CMSC 423 Fall 2009: Project Specification

Introduction

The project will consist of four components due throughout the semester (see below for timeline). Basic rules:

- You are allowed to work in teams of at most 2 people.
- The teams can change throughout the semester (i.e. you can work on part 1 with one of your colleagues and on part 2 with another one if you wish). Please clearly indicate on your submission who the members of the team are (both will get the same grade, irrespective of contribution).
- You can use any programming language you wish.
- Your software must compile and run on the Glue machines so make sure that you test it before submitting it. **You can get at most 50% of the grade if we have difficulties compiling or running your code.**
- The projects must be submitted using the "submit" command on the Glue system. Note: this is different from the "submit server". You must be logged onto glue.umd.edu or linux.grace.umd.edu in order to run the "submit" command.
- Your code must be accompanied by a README file that explains the steps necessary to compile and run your project.
- 10% of the grade for each component of the project will be awarded for "best programming practices" - make sure your code is neat, well organized and thoroughly commented.

Deliverables/Timeline

- FASTA parser.
  Due: 9/17/09 Weight: 10 %
- Global alignment of two DNA sequences.
  Due: 10/20/09 Weight: 10%
- Local alignment with affine gap penalties.
  Due: 11/03/09 Weight: 30%
- Overlapper for assembly & incorporation into Minimus assembler.
  Due: 12/3/09 Weight: 50%

Part 1 - FASTA parser
Due Thursday, September 17, 2009
Overall weight: 10% of total project grade

The first part of the project requires you to write code that can parse a FASTA file. For this part, you are not allowed to use any of the Bio* libraries available for your programming language of choice.

Specification: Your program should read in a FASTA file (sample is available on the Glue system in /class/fall2009/cmsc/423/0101/public/test.fasta) and output a list of sequence identifiers for all sequences that satisfy one of the following:

1. are less than 100 bp in length
2. contain at least one character that is not A,C,T, or G

Details:

- For clarifications on the FASTA format see the Wikipedia entry: http://en.wikipedia.org/wiki/FASTA_format
- In addition, you can assume that a sequence identifier follows right after the ">" sign. If a sequence doesn't follow this rule you can exit with an error.
- You can also assume that the identifier ends with the first "space" character (space, tab, or end-of-line)
- **Interface:** Your program must accept the input fasta file either through the standard input, or as the only command-line parameter. The output should be provided on standard output.
- Any questions about this assignment should be sent to both myself and the TA.

Part 2. Global sequence alignment

Due: October 20, 2009

Overall weight: 10% of total project score

Specification: You must implement the dynamic programming algorithm described in class to construct the global (end-to-end) alignment of two DNA sequences. Your program must accept two DNA sequences in FASTA format and output a global alignment of these sequences in the following format:

```
Edit distance = 7

Seq1  ATTC-TCAT--TAGGACCGGC
       ||  ||||  ||| ||| ||
Seq2  -TTGATCATGGTAG-ACC-GC
```

Note: the vertical bars indicate characters that match between the two sequences.

If the sequences are too long to be displayed on one line (assume a line has 80 characters), the alignment should wrap around as shown below:

```
Edit distance = 12

Seq1  ATTC-TCAT--TAGGACCGGC
       ||  ||||  ||| ||| ||

Seq2  -TTGATCATGGTAG-ACC-GC

Seq1  GCACATCA-G-TAGGACC
      | | |||| | ||| |||

Seq2  GTAGATCATGGTAG-ACC
```

Interface: Your program should accept 5 parameters on the command line: the names of the files containing the two sequences, and the scores for a match, mismatch, or gap in the alignment. Below are two examples:

Simple: `myProg file1.fa file2.fa 3 -1 -2` (parameters are simply listed in order)

With options:
`myProg -s1 file1.fa -s2 file2.fa -match 3 -mismatch -1 -gap -2` (use command-line options)

You can pick any option you wish, and even allow certain parameters to be missing (in which case they would be assigned default values), however you must indicate in a README file how to run your program (and what the default parameters are if they are not specified).

