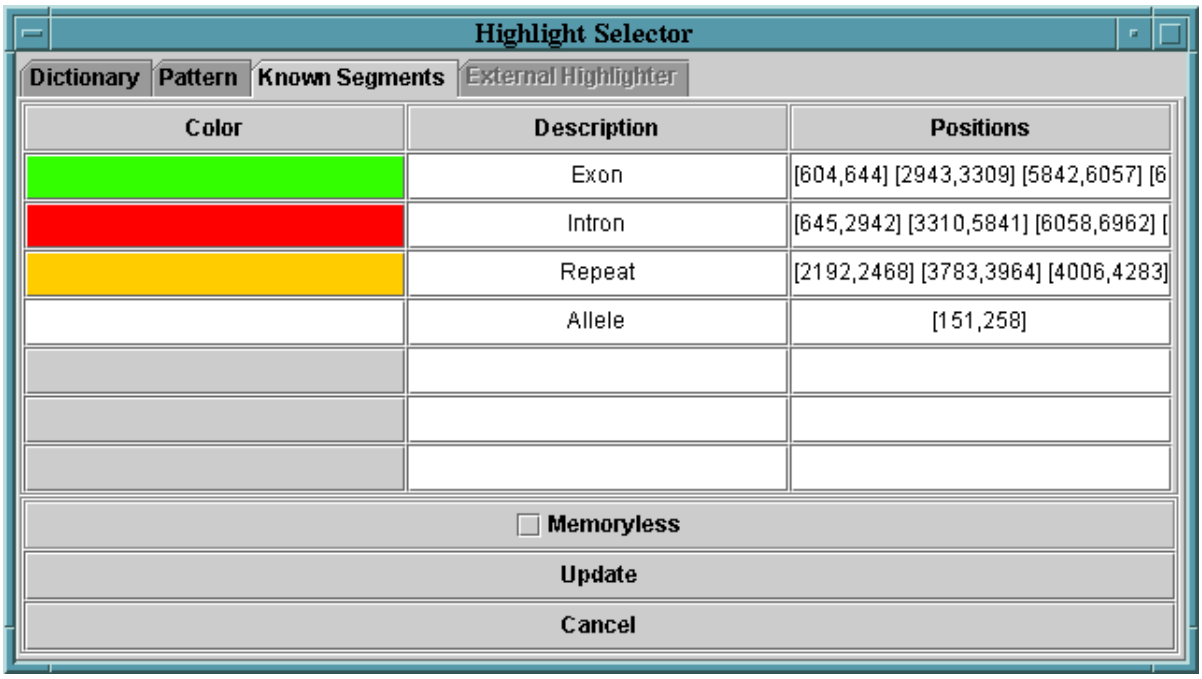




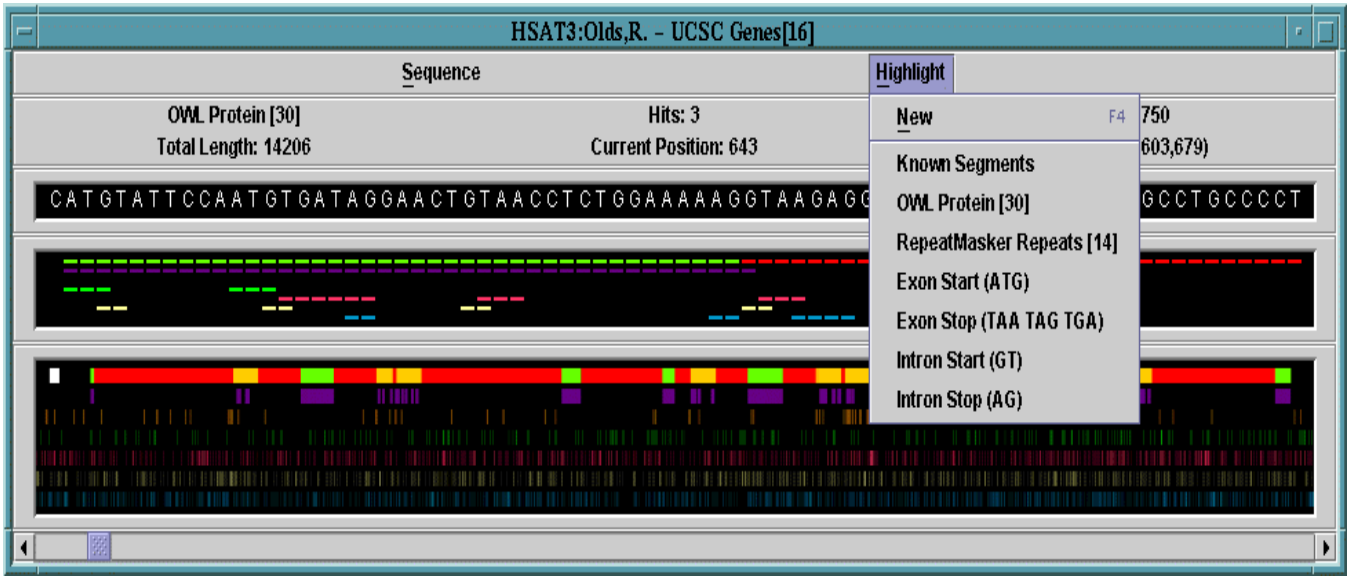
In addition to known patterns and dictionary queries, a sequence can be annotated via known segments

if some of the structure has been solved:

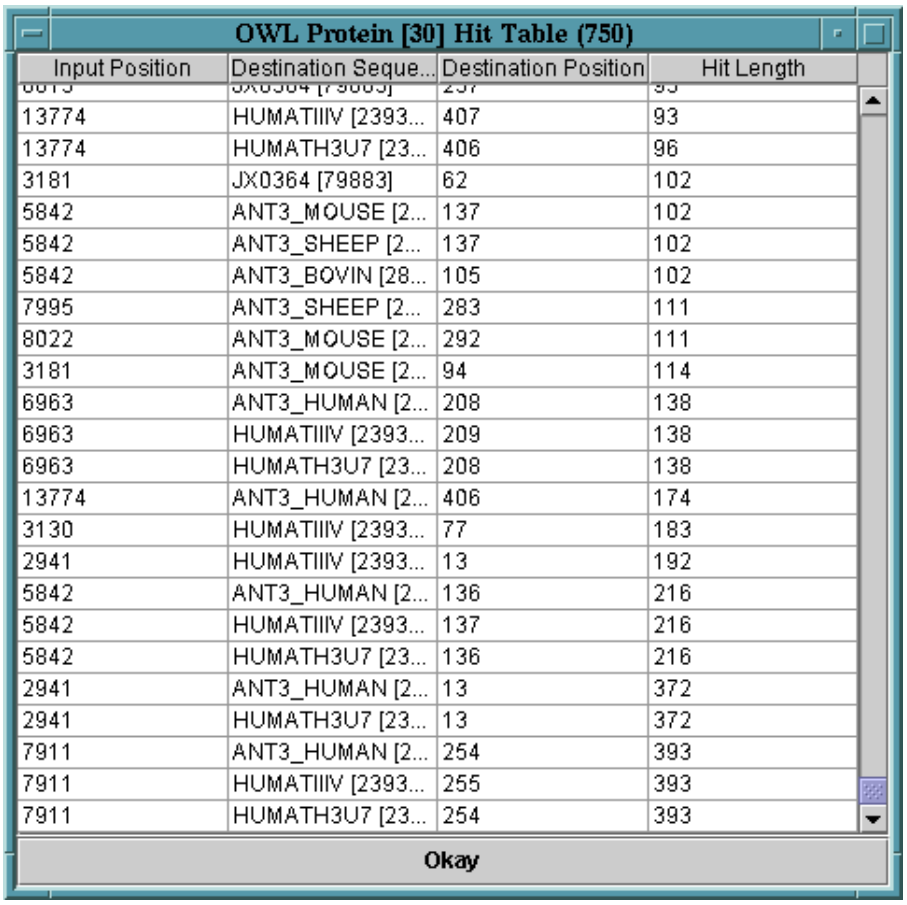




The user is free to add highlights to a particular sequence to light up interesting areas of activity:



