Lisp in Summer Projects Submission

Submission Date	2013-10-22 14:37:27
Full Name	Piotr Mieszkowski
Country	Poland
Project Name	GAS
Type of software	command-line/terminal app
General category	utility
LISP dialect	Commmon Lisp
GitHub URL	https://github.com/pefimo/gas/tree/lisp_in_summer_projects_2013
Did you start this project?	Yes, all the code is written by me
Project Description	I want to describe my project in this form.
Purpose	The purpose of this program is to help the user connect fragments of DNA sequences provided by a DNA sequencing company (or other source of DNA fragments) as a set of FASTA or ABIF files, into a single sequence, and exporting this sequence to a text file.
Function	This program processes DNA sequences provided as input files, searching for the longest common sub-sequences and collecting them. Then, the user selects the best one and the program joins two sequences at the region selected by the user, producing a text file suitable for further processing or embedding in a scientific publication.
Motivation	The author of this project finds biology one of the most interesting sciences. Being a software developer, he wants to help his friends from the life sciences' field achieve better results with less effort by creating a simple and free tool, automating what they had to do manually.
Audience	The GAS utility might be useful to scientists working with files received from DNA sequencing companies. Limitations of the process of DNA sequencing make it hard

to get results as a single DNA sequence (unless it is very short). Usually a set of sub-sequences is generated and the scientist has to assemble them into one, long sequence by hand or using expensive software.

Methodology

GAS is built around several basic concepts: (1) DNA sequences, which are loaded from (2) files and can be operated on after being loaded into (3) workspaces. When two sequences are matched against one another, a (4) set of matches is created and can be used to join these sequences, producing a FASTA file.

The two most interesting parts of the program are: the common subsequence lookup code (GAS-SEQOP package) and the command-handling code (GAS-COMMANDLINE package).

Subsequence Lookup

The subsequence lookup is done by creating sets of n-grams from maximum to minimum length, collecting intersections of these sets. (There are two parameters available for the user to adjust maximum and minimum match-length. Using the SET command, the user can adjust these parameters.) Displaced arrays are used to reduce memory footprint of the program and to allow some sophisticated algorithms planned for the future of this project and not yet implemented.

There are two sequence comparison functions used in the subsequence matching code: weak and strong. The weak one compares sequences using string comparison function CL:STRING-EQUAL, while the strong one compares them character by character, being significantly slower, but IUPAC notation-aware (it is a nucleic acid notation consisting of 15 letters, 4 for bases that DNA is built of and 11 for ambiguous situations), thus it can yield better matches.

Command Handling

GAS command-handling code tries to extensively use Common Lisp's features like doc-strings and macros to create an elegant framework. The DEFCOMMAND macro can be used to define commands that have a help information, are compiled during macro-expansion and take arbitrary number or arguments defaulting to NIL if not supplied.

When a command evaluates to non-NIL value, GAS continues reading and evaluating commands, otherwise it is considered an exit command and finishes the GAS REPL.

Conclusion

GAS reads ABIF and FASTA files, processes DNA sequences read from such files and gives the user the ability to quickly locate the longest common subsequence and assemble sequences, producing a FASTA file. The most important goal of this project is thus accomplished.

There is very much room for improvement though. Some sophisticated subsequence lookup algorithms could be added to improve the overall efficiency, e.g. assigning weights to subsequences collected and promoting the ones found at the beginning and at the end of the input

	sequences, or demoting the ones containing long string of one base. The user interface could be made more accessible to the less computer-literate users. Future plans for the GAS project include implementing more efficient common subsequence lookup. (Current work concentrates on using different data structures.)
Build Instructions	1. Download and install SBCL. 2. Download the GAS source code from https://github.com/pefimo/gas/releases/tag/lisp_in_summer_projects_2013 3. Extract the source code tarball/zipfile and enter the "src" subdirectory. 4. Start SBCL REPL and evaluate the following expression: (asdf:oos 'asdf:compile-op 'gas)
Test Instructions	No test are provided with this project.
Execution Instructions	Download two random DNA sequences in the FASTA format, e.q.: 1. http://www.ncbi.nlm.nih.gov/nuccore/194321241?report=fasta (as AY225306.fas or AY225306.txt) 2. http://www.ncbi.nlm.nih.gov/nuccore/120543366?report=fasta (as AF484535.fas or AF484535.txt) and save them in two files in the "src" subdirectory of the GAS project. After compiling the project (see Build instructions), evaluate the following instruction: (gas-init:toplevel) Enter following commands: load AF484535.fas load AY225306.fas ls After this command, internal names of the sequences will be displayed. Use them to match the sequences: m gi|194321241|gb|AY225306.2| gi|120543366|gb|AF484535.2| And then join these two sequences using the command: join gi|194321241|gb|AY225306.2|:gi|120543366|gb|AF484535.2|
Describe any bugs or caveats	1. This software wasn't tested with Common Lisp implementations other than SBCL. 2. Then no common sub-sequences are found, JOIN command will result in error. 3. The command-handling module doesn't handle in-existent files correctly.
Screen shots	01_welcome-and-help.jpg 02_load-seq-and-basics.jpg

	03_match-and-join.jpg 04_exit-and-see-result.jpg
Official	I have read rules and have abided by them. I am 18 years of age or older. I am not living in Brazil, Quebec, Saudi Arabia, Cuba, Iran, Myanmar (Burma), North Korea, Sudan, or Syria.