CSCI 4470/6470 Project

Fall 2019

Due Friday November 22, 2019

This programming assignment is to develop a deliverable program that solves a realistic problem in computational biology research. Only dynamic programming (and no biological background) is needed to accomplish such a project. This handout give descriptions of two projects: 1. **Pairwise Alignment** and 2. **Structure Prediction**. Undergraduate students can choose either project while graduate students need to complete the second project.

1 Pairwise Alignment

In homework assignment #4, dynamic programming is suggested to solve a problem called Max Sequence Matching, which is an extension from the LCS problem, with the goal to maximize the sum of pairwise matching scores μ, instead of the count of exactly matched symbols between the two input sequences. This project **Pairwise Alignment** is a further extension from Max Sequence Matching. It requires that the score scheme $\mu(z, w)$ applies not only to two matched symbols z and w across the two sequences but also to unmatched single symbol z, scored by $\mu(z, \_)$ or $\mu(\_, z)$ depending upon if z occurs in the first or second sequence. Here underscore '_' is not a symbol in either sequence but called *gap* and used to indicate there is no matching for symbol z.

For example, between two sequences `relativity` and `reality`, the following matching

```
re lation
realit y
```

has the total score

$$\mu(\mathtt{r},\mathtt{r}) + \mu(\mathtt{e},\mathtt{e}) + \mu(\_,\mathtt{a}) + \mu(\mathtt{l},\mathtt{l}) + \mu(\mathtt{a},\mathtt{i}) + \mu(\mathtt{t},\mathtt{t}) + \mu(\mathtt{i},\_) + \mu(\mathtt{o},\mathtt{y}) + \mu(\mathtt{n},\_)$$

We call the matching an *alignment*, by putting gap symbol _ in wherever a symbol has no match in another sequence:

```
re_lation
realit_y_
```

This way, the two aligned sequences have the same length. The **Pairwise Alignment** is to find an alignment that achieves the highest total score, given two input sequences along with a score scheme μ.

1.1 Input data

Input data to your program are in two plain text files. The first file, named `sequences.txt`, contains two DNA sequences (of letters `A, C, G, T`) in the format:

```
>names of the sequences
ACGGCGCGTTCTT
AGTGCCTCCT
```

To facilitate your testing process, you may assume that the length of a sequence has no more than 200 letters.

The second file called `scheme.txt` contains a 5×5 matrix for score scheme μ, such that each cell is the matching score between the two symbols in the corresponding row and column. The file has the following format:

```
    A    C   G   T   _
A   5   -2  -2  -2  -6
C  -2    5  -2  -2  -6
G  -2   -2   5  -2  -6
T  -2   -2  -2   5  -6
_  -6   -6  -6  -6   0
```

1.2 Output

The output may be saved to a plain text file and should contain the optimal alignment score together with the alignment for the two input sequences based on the input score scheme μ, in the format (**note: an early result was wrong and have been reported by Michael Heam and Kang Tze Ng – thanks!**)

```
>alignment
 ACGGCGCGTTCTT
 A__GTGC_CTCCT
>score
 11
```

1.3 Reference

Sean Eddy, What is dynamic programming? *Nature Biotechnology*, Vol 22, No. 7, July 2004.
URL: `http://cobweb.cs.uga.edu/ cai/courses/algo/2019Fall/Readings/HowDPWorks.pdf`

2 Structure Prediction

This project is to compute the maximum number of permissible letter pairs from the input sequence of 4 letters (`A, C, G, U`). For these letters, legal pairs are `A-U, U-A, G-C, C-G` while others (such a `G-A`) are illegal. In addition, a pairing between two letters is exclusive, e.g., if a letter `U` on the sequence that is paired with a letter `A`, that letter `U` cannot be paired with another letter `A`. This problem is called **Structure Prediction** because it is a meaningful abstraction of the significant problem *RNA secondary structure prediction* in computational biology, where `A, C, G`, and `U` are nucleotide bases that constitute RNA sequences. A nucleotide base pair brings two bases physically together, contributing to the structure into which an RNA molecule may form. The maximization of such base pairs corresponds to the minimization of free energy that may stabilize the formed structure. The reference paper by Eddy gives more background to interested readers.

Problem **Structure Prediction** resembles the homework question Max Parenthesization (see Assignment #4), where the number of exclusive bracket pairs `()` and `[]` are to be maximized and can be solved with an "inside" dynamic programming method. But **Structure Prediction** problem permits pairs `)(` and `][` as well as `()` and `[]`. Nevertheless, pairings still have to be in nested and/or parallel fashions. For example, given input sequence `AUGUCAGCGUU`, the following structures are all legal:

```
AUGUCAGCGUU        AUGUCAGCGUU        AUGUCAGCGUU
{}{.}{{}.}.        {..}{.}{}..        {{{.}}.{}.}
```

where the denotational symbols { and } underlining the sequence indicate which two letters are predicted to be paired and the dot symbol '.' indicates a letter predicted to be not paired with any other.

