AiB Project 2: Linear and Affine Global Alignments

Authors: Astrid Dahl (201806016), Ingeborg Bitsch Jensen (202005428), Henry Charlton (202303414), Carolina Jørgensen (201707243)

Introduction

Both of the mandatory aspects of the project are functional. They have been tested on the sequences given in the project statement.
The empirical running times show affine gap cost to be the faster algorithms of the two which is unexpected.

Methods

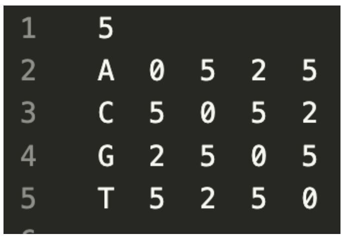
<u>General</u>
The programs were implemented as discussed in class, but use numpy arrays as a datatype. This is the only required library for the scripts to run. We implemented our own fasta parsing functions within the scripts that require them.

<u>Value Checking</u>
If the sequences provided (specified in the second input file) include characters that are not included in the alphabet (as specified in the first input file), the script will throw a ValueError.

Use: global_linear.py

This script takes two files:

1. A control file that specifies the gap cost and scoring matrix. This file has the following format, where the first line denotes the gap cost, and the remaining lines specify the scoring matrix. This is the format suggested in the project statement. Note that the appropriately formatted file is included in the global_linear_testing directory as "control.txt"



```
5
A 0 5 2 5
C 5 0 5 2
G 2 5 0 5
T 5 2 5 0
```

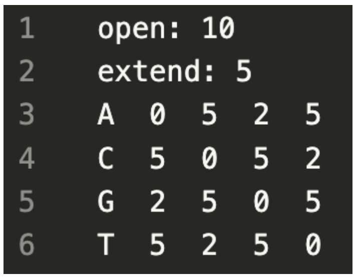
2. A fasta file with the two sequences to be aligned. This file must follow fasta format for it to be correctly parsed.

In addition to the two files, the scripts take a user input (y/n) asking whether to align the two sequences. If (y) the script will store the alignment in a new file {input_file}_alignment.fasta. If (n), it will just output the optimal alignment cost to the command line.

Use: global_affine.py

This script also takes two files:

1. A control file that specifies the gap opening cost, gap extension cost, and scoring matrix. This file has the following format, where the first line denotes the gap open cost, the second the gap extension cost, and the remaining lines specify the scoring matrix. This is the format suggested in the project statement. Note that the appropriately formatted file is included in the global_affine_testing directory as “control.txt” Also note: The open cost is the total cost for the first nucleotide of the gap (the extension cost only begins with the addition). The cost of a gap of length k is calculated as: g(k) = a + b*(k-1), where a is opening cost and b is extension cost.

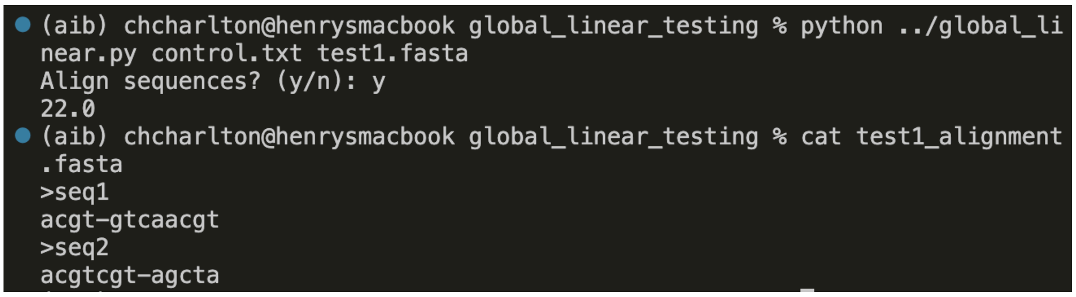


```
open: 10
extend: 5
A  0  5  2  5
C  5  0  5  2
G  2  5  0  5
T  5  2  5  0
```

2. A fasta file with the two sequences to be aligned. This file must follow fasta format for it to be correctly parsed.

In addition to the two files, the scripts take a user input (y/n) asking whether to align the two sequences. If (y) the script will store the alignment in a new file {input_file}_alignment.fasta. If (n), it will just output the optimal alignment cost to the command line.

Use: Example



```
● (aib) chcharlton@henrysmacbook global_linear_testing % python ../global_li
  near.py control.txt test1.fasta
  Align sequences? (y/n): y
  22.0
● (aib) chcharlton@henrysmacbook global_linear_testing % cat test1_alignment
  .fasta
  >seq1
  acgt-gtcaacgt
  >seq2
  acgtcgt-agcta
```

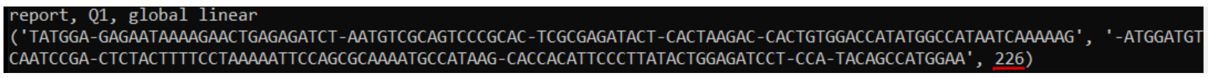
In this case the global_linear.py script is called on the control.txt and test1.fasta files in the global_linear_testing directory (where those files exists), and so it needs the ../ in front of it since the script exits in the outer directory. The script is instructed to store the alignment, which it does in the same directory in which it is called.