The sequence depicted above is a known gene (HSAT3), see appendix. Using the known segments interface, the we've marked the exons, introns, and repeats in this sequence. We also highlighted the standard patterns: ATG, { TAA, TAG, TGA }, GT, and AG. In addition, we screened this sequence against two known dictionaries of proteins and repeats. Observe the close correspondence between areas of similarity and actual exons and repeats. The user is free to load a list of the actual matches -- the program pulls up a table of hits -- which the user can sort by all the values in any of the columns. To a biologist, the names of the sequences with long matches actually suggest the function performed by the proteins encoded for by the exons of this particular sequence:



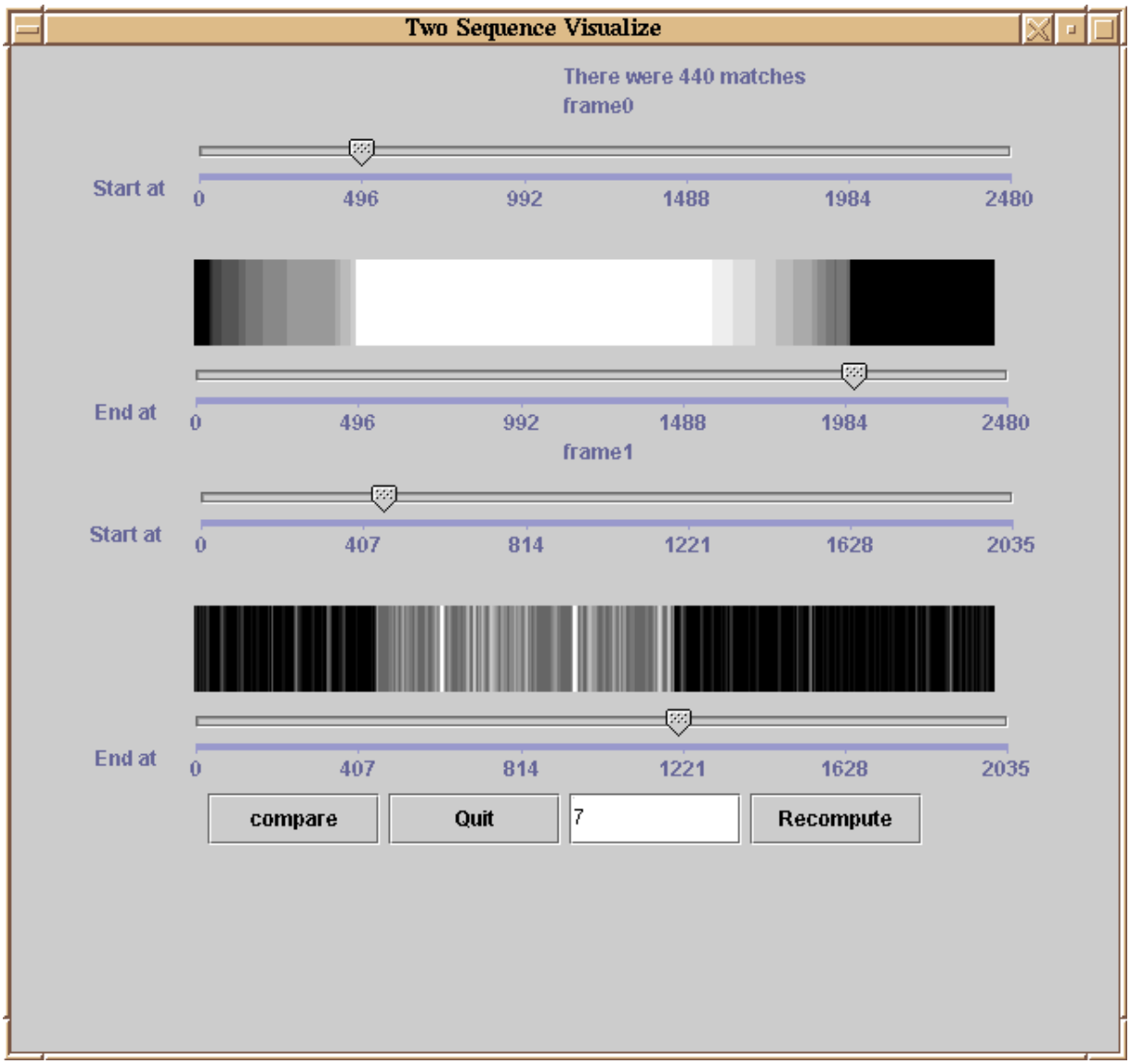
OWL Protein [30] Hit Table (750)

Input Position	Destination Seque...	Destination Position	Hit Length
13774	HUMATIIIV [2393...	407	93
13774	HUMATH3U7 [23...	406	96
3181	JX0364 [79883]	62	102
5842	ANT3_MOUSE [2...	137	102
5842	ANT3_SHEEP [2...	137	102
5842	ANT3_BOVIN [28...	105	102
7995	ANT3_SHEEP [2...	283	111
8022	ANT3_MOUSE [2...	292	111
3181	ANT3_MOUSE [2...	94	114
6963	ANT3_HUMAN [2...	208	138
6963	HUMATIIIV [2393...	209	138
6963	HUMATH3U7 [23...	208	138
13774	ANT3_HUMAN [2...	406	174
3130	HUMATIIIV [2393...	77	183
2941	HUMATIIIV [2393...	13	192
5842	ANT3_HUMAN [2...	136	216
5842	HUMATIIIV [2393...	137	216
5842	HUMATH3U7 [23...	136	216
2941	ANT3_HUMAN [2...	13	372
2941	HUMATH3U7 [23...	13	372
7911	ANT3_HUMAN [2...	254	393
7911	HUMATIIIV [2393...	255	393
7911	HUMATH3U7 [23...	254	393

