INFO-H-304 2019-2020

COMPLÉMENTS DE PROGRAMMATION ET D'ALGORITHMIQUE

Projet

PROTEIN SEQUENCE ALIGNMENT USING THE SMITH-WATERMAN ALGORITHM

1 Introduction

Sequence alignment is one of the most commonly encountered problems in bioinformatics. The goal of sequence alignment is to compare two (or more) sequences, usually one unknown and one known found in a database, against some similarity measure. The sequences to be aligned can be either **nucleic acid sequences** for DNA and RNA or **peptide sequences** for proteins. The main difference between the two types is the number of different molecules. A nucleic acid sequence for DNA or RNA mainly consists of 4 types of nucleobases: cytosine (C), guanine (G), adenine (A) for both DNA and RNA, plus thymine (T) for DNA or uracil (U) for RNA. A protein sequence, however, consists of about 20 different kinds of amino acids (see [Wik15b] for a complete list). In this project, only peptide/protein sequences will be used. Over the last few decades, large databases containing the nucleic acid or peptide sequences of many known organisms have been built. These databases can be used by researchers or doctors to determine the origin and properties of some unknown organism, if it is properly sequenced. Because the sequencing technique often produces ambiguities, a database search is more than just finding a match for the query string. Various approximations and local optimizations have to be made. The Smith-Waterman algorithm uses dynamical programming to perform sequence alignment and is one of the most widely used algorithms for local sequence alignment.

2 Project goal and requirements

The general goal of the project is to produce a working program receiving as input a protein sequence to be aligned with a large database of sequences, using the Smith-Waterman algorithm. More precisely, the program should satisfy the following requirements (see below for details).

Language	• C++ and its standard template library • Third-party packages should not be used, except for the optional last step optimizations • It should still be possible to compile a version of the program without any third-party packages (hence without the optimizations) • Create a Makefile to facilitate the compilation for each possible version
Mandatory command-line parameters	• A file containing a protein sequence in the FASTA format (query sequence) • A file containing a protein database in the BLAST binary format
Optional command-line parameters	• A file containing a BLOSUM scoring matrix (default: `BLOSUM62`) • An integer indicating the gap open penalty (default: 11) • An integer indicating the gap extension penaly (default:1)
Output	• Basic information about the database, the query and the execution time • Intermediate deadline : the sequence from the database that is an exact match with the query sequence (the line should display the correct name of the sequence, obtained from the database, even if it was replaced by a fake name in the query file) • Final deadline : a list of sequences from the database with significant alignment with the query sequence (each line should display the name of the sequence and its score)
Project delivery	• The project should be delivered as a link to a git repository holding the source code • The git repository should be set up at the beginning of the project and shared with the members of the group, to allow collaboration, and to the professor and teaching assistant, to allow tracking your progress and the evaluation • For each deadline, the last version available on the day of the deadline on the git repository will be the one to be evaluated

3 How to do a real world local sequence alignment

3.1 The `swipe` program

In order to better understand how your program should work, we will first consider an existing program, called `swipe` and developed by a Norwegian researcher, which can be used to perform real-world local sequence alignment with freely available resources. The detailed description and performance analysis of `swipe` can be found in the paper [Rog11].

To use `swipe`, download the source (see below for all download links), uncompress it and run

```
1 make swipe
```

in the source directory to compile it. Once finished, you can copy the executable file `swipe` to another directory, together with all the files you will need to download in the following steps.

3.2 The protein database

A protein database contains all the known (or unconfirmed but predicted to exist) protein sequences. We will be using one of the most well-maintained, manually annotated database called Swiss-Prot. The whole database is contained in a single file, which can be downloaded from the UniProt website (see link below). Use GZip to decompress the file, and you will find the database file `uniprot_sprot.fasta`. This file is essentially a text file, which you can view with any text editor. Note that because of its size, opening it with a poorly designed text editor may make your computer unresponsive.

