Assignment 6 — Dynamic Programming

Due: Wednesday, November 13th

Credit: This is a lab by Christopher Siu.

Many problems in bioinformatics amount to approximate string matching. For example, whenever a new gene is sequenced, searching known genes for close matches allows us to attempt to infer the new gene's functions.

Deliverables:

GitHub Classroom: `https://classroom.github.com/a/1wR915rB`

Required Files: `compile.sh`, `run.sh`

Optional Files: `*.py`, `*.java`, `*.clj`, `*.kt`, `*.js`, `*.sh`

Part 1: Sequence Alignment

Specifically, a gene is a string over the alphabet $\Sigma = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$, and the closeness of two genes is measured by the extent to which they are aligned. An *alignment* of two strings, x and y, is an arrangement of their characters in columns, in the same orders as they originally appeared, potentially interspersed by spaces.

For example, given $x = \texttt{ACGAT}$ and $y = \texttt{TACGCA}$, one possible alignment is:

```
A  -  -  -  C  G  A  T
T  A  C  G  C  -  -  A
```

…where the '`-`' character is used to indicate a space. Note that a typical further requirement is that no column may contain two spaces. Another possible alignment is:

```
-  A  C  G  -  A  T
T  A  C  G  C  A  -
```

Of these two alignments, the second appears to be better — certainly, it contains more exact matches. The *score* of an alignment is specified by a $(|\Sigma| + 1) \times (|\Sigma| + 1)$ *scoring matrix*, δ. For example, the latter of the above alignments has score $\delta(\texttt{-}, \texttt{T}) + \delta(\texttt{A}, \texttt{A}) + \delta(\texttt{C}, \texttt{C}) + \delta(\texttt{G}, \texttt{G}) + \delta(\texttt{-}, \texttt{C}) + \delta(\texttt{A}, \texttt{A}) + \delta(\texttt{T}, \texttt{-})$.

In your programming language of choice per Assignment 1, implement a dynamic programming algorithm to find the highest scoring alignment of two strings.

You may assume that both of the given strings will be non-empty and will contain only the characters '`A`', '`C`', '`G`', or '`T`'. You may also assume that the scoring matrix will always contain exactly 5 rows and 5 columns, each in the order '`A`', '`C`', '`G`', '`T`', '`-`', will always be symmetric about the diagonal, and will always provide integer scores.

For example, the above strings, along with a simple scoring matrix[1], would be represented as:

```
ACGAT
TACGCA
x A C G T -
A 1 0 0 0 0
C 0 1 0 0 0
G 0 0 1 0 0
T 0 0 0 1 0
- 0 0 0 0 0
```

Since this scoring matrix only rewards exact matches, in this situation, the second of the above alignments would indeed be better than the first (and, in fact, is the highest scoring alignment overall).

[1] Note that the scoring matrix is *not* your dynamic programming table; it is given to — not populated by — your algorithm.

Your program must accept as a command line argument the name of a file containing two strings and a scoring matrix as described above, then print to `stdout` the highest scoring alignment according to the following format:

- The first given string, assumed to be x, must be printed above the second given string, assumed to be y.
- The columns of the alignment must be space-separated, and '`-`' characters[2] must be used to represent spaces within the aligned strings.

For example:

```
>$ ./compile.sh
>$ ./run.sh in1.txt
x: - A C G - A T
y: T A C G C A -
Score: 4
```

You may further assume that the highest scoring alignment will be unique. Your program will be tested using `diff`, so its printed output must match *exactly*.

Part 2: Submission

The following items must be demonstrated in lab on the day of the deadline:

- Definition, base cases, formula, and solution for a dynamic programming algorithm to find the highest scoring alignment of two strings.

The following files are required and must be pushed to your GitHub Classroom repository by the deadline:

- `compile.sh` — A Bash script to compile your submission (even if it does nothing), as specified.
- `run.sh` — A Bash script to run your submission, as specified.

The following files are optional:

- `*.py`, `*.java`, `*.clj`, `*.kt`, `*.js`, `*.sh` — The Python, Java, Clojure, Kotlin, Node.js, or Bash source code files of a working program to align two strings, as specified.

Any files other than these will be ignored.

Additionally:

- There will be two test runs, one after 8pm the date before the due date and one at the end of lab (10am) on the due date. A feedback containing only what your program scores (on the same test cases that will be used to grade it). I may be able to give you an idea of what your program is failing on (if you have made an effort to test it), so please take advantage of these runs.
- Late submissions made within 24 hours of the deadline receive a .7 multiplier. If you decide to make a late submission, please notify me by sending an email to vnguy143@calpoly.eduwith both the subject and the body in the format: "CSC349,late,asgn<i>", i being the assignment number.

[2] This is the standard "hyphen-minus" character, ASCII code `0x2D`, not a dash.