Project 1: Nucleotide Sequence Assembly from DNA Microarray data

Due Date: Mar 2, 2012, 12:01am (i.e. Friday night). Grace period 8 hours, no late projects will be accepted.

Project Goals:

The purpose of this project is to give you practical experience with the challenges related to determining DNA sequences. As in any rapidly evolving field of research, sequencing technologies are constantly changing, and sequence assembly techniques are changing with them. For this project, we will work with the problem of assembling DNA sequences generated with DNA Microarrays.

Data from DNA Microarrays can be used for Sequencing by Hybridization (SBH), which will be described in detail in class. SBH assembly is an ideal platform for a class project because it confronts the major problem that every sequencing technology faces – ambiguity – without requiring a very deep background in mathematics that is necessary for other assembly algorithms or other sequencing technologies.

Two projects are defined in this document: the "implementation project" and the "applications project". Pick exactly one.

Background:

Hybridization is the process by which an unpaired strand of DNA attracts a complementary strand of DNA. DNA Microarrays exploit hybridization by attaching thousands of unpaired, carefully selected DNA strands to a tiny chip. By recording which DNA strands are hybridized, DNA Arrays can be used for a tremendous range of genetic diagnostics, and also, to a lesser extent, genome sequencing.

DNA microarrays would be superior DNA sequencing apparati if their application was not complicated by the problem of cross-hybridization. We'll learn about cross-hybridization in class. This complication is the topic of open research, so it is outside the scope of this class. We will therefore make the simplifying assumption that cross-hybridization does not occur.

Definitions:

Reads: All modern DNA sequencing technologies produce their data by breaking strands of DNA into short segments of nucleotides called reads. Read lengths vary based on the underlying technology. Illumina sequencers produce very short reads (25-75 bases), while pyrosequencing

produces much longer reads (400+ base pairs). DNA arrays, such as those produced by Affymetrix, can have reads between 10 and 100 base pairs.

Contig: The sequence of nucleotides in DNA is highly ordered and nonrandom, containing many regions that are repetitive. When attempting to sequence DNA with technologies that employ short reads, it becomes impossible to unambiguously join reads in repetitive regions, making the overall sequence unknown. Blocks of reads that can be unambiguously assembled, possibly because they contain little repetition, are called contigs.

ClustalW: ClustalW is a venerable sequence alignment algorithm useful for both protein and DNA sequences. You will be provided with this software to test your alignment results.

BLAST (megaBLAST): BLAST (Basic Local Alignment Search Tool) is a local sequence comparison algorithm designed for searching databases of protein and nucleotide sequences. You will use BLAST, provided as a web service, to test your data.

SSAKE: SSAKE (Short Sequence Assembly by K-mer search and 3' read Extension) is an algorithm for sequence assembly that uses prefix trees to join reads into contigs. SSAKE was developed by Rene Warren at the British Columbia Cancer Agency, a Canadian research agency.

Implementation Project: Implement an assembler for SBH

You will be provided reads in a fasta format (like the example fasta file that we aligned in the Introduction to Unix lectures). You will write software that reassembles the reads into the final sequence. The reads might not all be part of the same contig, in which you should return all contigs. The output contigs should also be returned as a fasta file.

A detailed mathematical explanation of this problem and how it should be solved is available in the course textbook "An introduction to Bioinformatics Algorithms" by Neil Jones and Pavel Pevzner, starting from page 262 to page 280.

/proj/cse308.s13/project1/testData This directory includes nucleotide sequences of the Herpes virus and a gene from yeast (genBank entries). It also includes a set of reads for each and an "answer key" for each, indicating the correct order of the reads. You can use this as test data for your code, to check the accuracy of your assemblies against the full assembly (by hand and via clustalw).

/proj/cse308.s13/project1/SSAKE SSAKE is a sequence assembler similar to the one you will implement. You can use this assembler to check results on smaller test data that you make yourself.

/proj/cse308.s13/project1/ClustalW ClustalW will be a useful tool for comparing your assemblies with SSAKE assemblies as well as the template sequences you are provided.

