Bilkent University
CS481 | CS583 - Bioinformatics Algorithms

Spring 2025
Deadline: **March 21, 2025**

Homework Assignment #2

GENERAL INSTRUCTIONS

FAILURE TO FULFILL ANY OF THE FOLLOWING ITEMS WILL RESULT IN A GRADE SCORE OF 0 (zero) WITHOUT ANY CHANCE OF REDEMPTION.

- You must **write your code yourself**. Sufficient evidence of plagiarism will be treated the same as for plagiarism or cheating.
- You **cannot** use any libraries (e.g., Biopython) that provide the algorithms that are required in this assignment. You should implement the algorithms yourselves.
- **C / C++, Python or Java** will be used as programming language. STL is allowed. The assignments are created with C / C++ in mind, so using other languages would require minimum tweaking.
- Your code must **compile**.
- Your code must **be complete**.
- Your code must **run on dijkstra.cs.bilkent.edu.tr** server.
- Your code must make use of **argument parsing**.
  Refer to: `https://docs.google.com/presentation/d/1rUODhBg6yVXbfEtNuK73pjc1yWSPG9OYnGNH-b-kklQ`
- Submit your answers **ONLY** through the Moodle page.
- **Zip** your files and send them in only one zipped file.
- File name format `surname_name_hw#.zip`.
- The zip file must contain the following items:
  - All the source files.
  - `Makefile` to compile the source code and produce the binary. Even if you use Python, include this Makefile.
  - A `README.txt` file that briefly describes how your program works.
- All submissions must be made **strictly before the stipulated deadline**.
- A **bonus** will be given for the fastest code that solves the assignment successfully.

1) HELPFUL RESOURCES

In this assignment, you will need to perform traceback (i.e., backtrace) on the alignment matrix and create the **CIGAR** and **MD:Z** strings of the alignments. You can refer to these links following sources for help:

- Course Slides - 7
- `https://genome.sph.umich.edu/wiki/SAM#What_is_a_CIGAR.3F`
- `https://samtools.github.io/hts-specs/SAMv1.pdf`
- `https://samtools.github.io/hts-specs/SAMtags.pdf`

2) LOCAL AND GLOBAL ALIGNMENT

Aim: In this assignment, you will learn about the most common algorithms for sequence alignment, both global and local using Needleman-Wunsch and Smith-Waterman algorithms, respectively.

2.1 Data

- **Input**
  1. An option showing if the alignment to be performed is global or local. Given by the switch `-g` or `-l`.
  2. A FASTA file containing the patterns (i.e., reads) to search. The file is given using the `-p` switch. Each line corresponds to a pattern string P.
  3. A FASTA file containing the texts (i.e., references) to be searched. The file is given using the `-t` switch. Each line corresponds to a text string T.
  4. The scoring function for the match, mismatch, and gap penalties. The scores are given using the `-s` switch with this given order.
- **Output** Please see the details of the tasks below.
  1. For global alignment: A text file containing the pattern and text pair with the longest overlap, the CIGAR string, the MD:Z string, and their alignment score. The file is specified with the switch `-o`.
  2. For local alignment: A text file containing the pattern and text pair with the highest score, the CIGAR string, the MD:Z string, and their alignment score. The file is specified with the switch `-o`.

2.2 Tasks

The patterns and texts that are on the same line in their corresponding files constitute a pair. You can assume that the files have an equal number of lines (i.e., strings), and the length of the strings are the same. The pattern is the first component, and the text is the second component of the pair.

2.2.1 Global Alignment

1. Perform global alignment (i.e., NW) on each pair given in the files with the specified scoring function.
2. Applying traceback on the alignment matrix, create the CIGAR and the MD:Z strings for each pair.
3. Using the CIGAR string, determine if there is an overlap between the pattern or text. **An overlap** is defined as the longest exact match (i.e., denoted with the letter 'M' in the CIGAR) between the pattern and the text. The overlap can be observed anywhere in the alignment.

   Example overlap:

Text:	ATCAGCGATCATCGGCATAT
Pattern:	ATCAAGCGTCGGCATATGGC
CIGAR:	4M1D3M1I2M3I7M3D
MD:Z:	4^A12^GGC

4. Find the pair with the longest overlap among all pairs. You can output the first pair with the longest overlap if multiple pairs have the longest overlap length.
5. Report the pair with the longest overlap, its CIGAR and MD:Z strings, and its alignment score.

See below for the outputting format.

2.2.2 Local Alignment

1. Perform local alignment (i.e., SW) on each pair given in the files with the specified scoring function.
2. Find the pair with the highest alignment score among all pairs.
3. Applying traceback on the alignment matrix, create the CIGAR and MD:Z strings of the highest-scoring pair.
4. Report the pair with the highest score, its CIGAR and MD:Z strings, and its alignment score.

See below for the outputting format.

3) EXAMPLE

These examples show how the command line arguments and the outputs of your program should be:

3.1 Input

```
user@dijkstra$ cat patterns.fasta
> Pattern 1
GATTACACCCCCCCCCCCC
> Pattern 2
ATCAAGCGTCGGCATATGGC

user@dijkstra$ cat texts.fasta
> Text 1
GTCGACGCATTTTTTTTTT
> Text 2
ATCAGCGATCATCGGCATAT
```

3.2 Execution & Outputs

```
user@dijkstra$ ./hw2 -g -p patterns.fasta -t texts.fasta -o global.txt -s 1 -1 -1
user@dijkstra$ cat global.txt
Longest overlap:
pattern=ATCAAGCGTCGGCATATGGC
text=ATCAGCGATCATCGGCATAT
Score=8
CIGAR=4M1D3M1I2M3I7M3D
MD:Z=4^A12^GGC
```

```
user@dijkstra$ ./hw2 -l -p patterns.fasta -t texts.fasta -o local.txt -s 1 -1 -1
user@dijkstra$ cat local.txt
Highest local alignment score:
pattern=ATCAAGCGTCGGCATATGGC
text=ATCAGCGATCATCGGCATAT
Score=11
CIGAR=3M1I4M4D9M
MD:Z=7^ATCA9
```