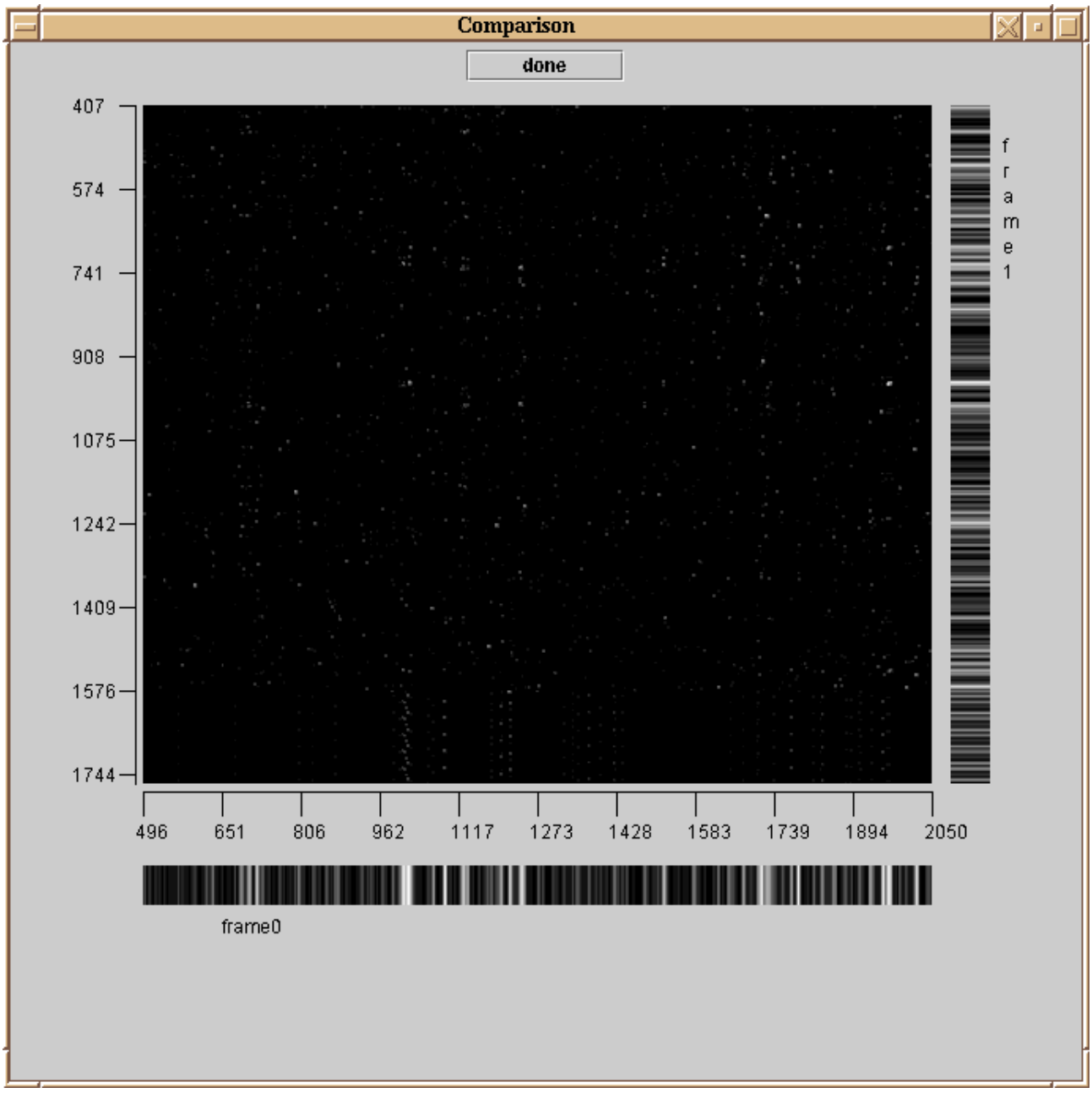
Okay

Finally, the user can easily pull out a matching sequence and run sequence-to-sequence comparison on the two strands:

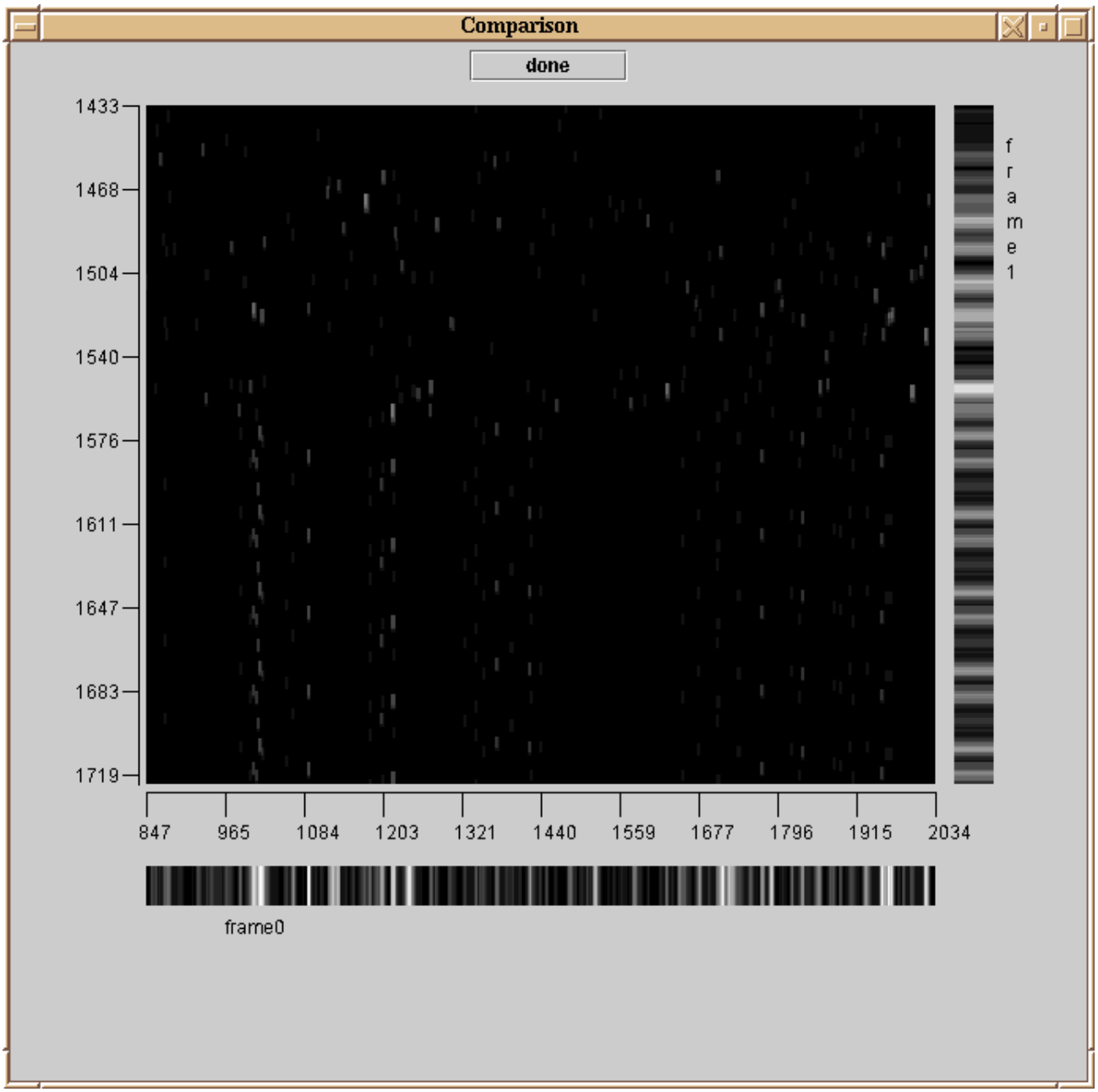
Once the user specifies the sequences to compare, the user can select a portion of each of the images to do mutual hit comparision. Grayscale levels on each panel represent the relative number of hits for that particular location for that sequence. The user is also able to interactively change the cutoff length and see the number of hits vary. When the user changes the cutoff and hits the recompute button, the grayscale image changes dynamically displaying increasing or decreasing number of hits.



The user is presented with a two-dimensional hit matrix display where grayscale intensity at a particular position represents the relative number of matches at that position between the two sequences along horizontal and vertical axes. Sparse white dots mean very little similarity between the sequences, and the matches are very short. Small Dense white dots represent lots of small hits while large blocks of grayscale box represent long matches. The horizontal and vertical sidebar shows the accumulated hits for that sequence in that position.



The user is able to zoom in and zoom out to study the area of interest. Here is a closeup of a hit dataset.



Our final product works; it is a simple implementation of functionality that is demanded. Yet it is disappointing how few features we actually managed to implement in the short time we had. Some blatant omissions come to mind: animations of the amount of activity in a region as a function of the cutoff length, various representations of the amount of activity -- not just a gradient of colors -- graphs, nonlinear scaling of the graphs / colors, etc. A million ideas come to mind just thinking of the table of hits: we did implement the sorting of hits, but there are other features just begging to be implemented. For example, clicking on a hit could move the local viewing region of the sequence to the position where the hit occurs, information could be gathered not just based on hits but incorporating their sources: which sequence produced the most / longest hits? Which sequence had the most total / consecutive area covered? Two-sequence representation could get away from flatland and attempt to draw surface maps with peaks where the sequences are similar. The list goes on and on. We didn't get a

chance to allow for user-defined highlights, which would let a user plug in his or her own class file for annotating the sequence. However, we developed a framework for a HitSet class that would make such a future extension very simple. This would be useful for annotating structures that have variation (for example, pyrimidine tracks -- regions that are rich in G's and C's -- that occur frequently near the ends of introns) as well as the presentation of various statistical analyzers (for instance frame-tests which compute conditional probabilities -- if this segment were in an exon, what is the probability that it would be in frame 0, frame 1, or frame 2).

A basic aim that we failed to meet was to put the applet on the web. To do this, we needed to learn about jar files and how to sign applets and to create certificates to allow users to set up their browsers to trust our product. Despite these holes, we are not too upset. The number of ideas on our wish-list is growing and we are psyched enough about the project that we plan to continue working on it over IAP to produce a more robust and a more useful applet, getting away from version 0.01.

