Homework 4 CSCI 451

Nicholas Rust

due: 05 November 2019

1 Exercise 2 (textbook section 6.9, pg 152)

Scoring:
Mismatch = 1
Gap = 2

```
ACTCTCGATC
ACT-TCGATC
```

Mismatch: 0
Gap: 1
Distance from S_1 to S_2 is 2.

```
ACTCTCGATC
ACTCTCTATC
```

Mismatch: 1
Gap: 0
Distance from S_1 to S_3 is 1.

```
ACTCTCGA-TC
ACTCTCTAATC
```

Mismatch: 1
Gap: 1
Distance from S_1 to S_4 is 3.

```
ACT-TCGATC
ACTCTCTATC
```

Mismatch: 1
Gap: 1
Distance from S_2 to S_3 is 3.

```
ACT-TC-GATC
ACTCTCTAATC
```

Mismatch: 1
Gap: 2
Distance from S_2 to S_4 is 5.

```
ACTCTCT-ATC
ACTCTCTAATC
```

Mismatch: 0
Gap: 1
Distance from S_3 to S_4 is 2.

Summation of distances for each string:
S_1 : 6
S_2 : 10
S_3 : 6
S_4 : 10

Therefore either S_1 or S_3 is the center string, here we use S_1.

```
ACTCTCGA-TC
ACT-TCGA-TC
ACTCTCTA-TC
ACTCTCTAATC
```

Notice when aligning the fourth string we have to put a gap in every preceding string as well as the center.

2 Exercise 3 (textbook section 6.9, pg 152)

Step 1: Compute optimal global alignment for each pair of strings. I did this above for center star.
Step 2: Use $1 - y/x$ to compute the distance matrix.

	S_1	S_2	S_3	S_4
S_1	0	0	.1	.1
S_2		0	.111	.111
S_3			0	0
S_4				0

Step 3: Compute the guide tree using neighbor joining magic. The order in which they are joined will be S_1, S_3, S_2, S_4.

Step 4: Follow the tree bottom-up to find the correct alignments, which will look something like:

```
ACTCTCG-ATC
ACT-TCG-ATC
ACTCTCT-ATC
ACTCTCTAATC
```

3 Exercise 4 (textbook section 6.9, pg 152)

Something similar to center star could be effective. Center star does a global alignment of the strings, if we wanted to find local alignments that maximize the score we could find a center string on the global level. We then do a pairwise local match with this string(S_c) and another(S_2), but take whatever subset of S_2 provides us with the most maximized score. We can then search for similar substrings of strings $S_3, ..., S_k$ using the z-algorithm (with room for a number of differences). This will give us a semi-decent guess of the most optimized local alignments. It won't be the fastest algorithm since we still have the O(n^2k^2) time complexity to find the distances between each string which will be the most significant time complexity.

4 Programming project: Center Sequence Finder

Figure 1: Program Output