CMSC 423: Project #1: Multiple Sequence Aligner

Due: Tuesday, November 15th at 11:59pm.

You will build a multiple sequence aligner. You'll implement the 2-approximation to MSA using affine gaps. Your program must be written in Java. You should work in groups of 2. Part of your grade for the project will be based on a confidential survey completed by your partner.

Your program will perform alignment using an affine gap scoring system with parameters read from a scoring matrix file and a multi-FASTA file (format given below) and output the 2-approximation to the MSA under the sum-of-pairs score also as a multi-FASTA file with gaps inserted.

Format of the command: Your program will be run using the following command:

```
java msa.jar score.matrix in.fasta out.fasta
```

Your program must run and compile on the submit server.

Format of the multi-FASTA file: The multi-FASTA file contains any number of sequences. A sequence begins with a line containing a `>` character in the first column. Following the `>` on the same line there is name for the sequence. This name ends at the first space or at the end of the line whichever comes first. Characters after the first space on this line are ignored. Every name must be unique among the sequences in the file; if you find a duplicate name, report an error (see below). On the line after the `>` line, the sequence begins. The sequence consists of any alphabetic character (`A-Z`,`a-z`). Upper- and lower-case are distinct. Spaces and newlines can be put anywhere in the sequence and should be ignored. The sequence ends with either the start of another sequence or the end of the file.

Example multi-FASTA file:

```
>seq1 this text is ignored
ACGTTTGAAATAGATACTGATCTGCATGACTAGACGAGTACGGGGGTATATATAAAAAA
>seq2
ACGTTTGAAATAG  ATACTGATCTGCATGA
CTAGACGAGTACGGGGGTATATATAAAAA
A
>seq3
    ATACTGAT
CTGCATGA
CTAGACGAGTACGGGGGTATATATAAAAA A
```

Format of the scoring matrix file: The file contains lines giving the cost of aligning two characters together. Each line is of the format:

```
c1 c2 score
```

where `c1` and `c2` are alphabetic characters or the gap character "`-`", and where `score` is a floating point cost of aligning `c1` to `c2`. The scores should be given with the understanding that the *minimum* cost alignment will be sought.

The special case where `c1`=`-` and `c2`=`-` gives the gap opening cost (the cost of starting a new run of gaps). The case where `c1`≠`-` and `c2`=`-` gives the gap extension cost for pairing `c1` with a gap. This represents a slight extension of the scoring systems we've seen in that a gap extension cost depends on what character will be aligned with the gap. The true cost of aligning a gap to a gap in a MSA is always taken to be 0 (and is not allowed in a pairwise alignment).

Example score matrix file:

```
- - 10
A - 5
T - 5
C - 5
G - 5
A A -5
A T 2.5
A C 2.5
A G 2.5
T T -5
T C 2.5
T G 2.5
C C -5
C G 2.5
G G -5
```

Above, the gap opening penalty is 10, the gap extension penalty is 5 no matter what the gap is aligned with. The cost of a mismatch is 2.5 and the cost of a match is -5.

Only one of `c1 c2` and `c2 c1` should be included in the scoring matrix. If both are, report an error as described below. If a score needed by the input sequences is not defined, report an error.

Handling multiple equally-good solutions: If there are multiple solutions when computing the alignment between a pair of sequences, you should locally favor matches and mismatches over gaps and favor gaps in the sequences listed first in the input file over gaps in the sequences listed later in the input when doing the traceback.

Format of the output: Your computed MSA should be output as a multi-FASTA file in the same format as described above, except that now sequences are allowed to contain "`-`" characters representing where you have placed the gaps. The filename for the output will be given on the command line. Sequences should not contain any spaces except that they should be wrapped so that each row of the sequence is 80 characters long (except the last row which might be shorter).

You should also output the sum-of-pairs score on the terminal in the format:

```
SP-score = -50
```

where `-50` is replaced by the actual computed SP-score for the alignment you have computed. You can (and probably should) output some informative messages to track the progress of your program. Only the above line should start with `SP-score`, however.

Reporting errors: If there is an error in the input FASTA file, you should output the following:

```
error: bad fasta file - DETAILED MESSAGE HERE
```

replacing the text "`DETAILED MESSAGE HERE`" with some short explanation of the error.

If there is an error in the scoring matrix, you should output the following:

```
error: bad scoring matrix - DETAILED MESSAGE HERE
```

replacing the text "`DETAILED MESSAGE HERE`" with some short explanation of the error.

In either case, your program should exit after printing the error message.

Grading: Your grade will be based on (a) correctness, (b) ability to achieve reasonable & practical speed, and (c) coding style.

Submission: You will submit using the CS submit server: `submit.cs.umd.edu`.