The main problem with our product is that it just sort of evolved without a real thought-through design. While we wanted to think things over before coding them up, we didn't get a chance to do that because of the limited time. We simply could not afford to spend two weeks designing the thing and have zero lines of code until then. We started work immediately after the last homework assignment and still had no shortage of stressful nights all the way till the end of the term. Before future work on this project is undertaken, the entire system should be scrapped and redesigned. However, the benefit of version 0.01 is that it highlighted the areas that need attention. They are:

First of all, the networking model. This is something that had no relevance to the class, so we made an effort to get it out of the way as soon as possible. As of now, we've got a very simple blocking server in C++ which serves as a wrapper for the dictionary code RPC. We also have a Java client model for interacting with the server. The server could be made more fault-tolerant and more reliable. The client is mostly fine, but it would be nice to come up with a good interface for passing data (right now, we are just passing strings) which would utilize the bandwidth fully; perhaps compression is in order. Speed is a problem.

Second, the user interface needs to be designed with the biologist in mind. Although we wanted to, we didn't get a chance to ask our friends in course seven for what they would want in a visualization tool -- the sort of representations of genetic material they found most expressive in their bio books, etc. We had enough trouble just trying to get the GUI to work, seeing as how neither of us programmed GUI's before.

Finally, we suspect that our usage of the Java model is highly suboptimal. Our interfaces probably generate objects right and left, sucking off memory and valuable resources from the actual visualization methods. Now that we have some idea of how Swing works, redesigning the code could produce a faster and more light-weight program.

The major source of both help and trouble was Java. The great things about it are the tremendous wealth of specifications, example code, and tutorials available on the web for this language. It was nice not to have to worry about platform-dependent issues and just to know that our code will run anywhere. It was also great to avoid dealing with motif and unix network programming, for the most part. Finally, it was very useful to be able to go home for Thanksgiving and to continue to work on the project there, now running the applet in Windows instead of Linux. But the troubles with Java almost outweighed the benefits. It's buggy... it's slow. Compiling the code takes forever, even with well-designed makefiles.

Running it takes even longer. Various bugs creep up, driving you nuts trying to pin them down, only to find out that everything works fine on a different platform, meaning that it's a Java bug. Some features of our program fail on the Suns -- sometimes the networking blocks, sometimes the menus don't get drawn correctly. On the SGI's, the networking almost always fails and the last menubar is always drawn wrong. In Linux, the program just hangs every now and then. Surprisingly enough, Windows has been the most stable platform... We can only hope that Java 1.3, to be released soon, will fix most of these bugs. When will Java compile to faster code is another question.

Top

Individual Contributions

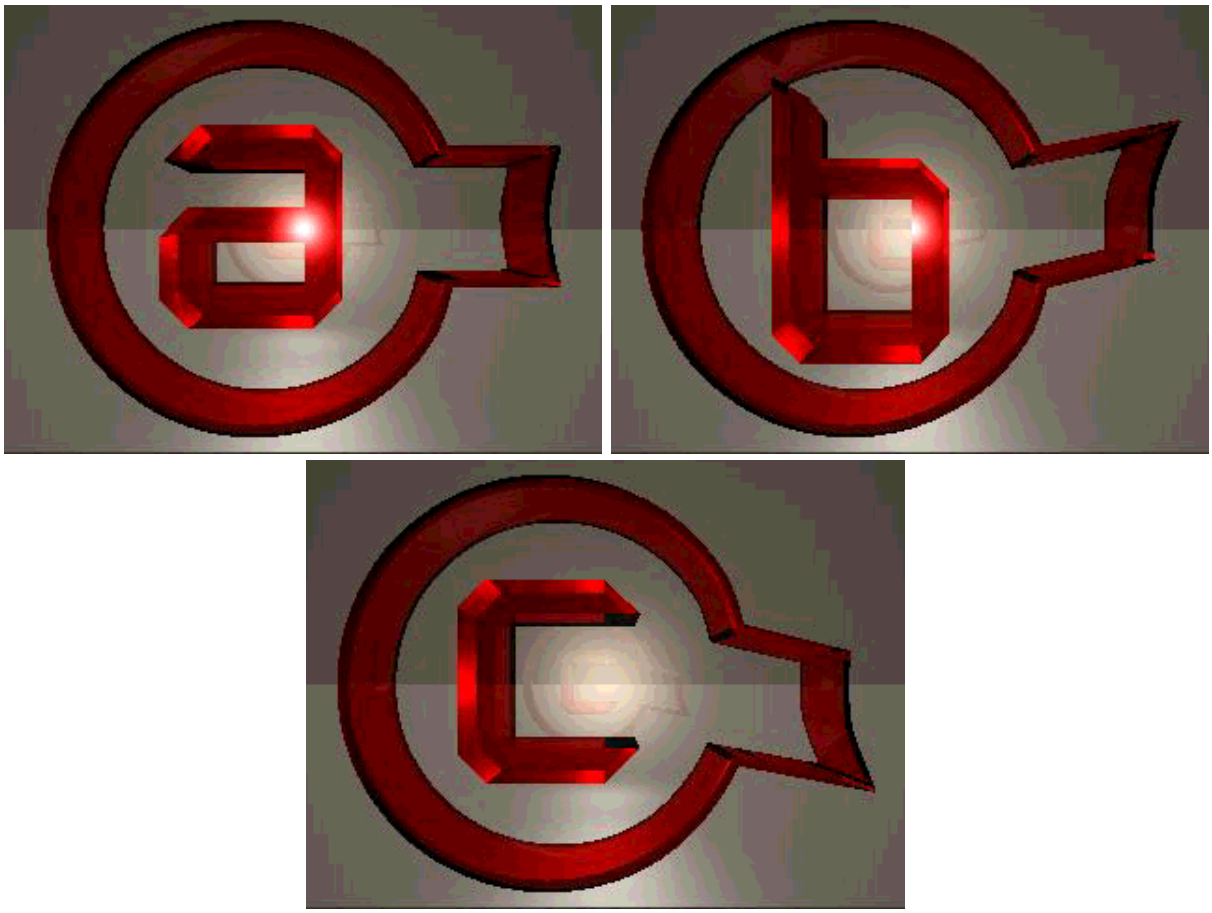
Om worked on the code for sequence-to-sequence visualization and also created the help and about dialogs. Val wrote everything else. Both teammates brainstormed possible representations. We really felt the impact of a missing teammate...

Top

Lessons Learned

While we didn't write a flexible user interface, we certainly learned enough to write a better one next time. Some of the lessons that were reinforced included the idea that the first prototype will be thrown away -- when we redesign the applet for its next version, we will probably have to rewrite most of the code from scratch. As mentioned above, the hardest and easiest problems came from Java. Its speed and bugs really crippled what we could implement and how quickly we could implement it. At the same time, it really eased our life by providing us with ready-made code for GUI's through its Swing interface.

By a complete accident, we learned a bunch about fonts. We wasted the first week on the idea that our sequence representation should look cool. We wanted to render the text using ray-tracers, incorporating shadows, specular highlights and all. We even considered using the color of the shadows and the color of the text plus the color of the background to convey information about a particular character. In the process of looking for fonts, we discovered a wealth of them at sites like http://fontz.de. However, most of these fonts are true-type -- we found programs that convert ttf to bdf, bdf to pcf, etc. We found a program called font3d which creates include files for a shareware raytracer POVRAY, given a string to render and a truetype font. We even found a monospace molecular font: here are our rendered red letters a, b, and c.



But the problem was that the images took a long time to load into the Java program and didn't look so good when they were reduced to a size of a text character. Finally, we settled on a less glamorous but more efficient solution. But now we know a bunch about dealing with fonts and even a bit about using fonts with raytracers.

Other new knowledge we reaped: user interface design, Java Swing, some networking, etc. Most importantly, we gained valuable team project experience, which will improve our performance in a future team project. We learned a bit about managing a small team faced with a high-paced open-ended task.

Top.

Acknowledgments

The most useful resources have been our TA Kari-Anne Kjolaas who guided us via weekly checkpoint meetings, a great search engine Google which let us find relevant info without fail, an extensive Java API Reference and a set of Java Tutorials. Finally, we ended up using a modified version of the image from the The National Human Genome Research Institute as a front-page for our program.

Top.