3.3 The query protein

Usually, biologists perform the sequencing of some unknown organism, and use the resulting protein sequence as the query. Since this is unrealistic for a computer science course, we will use known protein sequences (which are already in the database) as the query. UniProt has an easy to use interface which allows users to quickly find any known protein from just an identifier. To get the query protein used in this test, follow these steps:

1. Go to `http://www.uniprot.org/` and type `P00533` in the input box at the top of the page, just after the words **UniProtKB**.
2. You will be redirected to the search result page. You can see that this particular protein is called *the human epidermal growth factor receptor*. By clicking **Pathology&Biotech** on the navigation panel on the left, you can see that this protein is involved in lung cancer and neonatal inflammatory skin and bowel disease.
3. To download the sequence of this protein, go to the top of the page, click the button called **FORMAT** (just under the title **UniProtKB - P00533 (EGFR_HUMAN)**) and select **FASTA (canonical)**. Depending on your browser, you will either see the whole sequence as text directly or be prompted to save the file `P00533.fasta`.
4. Save this file to the same directory where you saved `swipe` and the database file.

3.4 Converting the database to binary format

Before you can actually run `swipe` to do the alignment, one more step is needed. The database file you have downloaded is in the FASTA format, which is text-based. Due to overheads from text encoding and filesystem access, directly using the text format to align sequences is very uneconomical. Therefore, it is necessary to convert the FASTA file into the BLAST binary format, which is originally developed by the US National Center for Biotechnology Information (NCBI). In order to do so, follow these steps:

1. Download NCBI BLAST+ (see link below), which is a set of tools used for alignment and data manipulation. (Download the version corresponding to your operating system. You can also download the source `-src` files and compile them yourself, but this should not be necessary.)
2. Extract the files, and copy the file `bin/makeblastdb` to the directory where you decompressed the Swiss-Prot database (`uniprot_sprot.fasta`).
3. Run the command

```
./makeblastdb -in uniprot_sprot.fasta -dbtype prot
```

4. When the conversion is done, you will find a few new files in the directory, with extensions such as `.pin`, `.phr` and `.phq`. These files are the binary BLAST database files.

3.5 Running `swipe`

Now you can run `swipe` to align your query protein with all the proteins in the database. To do it, use the command:

```
./swipe -d uniprot_sprot.fasta -i P00533.fasta > result.txt
```

Once the command finishes, the result of the alignment will be written to `result.txt`, you can open it with a text editor to see the kind of output you are expected to produce for your project.

The first thirty lines give you the technical information about this alignment such as the name and size of the database and the sequence to be aligned, the scoring matrix and gap penalties used, etc. After these lines comes the information about the speed of this alignment.

The most important lines in the output file are about the sequences in the database which align well with the query, sorted by the score obtained in the Smith-Waterman algorithm. In the example above, you will see lines like these:

```
                                                                 Score    E
Sequences producing significant alignments:                      (bits) Value

gnl|BL_ORD_ID|113555 sp|P00533|EGFR_HUMAN Epidermal growth facto...  2518   0.0
gnl|BL_ORD_ID|113556 sp|P55245|EGFR_MACMU Epidermal growth facto...  2503   0.0
gnl|BL_ORD_ID|113557 sp|Q01279|EGFR_MOUSE Epidermal growth facto...  2299   0.0
gnl|BL_ORD_ID|503379 sp|P13388|XMRK_XIPMA Melanoma receptor tyro...  1286   0.0
gnl|BL_ORD_ID|118509 sp|Q15303|ERBB4_HUMAN Receptor tyrosine-pro...  1264   0.0

[Further output omitted]
```

It is clear that the first line gives the correct result, since we know that our query corresponds to this particular protein. It is interesting to note, however, that the epidermal growth factor protein from macaques has a very close score. Therefore, if the scoring method is not well chosen, a human protein can easily be mistaken for a similar one from another (albeit very closely related) species!

4 The project

The goal of the project is to write a program similar to `swipe`, which uses the Smith-Waterman algorithm to align protein sequences. To achieve this goal, the following steps are necessary:

1. Understand how to manipulate the database and query files. Because your program needs to use the widely used FASTA and BLAST binary formats, it is essential that before implementing the algorithm you know how to read and manipulate the information in these files. The FASTA format is well explained on Wikipedia [Wik15a], and a PDF document explaining the BLAST binary database format can be downloaded from the *Université Virtuelle* [Far10]. Furthermore, you can easily find various resources explaining how a C++ program can read from a binary file.

