Assignment 2

Hyoungshick Kim

October 30, 2023

1 k DNA sequences alignment

This is a programming assignment to test your understanding of dynamic programming.

- Your goal is to implement an efficient algorithm to implement multiple sequence alignment with k DNA sequences.
- A DNA sequence is a string formed from a four-letter alphabet {Adenine (`A`), Thymine (`T`), Guanine (`G`), Cytosine (`C`)} of biological macromolecules referred to together as the DNA bases. In this assignment, we will measure the similarity of genetic sequences by the frequency of the *exactly* matched alphabets. We align the k sequences, but we are permitted to insert gaps in either any sequence (e.g., to make them have the same length).
- You will write a code in the C programming language to print out the sequence alignment result into the output file named '`hw2_output.txt`' after finding the best sequence alignment from k DNA sequences in the input file named '`hw2_input.txt`'. The input file consists of (1) the number (k) of DNA sequences to be aligned; and (2) the k DNA sequences to be aligned. Each part is separated from the next part by a character `$`. The details are as follows:
  - The first part of the input file represents the number (k) of DNA sequences to be aligned.
  - The second part of the input file represents k DNA sequences; each sequence appears on a separate line of text.
- In the output file (`hw2_output.txt`), you should output the sequence alignment results with marks representing matched alphabets. In the last line, mark "`*`" on the columns containing identical alphabets across all sequences.

- The following is an example of input and output files:

```
[Input file: hw2_input.txt]
3
$
ATTGCCATT
ATGGCCATT
ATCCAAT

[Output file: hw2_output.txt]
ATTGCCA-TT
ATGGCCA-TT
AT--CCAAT-
**  *** *
```

- You will be judged by (1) the number of identical alphabets across all sequences returned by your submitted program, (2) the actual running time of the program and (3) the well-written document to explain your source code and the performance analysis of your algorithm. For test, we will use $2 \leq k \leq 5$, and $1 \leq n \leq 120$ where n is the maximum length of each DNA sequence. Also, 16GB RAM will be used for testing. Please test your code extensively with several inputs, so you are sure it works correctly.

- Your code should be written in ANSI C. We will use the GNU compiler (i.e., `gcc`) on Ubuntu Linux to compile your source code.

- You cannot use any pre-defined algorithms and data structures except arrays and strings. You should implement data structures (e.g., queue or linked list) by yourself if you need them. However, you can use I/O related functions.

- Please upload your source code (c files), a document to explain how your code works and the performance analysis of your algorithm to iCampus. All documents should be written in English.

- **Your assignments must be your own original work.** We will use a tool to check for plagiarism in assignments.