Conclusion

A new visualization tool is now available for highlighting common regions between sequences. The standard applications for the dictionary tool are augmented with a graphical user interface: a high-bandwidth channel for absorbing the results. The implementation is relatively fast, light, elegant, and informative. It should be extremely useful to biologists in general and to researchers in computational biology, especially in the fields of gene recognition and gene annotation.

Top.

Appendix

This is the entry for UCSC Gene #16 (HSAT3) which we visualized using GDV.

```
LOCUS       HSAT3       14206 bp    DNA             PRI       03-NOV-1994
DEFINITION  H.sapiens gene for antithrombin III.
ACCESSION   X68793 S52236 S52240
KEYWORDS    antithrombin; antithrombin III gene; AT3 gene; plasma protein;
            serine proteinase inhibitor; serpin.
SOURCE      human.
  ORGANISM  Homo sapiens
            Eukaryotae; mitochondrial eukaryotes; Metazoa/Eumycota group;
            Metazoa; Eumetazoa; Bilateria; Coelomata; Deuterostomia; Chordata;
            Vertebrata; Gnathostomata; Osteichthyes; Sarcopterygii; Choanata;
            Tetrapoda; Amniota; Mammalia; Theria; Eutheria; Archonta; Primates;
            Catarrhini; Hominidae; Homo.
REFERENCE   1  (bases 1 to 14206)
  AUTHORS   Olds,R.
  TITLE     Direct Submission
  JOURNAL   Submitted (12-OCT-1992) to the EMBL/GenBank/DDBJ databases. R.
            Olds, Institute of Molecular Medicine, John Radcliffe Hospital,
            Oxford OX3 9DU, UK
REFERENCE   2  (bases 1 to 14206)
  AUTHORS   Olds,R.J., Lane,D.A., Chowdhury,V., De Stefano,V., Leone,G. and
            Thein,S.L.
  JOURNAL   Biochemistry In press
REFERENCE   3  (bases 1 to 14206)
  AUTHORS   Bock,S.C., Marrinan,J.A. and Radziejewska,E.
  TITLE     Antithrombin III Utah: proline-407 to leucine mutation in a highly
            conserved region near the inhibitor reactive site [published
            erratum appears in Biochemistry 1989 Apr 18;28(8):3628]
  JOURNAL   Biochemistry 27 (16), 6171-6178 (1988)
  MEDLINE   89050967
REFERENCE   4  (bases 1 to 14206)
  AUTHORS   Olds,R.J., Lane,D.A., Ireland,H., Leone,G., De Stefano,V.,
            Wiesel,M.L., Cazenave,J.P. and Thein,S.L.
  TITLE     Novel point mutations leading to type 1 antithrombin deficiency and
            thrombosis
  JOURNAL   Br. J. Haematol. 78 (3), 408-413 (1991)
  MEDLINE   91337920
COMMENT     related sequences: M21636-M21645.

            NCBI gi: 28906
FEATURES             Location/Qualifiers
```

```
source          1..14206
                /organism="Homo sapiens"
                /chromosome="1q 23-25"
allele          152..259
                /note="polymorphism"
                /citation=[3]
mRNA            join(<605..645,2944..3310,5843..6058,6964..7101,
                7912..8302,10336..10400,13775..>13951)
                /gene="AT3"
exon            <605..645
                /gene="AT3"
                /number=1
CDS             join(605..645,2944..3310,5843..6058,6964..7101,7912..8302,
                10336..10400,13775..13951)
                /gene="AT3"
                /note="NCBI gi: 28907"
                /codon_start=1
                /product="antithrombin"
                /translation="MYSNVIGTVTSGKRKVYLLSLLLIGFWDCVTCHGSPVDICTAKP
                RDIPMNPMCIYRSPEKKATEDEGSEQKIPEATNRRVWELSKANSRFATTFYQHLADSK
                NDNDNIFLSPLSISTAFAMTKLGACNDTLQQLMEVFKFDTISEKTSDQIHFFFAKLNC
                RLYRKANKSSKLVSANRLFGDKSLTFNETYQDISELVYGAKLQPLDFKENAEQSRAAI
                NKWVSNKTEGRITDVIPSEAINELTVLVLVNTIYFKGLWKSKFSPENTRKELFYKADG
                ESCSASMMYQEGKFRYRRVAEGTQVLELPFKGDDITMVLILPKPEKSLAKVEKELTPE
                VLQEWLDELEEMMLVVHMPRFRIEDGFSLKEQLQDMGLVDLFSPEKSKLPGIVAEGRD
                DLYVSDAFHKAFLEVNEEGSEAAASTAVVIAGRSLNPNRVTFKANRPFLVFIREVPLN
                TIIFMGRVANPCVK"
intron          646..2943
                /number=1
repeat_region   complement(2193..2469)
                /note="Alu repeat element"
exon            2944..3310
                /gene="AT3"
                /number=2
intron          3311..5842
                /number=2
repeat_region   complement(3784..3965)
                /partial
                /note="Alu repeat element"
repeat_region   complement(4007..4284)
                /note="Alu repeat element"
exon            5843..6058
                /gene="AT3"
                /note="A"
                /number=3
intron          6059..6963
                /note="A"
                /number=3
exon            6964..7101
                /gene="AT3"
                /note="B"
                /number=3
intron          7102..7911
                /note="B"
                /number=3
repeat_region   complement(7277..7558)
                /note="Alu repeat element"
exon            7912..8302
                /gene="AT3"
                /note="4"
variation       replace(8131,"a")
```

```
                     /gene="AT3"
                     /note="polymorphism"
     variation       replace(8160,"a")
                     /gene="AT3"
                     /note="polymorphism; Pst1 site"
     intron          8303..10335
                     /number=4
     variation       replace(8521,"t")
                     /note="Nhe1 site; polymorphism"
     repeat_region   8670..8687
                     /rpt_family="att"
     repeat_region   complement(8692..8951)
                     /note="Alu repeat element"
     repeat_region   complement(8994..9264)
                     /note="Alu repeat element"
     repeat_region   8994..9264
                     /note="Alu repeat element"
     repeat_region   complement(9362..9633)
                     /note="Alu repeat element"
     repeat_region   9823..9867
                     /rpt_family="att"
     repeat_region   complement(9871..10152)
                     /note="Alu repeat element"
     exon            10336..10400
                     /gene="AT3"
                     /number=5
     intron          10401..13774
                     /number=5
     variation       replace(10427,"c")
                     /note="Dde1 site; polymorphism"
     repeat_region   complement(10732..11011)
                     /note="Alu repeat element"
     repeat_region   complement(12124..12405)
                     /note="Alu repeat element"
     exon            13775..13951
                     /gene="AT3"
                     /number=6
BASE COUNT     3715 a   3338 c   3217 g   3936 t
ORIGIN
        1 ccacaggtgt aacattgtgt tttccttgtc tgtgccaggg acaccttggc atcagatgcc
       61 tgaaggtagc agcttgtccc tctttgcctt ctctaattag atatttctct ctctctctcc
      121 ctctctccat aaagaaaact atgagagagg gaattacagg tagagggcta gaagtttttg
      181 gacattaact atttctatct tctgatttag ttaacgagaa acaaaaaatc ctgcagacaa
      241 gtttctcctc agtcaggtat ttcctaacca agtttgaggg tatgaacata ctctcctttt
      301 ccttttctat aaagctgagg agaagagtga gggagtgtgg gcaagagagg tggctcaggc
      361 tttccctggg cctgattgaa ctttaaaact tctctactaa ttaaacaaca ctgggctcta
      421 cactttgctt aaccctggga actggtcatc agcctttgac ctcagttccc cctcctgacc
      481 agctctctgc cccaccctgt cctctggaac ctctgcgaga tttagaggaa agaaccagtt
      541 ttcaggcgga ttgcctcaga tcacactatc tccacttgcc cagccctgtg gaagattagc
      601 ggccatgtat tccaatgtga taggaactgt aacctctgga aaaaggtaag aggggtgagc
      661 tttccccttg cctgccccta ctgggttttg tgacctccaa aggactcaca ggaatgacct
      721 ccaacacctt tgagaagacc aggccctctc cctggtagtt acagtcaaag acctgtttgg
      781 aagacgtcat ttcaagtgct ctccctccca ccccacctct tggggtaagg cctttcctaa
      841 gctacccctt gggtccctag cctaagaaac aaggggatg tcatccctgg tgtaaagatg
      901 ctgtgcagga agtcagcact cacgggatcc aggggacgct ccaaggggaa tccccagggc
      961 ctgccatcca tccgggaaga gagcaaatgc tacccatgag gacctcctca ctccctttt
     1021 gctctttctt ccactcagat ccaccccact ccacccccac ccaaatccca gtgacctttg
     1081 actaaagggc caaaactgct tccttttctc acaatgagag ttgtccctcc ctcaatgcca
     1141 cacacactcc cttcttcatc tgagttgtca caggaggcta gaaacggggt ggtggcacaa
     1201 ctgtcttggt tttaatttgt gcttcatagc cctcccaggt cctctcagcc tcaaattgca
     1261 tttccaaatg tagttgaagg gacagagtgg gcaaccgaag cagcagtgga gatgggaaga
```