2.1 Input Data

Input data to your program are in a plain text file, named `sequence.txt`. It contains an RNA sequence in the format:

```
>name of the sequence
AUGUCAGCGUU
```

2.2 Output

The output may be saved to a plain text file and should contain the maximum count of base pairs and the input sequence underlined with denotational symbols {, } and dot '.', in the format:

```
>AUGUCAGCGUU
 {{{.}}.{}.}
>max count of pairs
 4
```

2.3 Reference

Sean Eddy, How do RNA Folding Algorithms Work? *Nature Biotechnology*, Vol 22, No. 7, July 2004.
URL: `http://cobweb.cs.uga.edu/ cai/courses/algo/2019Fall/Readings/HowRNAFoldEddy.pdf`

2.4 Further Notes

It needs to be pointed out that answering the question posed in the **Structure Prediction** problem only offers a remote approximation to solving the underlying science problem. To make your program more realistically applicable, there are a few considerations that can be incorporated into your algorithm design. **These are optional, however.**

(1) A nucleotide base pair, e.g., `A-U`, can only be biologically plausible if there are at least 3 other bases between them. How would your dynamic programming handle this restriction?

(2) Since base pairs reduce free energy to achieve structural stability, contributions for different base pairs may be different, instead of being uniformly counted. In particular, your program can use a base pair *score function* γ to differentiate score contributions from different base pairs, e.g., $\gamma(\mathtt{A},\mathtt{U}) = \gamma(\mathtt{U},\mathtt{A}) = 2$, and $\gamma(\mathtt{G},\mathtt{C}) = \gamma(\mathtt{C},\mathtt{G}) = 3$, due to different numbers of hydrogen bonds needed for these base pairs. Then the objective function of the **Structure Prediction** is the maximize the total score contribution from base pairs. How would you modify your algorithm accordingly?

(3) In light of (2), there is another type base pairs `G-U` and `U-G` that can be considered. They are weaker than `A-U, U-A`, `G-C`, and `C-G` and have been omitted from our original problem description. We may

assume that $\gamma(\texttt{G},\texttt{U}) = \gamma(\texttt{U},\texttt{G}) = 1$. More realistically, the contribution function γ often comes in the form of (negative) *energy function* on these base pairs. Consequently, the objective function is to minimize the total energy contribution from involved base pairs. How would you modify your algorithm to include predictions of `G-U` and `U-G` pairs?

(4) Structural stability is not only determined by individual base pairs but also how the base pairs are organized together. While base pairs in a structure exist individually, "stacked" nested base pairs are biologically preferred over those non-stacked ones. For example, in the following for sequence `GCGAAACCGC`, the first structure would be preferred over the second:

```
GCGAAACCGC          GCGAAACCGC
{{{....}}}          {{{...}.}}
```

To allow structure prediction to prefer stacked base pairs, contribution score can be made more meaningful with consideration of every two stacked, nested base pairs, e.g., for the first structure predicted in the above example, the total score is contributed from the outermost pair stacked with the second outermost pair, i.e, `GC-GC` and from the second outermost pair stacked with the innermost pair `CG-CG`, etc.. You may assume that there is a predefined *stacking score scheme*, $\delta(x, u, v, y)$, in the forma of 6×6 matrix, where both (x, y) and (u, v) can be one of the 6 base pairs `A-U, U-A, G-C, C-G, G-U` and `U-G`. Under this assumption, only stacked base pairs will be predicted. That is, an isolated, single base pair will not be a part of the predicted structure.

Accordingly, recursive formulation for the objective function needs to be slightly adjusted to accommodate stacked, instead of individual, base pairs. How would you accomplish this?

3 Requirements

Undergraduate students (session 4470) can choose either project. Graduate students (session 6470) should complete project **2. Structure Prediction**.

For your project, you need to accomplish the following tasks:

(1) Design an objective function associated with an optimal solution and formulate recursive solution for the objective function;

(2) Write pseudocode for computing the objective function and pseudocode for traceback an optimal solution;

(3) Implement pseudocode in (2) into a program in any chosen programming language;

(4) **Submit a hard-copy report by due date to the instructor; it should include your completed work for parts (1) and (2)**;

(5) **Email to the TA (Mr. Di Chang) your program in a single zipped file** by the due date, which should include: *source code*, *executable code*, and *any auxiliary programs*;

(6) Schedule a demo date/time with and demo your program to the TA (Mr. Di Chang) by the end of classes.

Project report and demo that deviate from the requirements will not be graded.