Report: 3-5 pages, single spaced:

Section 1: Describe exactly the algorithm that you implemented, and its strengths and weaknesses relative to simple extension algorithms. (50%)

Section 2: Some regions are easier to sequence than others. Describe the regions that are difficult, and their biological purpose. If you could design a more sophisticated sequencing technology how could it sequence these difficult regions? (10%)

Section 3: DNA arrays are more successful for applications other than sequencing, including Gene expression profiling, SNP detection, alternative splice detection, and Chromatin Immunoprecipitation. Describe what two of these applications are, and why DNA arrays are effective for them. (40%)

What you will submit:

Source code and compilation script (e.g. makefile) for your software, to be compiled on Sunlab. Include instructions necessary for compiling your code.

Report, submitted as a pdf file.

Applications Project: determine the virus in the DNA sample

You are locked on a space ship with an unknown virus and a DNA microarray scanner. Worse, Dr. J. Craig Venter, the famous biologist, is on the ship with you, and infected. Beginning with a sample from Venter, who was too weak to do this work himself, you were unable to separate his DNA from that of the virus, but at least you have a number of sequence reads from your trusty Affymetrix 3000 7G. So trusty, in fact, that cross hybridization never occurs on this device.

Assemble these sequence reads using SSAKE. SSAKE does not know which reads are part of which genome, and, in fact, most contigs are not from the virus. Search for parts of the contigs in GenBank using megaBLAST. megaBLAST should reveal if the contig comes from a human, or something else… Fortunately, based on sample, your searches in genBank should be especially accurate*

But first, you must figure out how SSAKE works. Using the supplied python script, genBank2Fasta.py, break a genbank file down into artificial sequence reads of varying lengths with varying overlaps. Run a complete range of experiments indicating how the number contigs found by SSAKE varies with the minimum read overlap and the read length. Test all read lengths of 40, 60, 80, and 100 against all minimum read overlaps of 5, 10, 15, 20. This should result in a 4x4 grid of tests (maximum read overlaps should be 5 larger than the minimum read overlap in each case). You can generate the reads for this activity using the "-reads" command. Use "10" for coverage.

/proj/cse308.s13/project1/workData This directory contains the reads for your specific project. These are the reads containing human and viral DNA mixed together. Note that data is different for different groups, so if you have selected the applications project, see the group rosters below.

/proj/cse308.s13/project1/testData This directory includes nucleotide sequences of the Herpes virus and a gene from yeast (genBank entries). Use these sequences to generate your reads of varying lengths and overlaps.

/proj/cse308.s13/project1/SSAKE SSAKE is a DNA sequence assembler. You will need it to assemble the reads that you have in the workData directory.

/proj/cse308.s13/project1/script genBank2Fasta.py is a python script that you will need to run in order to generate simulated reads from a genBank file. Using this script, you can generate new reads with varying lengths and overlaps.

megaBLAST: http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastHome

megaBLAST is a web-based tool for searching huge nucleotide databases (e.g. genBank) for sequences that resemble a given sequence. You can feed it parts of contigs (say 400-500 bps) and it will try to find the sequence in the database closest to your sequence. This should help you identify your virus.

Under the section "basic blast," select "Nucleotide Blast". In the next page, paste your contig into the box labeled "Enter accession number(s), gi(s), or FASTA sequence(s)." Under "Choose Search Set", to the right of the word Database, select from the drop down menu "Nucleotide Collection (nr/nt)". Finally, under "Program Select" make sure that "highly similar sequences (megablast)" is selected. Then press the BLAST button on the bottom.

Report: 3-5 pages, single spaced.