1321 tgaatggcag ggtcctctcc tctctctctc tgcttcttca gcctgccttc cacatctccc
1381 ttggtgccgc tgcttctctc cggctttgca cctctgttct tgaaagggct gcagaactgg
1441 actcagacca cgcaagaagg caagtccccc tcagctgccc cagcttccag ccagccccag
1501 gcttgcccaa cggaccacgt ccgtgaatct gcactgggtg cctgtctttc tctcccagga
1561 gaagatggga agatccagta cccacacaca gacccccttg tgtacacgca ggaaccataa
1621 accagctgga ggcagcccct gccccaccct gtcttatcta caaaaaatat tacaagagac
1681 tttatctctt gatttgcttc atcgagtgtc ccaactacct cattttttta aaatgtgaaa
1741 ttagcttcat ttaccttcat tgaatccatg ttggcgacta ttaaaaattc caggcaataa
1801 aaagggatga gagcctgaac taaagcagtg gcaataactg gtgaaagagt aaaaaaacag
1861 aactgattga ctctggggtg aactgattga ctctggggtt tgactaaatg aggaggagag
1921 agggaggaat ccagggtgat tctcaggttt ctgtacggga ttcactgagc ccactcacag
1981 gagcaggcct gtgggggaga attaattacc agttcagttt ggtcctgttt ccctgaagaa
2041 cttgtaggag ttcctggtgg aactgtccag caaatagtca gtctggagct cagtggaagg
2101 gttagggctg gagctagaga tgtaggaatc ttcagcacac agatattgcc attgtttttg
2161 tttgtttgtt tgtttgttgt tgctgttttg aagacacagt ctcactttgt cacccaggtt
2221 ggaggtcagt ggcacaatct cagctcactg caaccttcgc ctcctgggtt caagtgattc
2281 ttctgcctca gcctccctag tagcttggga ctacagtgtg cgccaccaca cccagctaat
2341 ttttgtattt ttagtagaga cagggtttca ccatgttgtc caggctgatc tcgaacaccc
2401 aacctcaagt gatctgcatg cctcagcctc caaagtgctg ggattacagc gtgagccacg
2461 cacccggcca gatattgcct ttgctccatc catttcttct ttttctcttg tgttgctgaa
2521 atctctctgc ctgcatctat cagagtcctt ccccaaacag tttctgtaga tggctccccc
2581 taccaccctg actcttcact gggcactaaa gccgattttt taggcatgca cattccatgt
2641 cacaaacagg aagcttctca ttctttttc tcccagcgtg gggaattgag cacataatac
2701 tccaaataac catcagatga ttctaattcc aacatgacca cgtccaggca actgaactgt
2761 cccctggcaa gaagtctagg actgaacctg tcccgggccc ctgtacttgg ttcaaaggat
2821 ttagcctttc tcttggccac accaggtggg ctggaatcct ctgctttact ggggcaaccc
2881 tgtggtgggc agtggggcta ggggttgcag cctagcttaa cttggcattt tgtctccttg
2941 caggaaggtt tatcttttgt ccttgctgct cattggcttc tgggactgcg tgacctgtca
3001 cgggagccct gtggacatct gcacagccaa gccgcgggac attcccatga atcccatgtg
3061 catttaccgc tccccggaga agaaggcaac tgaggatgag ggctcagaac agaagatccc
3121 ggaggccacc aaccggcgtg tctgggaact gtccaaggcc aattcccgct ttgctaccac
3181 tttctatcag cacctggcag attccaagaa tgacaatgat aacattttcc tgtcacccct
3241 gagtatctcc acggcttttg ctatgaccaa gctgggtgcc tgtaatgaca ccctccagca
3301 actgatggag gtacgaccaa aggtcttctg cccagccacc ttgttaggag caccttggg
3361 cttccatagg cccaagtcca atgattcctc aaccaacact gcagccacta ggggcgctca
3421 ttatgcatta cgattccctt tgaacatcac tgtgttataa ttccctttga aaatcatttt
3481 ttaaaaaatt agccaaggaa tcttggctat ctacttttta aatcctggtt tcctcttttg
3541 agcaccttaa aatgggggaa ggcttgtatc ttctctcaac ttcttttcag taattctttc
3601 atctatatgt ttactcatta atttgatcat ttatttattt attcattcag cacttcctct
3661 gtgccaggca atgtgtagtg ccagtccctc ctctggtgga agaagagtag ctttaccata
3721 tggtgacatc aggcatatag gctctcgtgg aaaaaaattc taggatagta ttttttttt
3781 tttgagatgg aatctcgctc tattgcccag gctggagtgc agtggtgcag tctcggctca
3841 ctccaaactc tgcctcccag gttcaagcaa ttctcccacc tcagcctcct gagtagatgg
3901 gattacaggc acacgccatc acgcccagct aatttctata tttttagtag agatggggtt
3961 tcaccacgtg gccagactgg tctcaaactt tttttttttt ttttttgaga cggagtctcg
4021 ctctgtcatc caggctggag cgcagtgcac gatctcagct cactgcaacc tctacctccc
4081 gggttcaagc aattctcagc ctcagcctcc cgagtagctg ggattacagg cccccggcac
4141 catgcctggc taatttttt tcttcttagt agagatgggg tttcaccatg ttggccaggc
4201 tagtcttgaa ctcctgacct cgtgatccac ctgccttggc ctcccaaagt gctgtgatta
4261 caggcgtaac gaccgcgcct ggcctcaaac tcttgacttc aagggatcgg cctgacttgg
4321 cttcccaaag tgctgggatt acaagcatga gccactgcac ggggcctagg atggtatatt
4381 gagaccaggg gcccaggaaa gccaagagaa gcctcaagga cgtgagagtg tttctggctc
4441 tgggaagtat ggatcatttc agctcagtga cttagttccc acccccttcc ccccactgcc
4501 ttttgtggga gggaagtagg gcatgataag atgaaatgtc atagattgat tgatcactgt
4561 tggcctctgg ggctatgaca agtcatggat ggaaacacta gatctttaat ctgtccttgg
4621 cttggctgca tgacagtctt tcttcaagtt ggatcacact ttggaagcag agttcatcaa
4681 tagggaggca tgagtccctt caagatggta tacggtgctt atttgaaact tggacactaa
4741 agtctgtggg tcttaggagg gttccttcta ttctagtggt caatttccat ggaacttcat
4801 caccttttgct cagggctctg gggtgagtta acccaagtct tcactctttg aaagaaattg
4861 tagatttaaa aactctgaag acacataata ctgccttctc tgggcccttc agtcattttt
4921 gtatacattg gtactggtct caaagtactt ccatatcact catgtctctg tccccaggta