Test

We stored the example sequences given in the project statement into two directories, global_linear_testing and global_affine_testing. Within these two directories we also defined the control files (control.txt) to specify the appropriate gap cost(s) and scoring matrix. We then called our scripts on these files to check whether they gave the correct results. In each case they gave at least the correct optimal score. For the shorter sequences where the examples listed the possible optimal alignments, we were able to verify that the alignment our scripts produced was correct however, for the fourth (longest) sequence, it is not the case that all of the possible optimal alignments were given in the project statement and so we cannot verify correctness.

Eval_questions

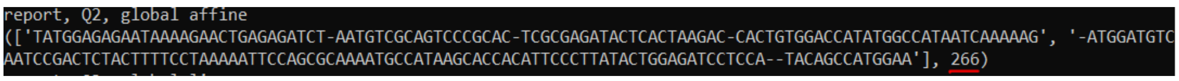
```
Question 1
----------

Compute the score of an optimal alignment and an optimal alignment of
seq1 and seq2 above using the programs global_linear using the above
score matrix M and gap cost g(k)=5*k
```



```
report, Q1, global linear
('TATGGA-GAGAATAAAAGAACTGAGAGATCT-AATGTCGCAGTCCCGCAC-TCGCGAGATACT-CACTAAGAC-CACTGTGGACCATATGGCCATAATCAAAAAG', '-ATGGATGT
CAATCCGA-CTCTACTTTTCCTAAAAATTCCAGCGCAAAATGCCATAAG-CACCACATTCCCTTATACTGGAGATCCT-CCA-TACAGCCATGGAA', 226)
```

The optimal score equals: 226.

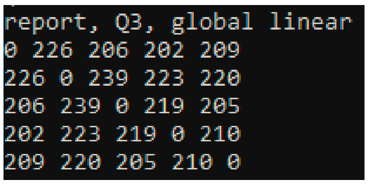
```
Question 2
----------

Compute the score of an optimal alignment and an optimal alignment of
seq1 and seq2 above using the program global_affine using the above
score matrix M and gap cost g(k)=5+5*k
```



```
report, Q2, global affine
(['TATGGAGAGAATAAAAGAACTGAGAGATCT-AATGTCGCAGTCCCGCAC-TCGCGAGATACTCACTAAGAC-CACTGTGGACCATATGGCCATAATCAAAAAG', '-ATGGATGTC
AATCCGACTCTACTTTTCCTAAAAATTCCAGCGCAAAATGCCATAAGCACCACATTCCCTTATACTGGAGATCCTCCA--TACAGCCATGGAA'], 266)
```

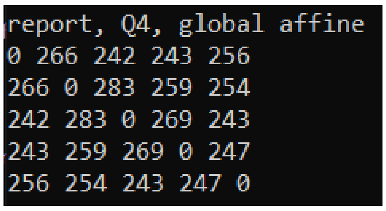
The optimal score equals: 266.

```
Question 3
----------

Compute the optimal score of an optimal alignment for each pair of the
5 sequences above using global_linear with the score matrix M and gap
cost g(k)=5*k. The result is a 5x5 table where entry (i,j) the optimal
score of an alignment of seqi and seqj.
```



```
report, Q3, global linear
0 226 206 202 209
226 0 239 223 220
206 239 0 219 205
202 223 219 0 210
209 220 205 210 0
```

```
Question 4
----------

Compute the optimal score of an optimal alignment for each pair of the
5 sequences above using global_affine with the score matrix M and gap
cost g(k)=5+5*k. The result is a 5x5 table where entry (i,j) the
optimal score of an alignment of seqi and seqj.
```



```
report, Q4, global affine
0 266 242 243 256
266 0 283 259 254
242 283 0 269 243
243 259 269 0 247
256 254 243 247 0
```

Experiments

Running times: Theoretical and Actual

Running time was measured using the Unix time command (user + sys).
In figure 1 we see that the empirical running time for the two algorithms follow each other as the input size increases (exponentially). However, opposite to their theoretical time complexities our implementations somehow result in affine gap cost being faster.
Still, we see in figure 2 and 3 that the implementations are contained within O(n^2) and O(n^3), respectively.

