Assignment 03

CSE 241

May 20, 2022

1 Description

You are to write a C++ program which reads in two sequences of RNA nucleotides from standard input (`cin`). The four bases are Adenine, Cytosine, Uracil and Guanine, denoted A C U G, respectively.

Two sequences of RNA may stick together. RNA sequences will stick together better when they are *aligned* (shifted over).

Your job is to calculate and print out the best alignment, given two sequences of RNA.

1.1 Scoring alignments

To do this, you must calculate a *score* of how well two sequences are aligned. Calculate this score by summing up all the elements (base pairs) which are lined up with one another. A and U match up well together and get a score of +1. C and G match up well together and also get a score of +1. Any other combination will get a score of -1.

For example, consider the following two sequences:

```
A U A U A U C G U G
C C A U A U A U C C
```

To make it visually clearer, I will add annotations on a third row. `X` means a bad match (score -1) and `O` means a good match (score +1).

```
A U A U A U C G U G
C C A U A U A U C C
X X X X X X X X X O
```

This alignment would have a total score of -8. Not very good.

However, consider what would happen if we shifted the first sequence over one position to the right:

```
  A U A U A U C G U G
C C A U A U A U C C
  X O O O O O X O X
```

This alignment has a score of 3! Pretty good!

Your job is to find the best alignment.

1.2 Reading in input

First, read numbers into an array of enums. Keep reading as long as your inputs are one of A, C, U or G. Any other (non-space) character should be treated as the end of that input.

Next, read in numbers into a *second* array in the same fashion.

If, at any point, you get bad input, *print an error message and exit* `main` *with a return value of 1 immediately*. You should not attempt to recover from bad input in this assignment.

You may assume that the maximum sequence length is 1000 for both sequences. I.e., you may fix the size of your arrays to be 1000.

1.3 Functional requirements

You must create (at least) two functions:

`int score_without_realigning(rna_base const *, size_t, rna_base const *, size_t)` — this will score the two arrays based on how they are currently aligned, starting from the first element in each array. The two arrays do not have to be the same length. If they are not the same length, you would loop only up to the length of the *smaller* array. Return the score.

`int best_alignment(rna_base const *, size_t, rsa_base const *, size_t, long &)` — this will try all possible alignments and return the score of the best alignment. The last parameter (reference to `long`) is a reference to a variable that will indicate how the sequences had to be aligned. A negative value will indicate how far the first sequence had to be shifted to the right for the best alignment. A positive value will indicate how far the second sequence had to be shifted to the right for the best alignment. A value of 0 would indicate that neither sequence had to be shifted.

1.4 Output

You must output the best score and the final alignment. Here is an example usage of the program:

```
$ ./a
A U A U A U C G U G *
C C A U A U A U C C Z
Best score: 3
Best alignment:
  A U A U A U C G U G
C C A U A U A U C C
```

1.5 Testing

I will provide some testing data in the next few days.