4981 agatcttaaa tgtaaccctt cctacaagaa gaacagaaca gaacactgct tccaaaccac
5041 acatgttcct ttggtcctcc cctctacaca aacgccatgt gttgggaaag cagggtgaga
5101 ctaaatctct ctggagaaaa gagaaattca gcaccaagct tttgatcaaa agcataatcc
5161 ccccctaaaa aaagtgccta ttggagcaaa atcaggaaaa ccaaaggcag agaacagata
5221 aaaccaaaag gccttttgta gcctgaggag agagcatgga aagggcagga ggggaacagc
5281 ctcacccatt ttgccttggg gatggtgaag gtgggcattg ggggattcca acttcaaagc
5341 atggatgact tctaagtcct tttcagccct gagctcttag attctgagcc tgtttaatcc
5401 cttgctgata gattcactct tccttttttca cccctaccac cagtatccca gagcctccat
5461 gagcagctgg ccccagtaga tgccacaaaa gtgtttgtta cgagaaggac accgtctgat
5521 tctcttctct gtccagaatc accaagagga cttttcccat tccagcaaga aaacgtctgt
5581 gtgttgatct agaggcgttt agagacttta ggtggcaacc tagtctctct ttttcccttt
5641 atccttccta cccttcattc ttcttttatc cttttattca tcagaacaca agagttgagc
5701 atttatgctg tcccaggtac tgtgcttgaa ggagttaaca actgaggtgg ctattagtca
5761 gagactgacc agcatgtgct caccacccat gttaactagg cagcccacca aacccaccac
5821 catttttttt tgacttctat aggtatttaa gtttgacacc atatctgaga aaacatctga
5881 tcagatccac ttcttctttg ccaaactgaa ctgccgactc tatcgaaaag ccaacaaatc
5941 ctccaagtta gtatcagcca atcgcctttt tggagacaaa tcccttacct tcaatgagac
6001 ctaccaggac atcagtgagt tggtatatgg agccaagctc cagcccctgg acttcaaggt
6061 gagttgcaga tgttacccct gacctccgag ttcttcctct ccactcagag attgaggagg
6121 tggagaaaca gcatccaaat tcacactgct ttgctgctga agactgctgg agggctgact
6181 aaaagttaga acccctgcaa tagttattct tacttgaaac ctgagaaatc aaaggtatcc
6241 atgcttggat tgtagtgact gcccagaaaa catgaattaa taatcaattc ttcattccat
6301 ccaccaactt caaatatata ccaaagggtg ttttgaagat gccagttcta caagatatct
6361 tacttaattt gaactgttat catggtcaaa taaagttggt acatgatgca tgttacattc
6421 tcctcttgga gattcatgaa gcacatgggc ctatgaaggt tctgagaaac tctgcaacaa
6481 agaaatctgt tggctttatt caatcggcat tcctcaaatg tatttgactg catgggcatt
6541 tctctcctcc atataacctg ccaaccccat ataacctgcc aaccccatat aacctgcaat
6601 cattcattgc ttcccctggc acatgccttg gaaattctac ttttgtgagt taaggttttc
6661 caaagtcaga gaaaataata ttttatcttc tttttcccag actattttcg ccttccttct
6721 tttcatttat ttcttcctat ttctttttgt cttttcttc tgataatatt tattaactac
6781 aggaaagatt catggaacta tattagatat gtgaggcttc ccaatttggg ttagagcaat
6841 ggcttcttaa tcaaatggtg ggaaaggaca gagggatggt gagaaaaata aaatgctgcc
6901 tgggaaaatg gagaagccaa ttgaatagca caggtgagta ggtttatttt ctgttctcct
6961 caggaaaatg cagagcaatc cagagcggcc atcaacaaat gggtgtccaa taagaccgaa
7021 ggccgaatca ccgatgtcat tccctcggaa gccatcaatg agctcactgt tctggtgctg
7081 gttaacacca tttacttcaa ggtactcaga atggccctgg agagacccca gggacttcct
7141 cttgctcttc agcttacccc ctttttttt aaatggcgag accgaagccc tgagagggca
7201 aatggactgc cgaaagctac acaggtacag gtcagcaggg caggtcaatc tattatttat
7261 ttatttattt atttttgaca gagtctcgct ctgtcgccca ggctggagtg cagtggcgtg
7321 atctcggctc actgcaagct tcgcctcctg ggttctcggc attctcctgc ctcagcctcc
7381 caagtagctg ggaatacagg cacccaccac catgcctggc taatttttg gttttttttt
7441 agtaaagacg gggtttcacc gtgttagcca ggatagtctt gatctcctga cctcgtgatc
7501 tgcccacctc ggcctcccaa agtgctggga ttacaggcat gagccaccgc gcccggcaga
7561 ttggcttctt tcacctagta aaatgcattt actgttcctt tgtgttttcg tggctttgtc
7621 actcatttct tcttagcatg gaatagcatt acatttggtc tggatgtacc acagtctgtc
7681 tattcatcta ctgaaggaca ttttggctgc ttccaaggtt tgacagttat gaataaacct
7741 actcataatt ccatcattct gacacagcca ttgttaacct ttttgtgcat atcccgccag
7801 tctttttttcc gaataattat atattaatgt aacactataa tatggatatg tctgtgtcaa
7861 taactatcct cctatgaatg tttgtgttct tactttgtga ttctcttcca gggcctgtgg
7921 aagtcaaagt tcagccctga gaacacaagg aaggaactgt tctacaaggc tgatggagag
7981 tcgtgttcag catctatgat gtaccaggaa ggcaagttcc gttatcggcg cgtggctgaa
8041 ggcacccagg tgcttgagtt gcccttcaaa ggtgatgaca tcaccatggt cctcatcttg
8101 cccaagcctg agaagagcct ggccaaggtg gagaaggaac tcaccccaga ggtgctgcag
8161 gagtggctgg atgaattgga ggagatgatg ctggtggtcc acatgccccg cttccgcatt
8221 gaggacggct tcagtttgaa ggagcagctg caagacatgg gccttgtcga tctgttcagc
8281 cctgaaaagt ccaaactccc aggtttgtct aggaaggagt ttcctccctt ctccacccgc
8341 aaggtagtct gaccaaaagt ggaagagttg gagaaagaat agaaaggagc aacaagtcag
8401 gactcctgga tactgatcct agtttctact gctaatttgt ggaaatctct tttccttttg
8461 agacctcagt ttcctcttct gtaaaaggga agtttgttct tggatctcca tgggcccagc
8521 cagcactggt gccctgtgag tctgtatcag gtagaggaga tgggaccagg tggagaggaa
8581 tttgaaaggg cattggaatt cagagcaaag agacagatat taagagctgg ggaaatgtgg