Additional details: Any questions about this assignment should be sent to both myself and the TA.
You can assume that the two FASTA files contain exactly one sequence (or if they contain more than one just use the first sequence in each file).

You can now use any of the Bio* libraries to read the FASTA files (but not to perform the alignments).

Part 3. Local sequence alignment with affine gap scores

Due: November 3, 2009

Overall weight: 30% of total project score

Specification: Extend the program you wrote in Part 2 to achieve the following:

- Perform local alignment (find best matching substring)
- Accept alignment scores formatted as a BLOSUM matrix (detail below)
- Gap penalties are "affine" - the combination of a gap opening penalty (paid once per group of gaps) plus a gap extension penalty (proportional to number of gaps within a group).

Output format: The output of your program must follow the format shown below - note the coordinates within the two strings of the aligned region, and that the '|' characters are replaced by the actual letter that matches between the two strings. Just as before, if the strings are too long to be printed on one line (assume one line is 80 characters) you will need to wrap the alignment around on multiple lines.

```
Score = 188, Identities = 49/152 (32%), Gaps = 6/152 (3%)

seq1   23   LPKTRTKALLTALTLAAAAAAAPALADVEFRHAL---DDSALDLSPIKGEEITDAVKSFR  79
             P         A       A   AL    FRH     D      S   G   T AV  F
seq2   3    MPSFNRSIAISATLAVGLLAPVVALGQEVFRHTVTGEDLKIMETSQPSGRD-TEAVRNFL  61
```

A sample output file is provided at http://www.cbcb.umd.edu/confcour/CMSC423-materials/Output.txt

Interface: follow the same rules as for Part 2, except that you must now accept two new values - gap opening and gap extension penalties - and a new file - the BLOSUM matrix specifying the substitution/match scores.

BLOSUM matrices: A sample matrix is provided at http://www.cbcb.umd.edu/confcour/CMSC423-materials/BLOSUM80.txt

Note: most of the Bio* libraries contain utilities for reading in BLOSUM matrices.

Additional details: Any questions about this assignment should be sent to both myself and the TA.
You can assume that the two FASTA files contain exactly one sequence (or if they contain more than one just use the first sequence in each file).

You can now use any of the Bio* libraries to read the FASTA files (but not to perform the alignments).

Part 4. Sequence overlapper for the Minimus assembler (AMOS package)

Due: December 3, 2009

Overall weight: 50% of total project score

Specification: You must write a program that reproduces the functionality of the hash-overlap program from the AMOS package. Specifically, your program must extract a collection of sequence reads from an AMOS bank, perform an all-v-all comparison between the sequences using a local alignment algorithm with affine gap penalties, then load the overlaps found into the AMOS bank.

In addition, you must integrate this program within the Minimus assembler, essentially replacing the call for hash-overlap with a call for your program within the script running the Minimus pipeline.

Your program must allow the user to specify the following parameters:

- minimum overlap length
- minimum % identity within the overlapping region
- alignment parameters (match, mismatch, and gap open, gap extension penalties)
- maximum region that can be ignored at the beginning/end of the aligned reads (i.e. how far from the end of a read is the alignment allowed to start)

All these parameters must have default values that will be used in case the user does not directly override the information. These values must be documented in a README file.

Documentation for the AMOS package can be found at http://amos.sourceforge.net.

You can use any Bio* utilities you find useful, including alignment routines.

You can (and should) use additional heuristics to ensure your program is efficient, e.g. use exact alignment tricks to figure out which sequences might overlap before performing the expensive inexact alignment operation.

Data sizes:

Expect that your program will have to assemble a data-set of ~ 100,000 sequences of length up to 1,000 bp.

Contest:

The top performing programs will receive extra credit: The student who writes the fastest program will receive a 20 point bonus. The next top 5 running times will receive a 10 point bonus. Note: correctness of the program will also be tested - your program must produce a reasonable assembly of the data-set in order to be considered for the speed competition.