2. Create a preliminary version of your project able to search for an exact match in the database. As a preliminary step, write a program that is able to find an exact match in the database. It should behave similarly as the swipe program, but rather than listing as output all sequences with a high score, it should only find in the database the exact sequence that you are looking for, and print its name, obtained from the database, in the output file (the name might have been replaced by a fake name in the query file to make the search more relevant). This will require you to correctly implement the input/output aspects of the project (including how to manipulate the database and query files), but rather than running the Smith-Waterman algorithm for each sequence in the database, this preliminary project should only perform an equality test between the query sequence and each sequence in the database. This preliminary version of the project should be delivered at the intermediate deadline of the project.

3. Understand the algorithm. Before you can implement the actual Smith-Waterman algorithm, you have to understand how it works. The Smith-Waterman algorithm was first proposed in [SW81], and later modified by Gotoh in [Got82]. A first step would be the careful reading of these papers, together with the `swipe` paper [Rog11] and any useful online resource you can find. In the algorithm, you will need to consider a few mathematical objects from outside the algorithm itself, such as the scoring matrix, the gap open penalty and the gap extension penalty. For the scoring matrix, NCBI has a series of them called the **BLOSUM** matrices (see below for download link). The one used by `swipe` as the default is the `BLOSUM62`. You should also use it as the default to facilitate comparison with `swipe`, but you should also allow your program to use a different matrix given by a file. The default values used by `swipe` for gap open penalty is 11 and gap extension penalty is 1, your program should also use these values as default but allow other values to be specified when the program is executed.

4. Implement the algorithm. After you are confident that you understand how the algorithm works and you have the technical ability to interact with the database and query files, you can start implementing the algorithm. The output of your program should be similar to that of `swipe`: you

may display some information about the database, the query and the speed. Your program must produce the list of sequences producing significant alignments, similar to the one printed before, with each line containing the name of the sequence and its score. Unlike in `swipe`, printing the actual query sequence and the aligned sequence in the database is not necessary.

5. (Optional) Optimize your code. Various optimizations are possible: from the simple multi-threading parallelization to sophisticated acceleration by using vector instructions in the CPU. It is also possible to use OpenCL or other GPGPU tools to achieve massive parallel processing using the graphics card.

5 Evaluation

In order to evaluate your work, your program will be tested using various query proteins. You can for example use the following proteins as test queries (a short, a medium and a longer one):

- `P07327`: Human alcohol dehydrogenase 1A
- `P00533`: Human epidermal growth factor receptor
- `Q9Y6V0`: Human piccolo

You will also explain your work during a brief defense, followed by a few questions. The following criteria will be used to grade the project:

- Satisfying all the requirements listed in Section 2
- Speed of your program
- Memory used by your program
- Overall quality of the code, including object-oriented aspects (well commented code is required to assess this criterion)
- Quality of the final defense, including the answers to the questions

Note that the first criterion (satisfying the requirements listed in Section 2) is required to obtain a passing grade. The next criteria will allow you to further increase your grade.

Download links

- Swipe, Smith-Waterman database searches with inter-sequence SIMD parallelisation: `https://github.com/torognes/swipe`
- Swiss-Prot database: `ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/uniprot_sprot.fasta.gz`
- Uniprot protein search: `http://www.uniprot.org/`
- NCBI BLAST+: `ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/`
- BLOSUM matrices: `ftp://ftp.ncbi.nih.gov/blast/matrices/`

References (articles available for download on *Université Virtuelle*)

[Far10] Michael S. Farrar, *NCBI BLAST Database Format*, 2010.

[Got82] Osamu Gotoh, *An improved algorithm for matching biological sequences*, Journal of Molecular Biology **162** (1982), no. 3, 705 – 708.

[Rog11] Torbjørn Rognes, *Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation*, BMC Bioinformatics **12** (2011), no. 1, 221.

[SW81] Temple F. Smith and Michael S. Waterman, *Identification of common molecular subsequences*, Journal of Molecular Biology **147** (1981), no. 1, 195 – 197.

[Wik15a] Wikipedia, *Fasta format*, `https://en.wikipedia.org/wiki/FASTA_format`, 2015.

[Wik15b] ———, *Proteinogenic amino acid*, http://en.wikipedia.org/wiki/Proteinogenic_amino_acid, 2015.