 8641 ttcccattac acaggcctca ctgacattta ttattattat tattattact tgagacagag
 8701 tcttactctg ttgcccaggc tggagtgcag cggtgcgatc tcggctcact gcaacctctg
 8761 cctcccgggt tcaagcgatt ctcatgcctc agcctcctga gtagctggga ttacaggcac
 8821 acgtcaccat acctggtaat ttctgtattc ttagtagaga tgggtttcac catgttggcc
 8881 aggatggtct tgaactcttg accttgtgat ccgcctgcct tggcttccca aagtgctggg
 8941 attacaggcg tcacgaccgc acctggcaca ttaaaatatc ttttaaagaa gttggctggc
 9001 cagggtggct cacgcctgta ataccagcac tttgggaggc tgaggtggga ggatcgtcta
 9061 agcccatgag ttcgagacca gcctggacaa catagtgaga tggtctctac aaaaaataaa
 9121 aaaaattagc caggcatggt gacgcacacc tgtagtccta gcttcttggg aggcagagct
 9181 gggaggattg cctgagtccg ggaggtcaac gctgtggtgt actgtgatca caccactgca
 9241 ctccagcctg agcaacagag tgaggtccta tcactaaata aataaataaa taataaaata
 9301 gtttacgatg ttaagtaatt agatttatct ttattgacct ttttttttt ttttttttt
 9361 tgagacgaag tcttgctctt gtcccccagg ctggagtgca gtggtgcaat cttggctcac
 9421 tgcaacctcc acccccaga ttcaagtgat tctcctgcct cagcctccca aggagctagg
 9481 attacaggcg cctgccacca cgcccggcta atttttgcat ttttagtaga aacggggttt
 9541 cactatgttg gccaggctgg tcttgaactc ctgacctcag gcgatctacc tgccttggcc
 9601 tcccaaagtg ctgggattac aggcgtgagc cactgtgcta ttgggctgtc tttaagctag
 9661 ttttgaaaac taaaaatgtt gccagactgg aaagaaagat gttccttctg gatggagtga
 9721 gttttttctg taagaacaga gtcttgccgt tctctctcca caaaaagctg aagcctgaga
 9781 atgaattatc aggagccatg ctgaacaagc ccaaagtact ttattattat tattattatt
 9841 attattatta ttattattat tattattttt gagatgcagt tttgctcttg ttgcccaggc
 9901 tggagtgcag tggcgtgatc ttggctcact gcaacctcca cctcccgagt tcaagcgatc
 9961 tcctgcctca gtcttccaag tagctgggat tacacgatgc gccaccacac ctggctaatt
10021 tttgtatttt cacgatagag acaaggtttc accatattag ttagagtgtc tccaactcct
10081 gacctcaggt gatctgtaca ccttggcctc ccgaagtgct gggattacag gtgtgagcca
10141 ctgcacccag cccccaaagt actttattat ttttaacaca tattcattgt gagagtatga
10201 ttaggtgaag atttaggatt tcttcttatg tttcaaaaag ccccaaagga tctcttaatc
10261 caaactgaat tcccatctgt ggattgaagc caactttctc ccatctcaca aagacttctc
10321 cggtcttcct tccaggtatt gttgcagaag gccgagatga cctctatgtc tcagatgcat
10381 tccataaggc atttcttgag gtgagtacac cttccccact ctcttagggt acagaaagga
10441 gatgcatgaa cagcaggaac acgtggaaaa ggcctgtttc cagtgttaag gcatgcaaaa
10501 ggcctccaca ggctgctata atacagccct ctccaaaacc ttcatggtgt gattgttctg
10561 ccttccctcc cactacctct tctgtagcag gtcaagcggg aacacaaaca tttagggagg
10621 gtgatatagg aaaagaagcc agcaaaggcc atcaagaaga aatttacagc atgaggagaa
10681 ccagaagagt atggggtcgc agaaacccag ggagaatttt ttttttttt tgagacagag
10741 cttcgttcgc tcgttgccca ggctagagtg caatggtgcg acctcactac aacttctgcc
10801 tcccgcgttc aagcgattct cctgcctcag cctcctgggt agctgggatg acaggcatgt
10861 gccatcacgc ccggctaatt tttgtatttt tagtagaaac agggtttctc catgttggtc
10921 aggcaggtct tgaactcccg atctcaggtg atccacctcc ttggctcccc caaagtgctg
10981 ggattacagg catgagccac tgcacccggc catacctagg gagaagtttt aagaaaatgg
11041 atagcatcta gtaagaagac tcctgggctg ggcatggtgg ttcacacctg taaccccagc
11101 accttgggag gctgagatgg aagatcactt tgagcccagg agtttgcaac cagccctgag
11161 caacatagtg agaccctgtc tctaccaaaa aaatcttaaa aaaaaaaaaa aaagtttgga
11221 gactgcccat agtttacctt tccctgagga cagaatagtg tggccacatg cctaattgta
11281 atggatgaag agcaaatgga aggtaagaaa gggaagctgg tgagtgtgca tcagtgtctt
11341 aaagtgtgct ccaactagag cactagacta cactggagga aacgaaaagg tggtcaaata
11401 aatgcatatc ctctcatggg agatgaacag tacacactga catgctgagg tctgacaagt
11461 cccacagtaa agaagacggt tgaatatcac ttaacgtgtt cccccaaatg agatgtgcat
11521 ggaaccctgt gttagagtaa tagcgtgtac agcctgtgga acttctggtt ctcaagtaaa
11581 cactaaacta tggaccaaca gcagtagtta tctggggagc tttatctttg gagattctgg
11641 atgggccctg agaatctctg tatgtactaa actcctcagg ggattcttat gcaaacaatg
11701 agatttggga accactggta tagactattt tttgcgggag ccaggctgtg agggatagga
11761 gattggacaa tggtagagat gttcctagaa tcaaagaaat cgaaaagaat gaaggttgta
11821 gtcaaagaga aaggtttcag aggatggtgc taagtaaata tagatccagg gatcaaactc
11881 agaggaaagt ggaattttaa cgggagcgag gaaatgtgat agcttgaaag aacccagtta
11941 taatctgaga aagatgctat taatataatt tcaaaggtag agtagttctg agatggcaag
12001 tccaaggtat agccatggac aggtttgctt aagtggaata aggcaatgct cattaggttt
12061 gatgaaagaa atgggagact agggtgttga acgggtctcc taatgagatt ttttttttc
12121 attgaaatgg agtcttgctc tctcgcctag gctggaatgc agtggtatga tctcagttca
12181 ctgcaacctc cgcctcctgg gttcaagcga tcctctcgcc tcagcctccc aagtagctgg
12241 gactacaggt gcccgccacc acgcccgact aaatttttgt atttttagta gagacggggt

```
12301 ttcaccatgt tggccaggct ggtctcaaac tcctgacctt aagtgatcca cctgactcgg
12361 cctcccaaag tgctgggatt acaggcatga gccacgtgcc cggcctactg agatattttt
12421 aattgcctca aatgatagca ggagttggag tggacagaaa ggctaagtgc aaaaatcatc
12481 agtgtgggga tataatctat aggacaatga atgtcaatga cctttaagac aatagcaaga
12541 gtagaggtat tgaggtcaga acaagggatt ttacaagagt gctgtattaa tggttttgga
12601 agttaagatg acactgctca caccctcttt cacatggatt tttggaagaa agaacactta
12661 ggaagactgc aagggaaatt gagtcctcag ggttttaact ctcattgaat atcctctggt
12721 aaggactcca gttagaagtg gtcaactcag acctccttga ggggtctgag ttactattag
12781 gaagaagcag aggtctggat tcattttatc cacctgagcc cagtacacaa tatgtaagat
12841 ttttccatgg ttcttacaac aaagccgttt tctttgaaaa ccttgggaat ccttaataaa
12901 caggacccca actaattgta gacctgagaa gccattaaaa accagaatct gatttgaata
12961 aaaggatcct ggtcatgcaa acactctagt ctgctaataa gctaataatt tagtgctgga
13021 atgagcatga aataggtaat atggggagat agcgggtaag gaagggagga acaaaggaag
13081 gggaaggaag agtgagaagg aaggagaaga catcatcaac cagctccaca aaacccaggg
13141 agccggttaa tcatgtgctt tcattaagag cagaaacaga gttttagtga tattctgggt
13201 cctgaggcaa aattttctga aggtgtttcc ctctagatcg ctatcagcca tgttcaaata
13261 ccattgtttc agtctattac tccaagaaaa tggcatctcg tccagccaga gaacccacct
13321 cttttcatag gccctaggtc ctgagtggct ctttggagta gctgtatctt ggatcttgat
13381 gctccaagag tgaaactgtt tctttcaact atggagttca gatcttgagc caaaaatctt
13441 tcagcggctg gtacaaaaaa aatccgctgt aaaaccattt acaatggtac cagccagaaa
13501 tgtgataacc tgtgtacatt cagatttctg ggttacctga atggaactct tacacttatt
13561 acctagcaca aggcttggac aaacacaagt accttacatt ctctgcatga aagaatgagt
13621 gaaagtagga ttctggaggg aatccaacct gacccaaatg tacttttac tggaaaacaa
13681 aagcatttga ggaattgctg tgtctgtgga tgatttacct gccaaaatga acggcagagt
13741 ggctaattta gttttattcc catgtgacct gcaggtaaat gaagaaggca gtgaagcagc
13801 tgcaagtacc gctgttgtga ttgctggccg ttcgctaaac cccaacaggg tgactttcaa
13861 ggccaacagg cctttcctgg tttttataag agaagttcct ctgaacacta ttatcttcat
13921 gggcagagta gccaaccctt gtgttaagta aaatgttctt attctttgca cctcttccta
13981 ttttggttt gtgaacagaa gtaaaaataa atacaaacta cttccatctc acattataaa
14041 tggactctgc atttgaaatg aagacaagga aaggggaaac atgctattgg ggcacatggt
14101 aaaatgatgc cttcaagttg ttctttaccc agtaaccaca tctggatcaa gaaaatgagg
14161 gagagagcga taaaagatgg tagacagcca gaaagggaag ggagag
```

Top.