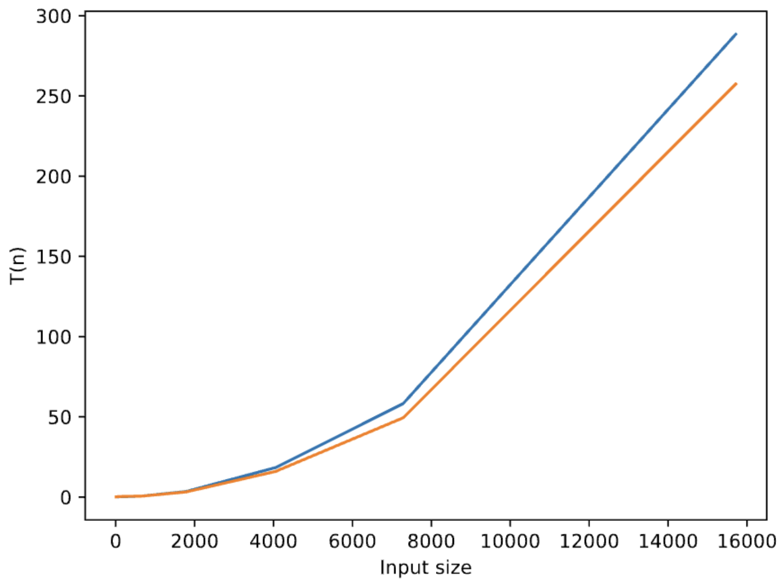


Figure 1: Plot of the actual running time for alignment using affine gap cost (orange) and linear gap cost (blue) showing how it depends on the input size (n).

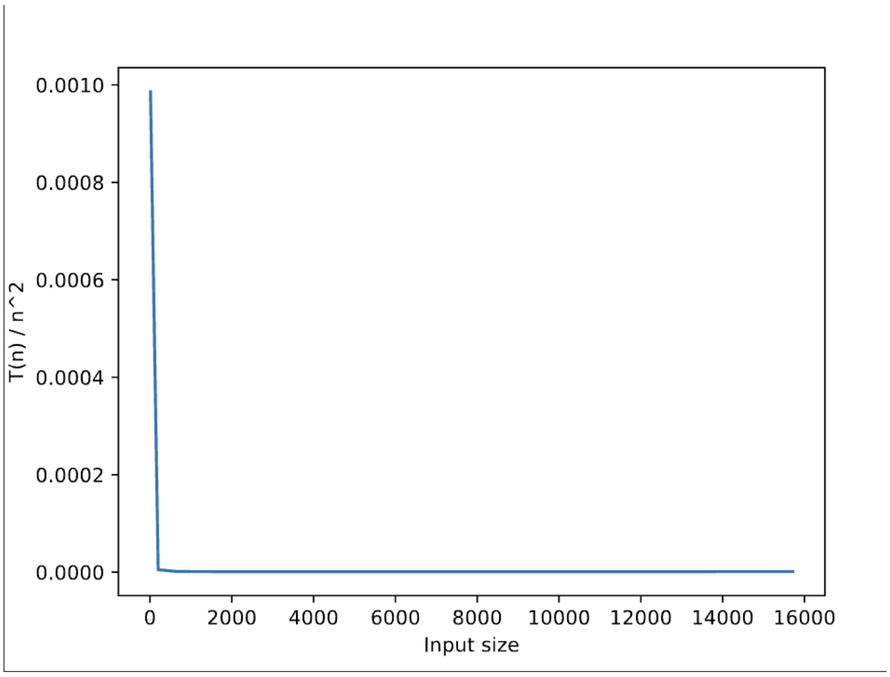


Figure 2: Empirical running time over the theoretical running time of linear gap cost. We see that all points are below some constant as n increases.

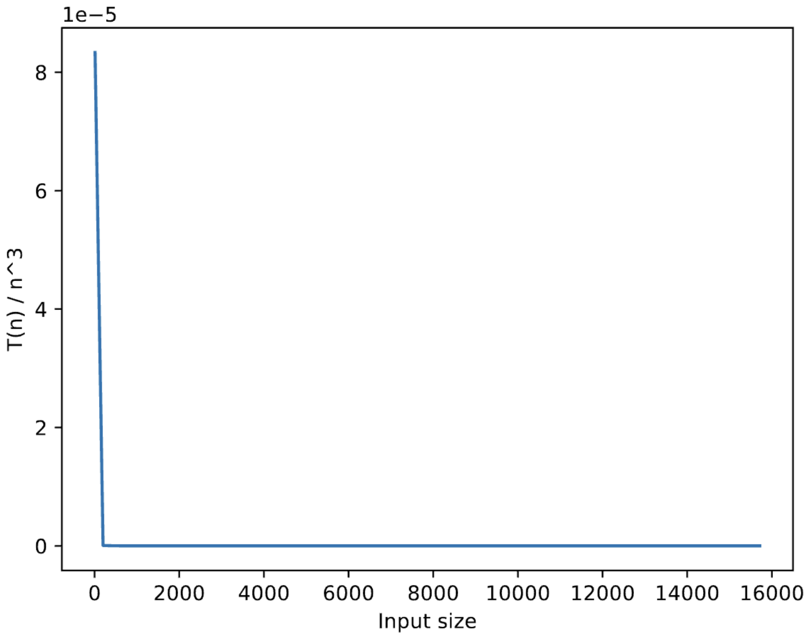


Figure 3: Empirical running time over the theoretical running time of affine gap cost. We see that all points are below some constant as n increases.