Section 1: Explain eulerian path assembly, and why it seems to separates the human and viral sequences. Mathematically speaking, why or why not is this process likely to remain error free for any groups of mixed genome reads? (33%)

Section 2: Describe the contents of the contigs identified with SSAKE. What organisms did they come from? Can you identify the virus? If so, identify the viral genome within genBank, download the sequence, and align the contigs to the sequence using ClustalW. (33%)

Section 3: Faced with reads of different lengths or overlaps, SSAKE can perform differently. Present a heatmap indicating the number of contigs you found for each length/overlap combination. Detail how eulerian path assembly algorithms, like those implemented for the implementation project, are challenged by extremes in read lengths and/or overlap. (33%)

Section 4: Why would your searches in genBank, for your particular sample, be especially accurate? (1%)

What you will submit:

ClustalW alignments of the contigs from your data that aligned to the viral genome you identified. An excel spreadsheet indicating the number of contigs for variations in read length and overlap.

Report. Submit as a pdf file.

Submitting your project:

All submissions files should be bundled into a tgz file and dropped off via Lehigh's drop box service for byc210@lehigh.edu.

Evaluation:

Implementation project: The executable will be evaluated by assembling new reads and testing those sequences against a known final sequence. There will be 100 randomly generated final DNA sequences, and the executable grade (50% of the project) will be the proportion of reads that are correctly assembled. The report will be evaluated as per the weights mentioned above, and will be 50% of the project.

Applications project: Your project results, 50% of the project grade, will be evaluated based on whether or not you were able to assemble the reads and identify the organism source of the contigs, as well as your plotting the relationship between read length and minimum overlap and the number of contigs identified. The report will be evaluated as per the weights mentioned above, and will be 50% of the project.

Some more useful Unix commands to know about:

wc:

wc stands for "word count". It takes stdin or an input file or a wildcard modified list of input files, and returns the number of lines, the number of words, and the number of bytes in the file. The number of lines is especially important because you can use wc to count the number of outputs you generate with the grep command.

tar, gzip, gunzip:

Tar and gzip are a simple way of combining multiple files into a single compressed file, a tgz. It works basically the same as a zip file. Tar rolls everything into a single file (a .tar file), and gzip compresses the tar file (into a .tgz). Gunzip uncompresses (the .tgz), and tar can return the tar file into the original multiple files.

Project 1 Groups Rosters

	Roster	Applications Read Set
1	Chu, Bicher	set1reads.fasta
2	Romero, Dodd	set2reads.fasta
3	Shardt, Feinstein	set3reads.fasta
4	Wallace, Gu	set4reads.fasta
5	Wu, Guo	set5reads.fasta

Grading rules:

Everyone in the same group will receive the same project grade: the numerical average of the individual project grades. Everyone writes their own report. Problematic groups will be split by the instructor, leading to individual grading.

On collaboration:

Everyone on the roster is stronger on some aspects of the course and weaker on others. In fact, project groups are selected deliberately to combine individuals with complementary skills. Do not be surprised if you feel you are working in an area that you know little about – everyone feels that way in an interdisciplinary situation. The way you make progress is by reaching out to those around you for help on the things you do not understand. Not only will it help you learn, but if you document it, you will get extra credit for it. This is part of the intended learning experience in this course, and I expect you to use collaboration as a primary means for understanding the course.

You can talk to anyone, inside or outside your group, as much as you want, about anything. I encourage that. However, there is a distinct line between talking about a project, which is encouraged, and actually doing someone else's project for them, which is discouraged. **That line is the point at which one person is no longer learning.** I am trusting your judgement on properly finding this line. Teach those around you as you learn from them: it's an important career skill for now and the future.

Having met almost everyone in the class, I think the teamwork should be very positive. To foster your collaboration skills and to further eliminate any possible friction, groups will be different for each project. Fatally irreconcilable groups will be separated into individual projects with individual grades

Extra Credit:

As mentioned in class, extra credit (up to +25%) is achieved by using a student-implemented assembler to fully separate the mixed up human and viral reads, annotating all the contigs generated (including which chromosome the human contigs came from), and reassembling the viral and human contigs as best as possible. Up to +25% extra credit can also be achieved by having numerous and positive interactions with other students in the course of the project (approx. 2% per interaction, on the instructor's discretion). Partial extra credit in both cases will be evaluated on a case by case basis. A maximum of +50% is possible.