CSE 5370: Bioinformatics
Homework 3

Due Monday April 3rd, 2023 at 8:59 CST

In this homework you will be writing code and reasoning about using dynamic programming algorithms for sequence alignment. The designed time to completion is 5-10 hours.

Logistics, Expectations, & Extra Credit

Assignment Submission & Specifications

Submissions should consist of both a *single* text write up page as well as however many code files are needed. In Question 4 you can also include the indicated portions of the problem as .txt files. These must be plaintext, solutions that are .rtf, .doc, .docx or other formats will not be graded and will receive zero credit. All of these files should be included in a single zip folder named "StudentLastName_UTAIDNumber_HW3_CourseNumber.zip" and submitted to Canvas. Submissions that do not conform to this naming scheme will be docked 20%. It is your responsibility to ensure that files are not corrupted (It is recommended to make sure that you test your .zip files to ensure that they can be decompressed) and that your code compiles/runs. Any corrupted files or code that does not compile or run will not be given credit. Grading of questions 2 and 3 will be done solely by the grading script. Non-typeset submissions will not receive credit. Code that does not conform to the specification will not be given credit.

Extra Credit

An extra 2.5% credit will be given to assignments where the text write up page is typeset with LaTeX(must include both the .tex and .pdf file inside your .zip file for credit). You will be required to typeset Homeworks 4 and 5 in LaTeX.

Academic Honesty & Office Hours

Many of the answers on CHEGG and similar sites that appear similar to questions on this assignment have incorrect answers. Students are encouraged to refer back to lecture recordings/slides and come to office hours before the assignment is due if they are struggling.

Late Submission Policy

All assignments and Projects are graded out of 100 points. Assignments submitted late will be penalized, at a rate of 4 penalty points per hour. The submission time will be the time shown on Canvas. Any assignment submitted more than 25 hours late will receive no credit for the assignment. Exceptions to late submission penalties will only be made for emergencies documented in writing, in strict adherence to UTA policy. For all such exception requests, the student must demonstrate that he or she made all efforts to notify the instructor as early as possible. Computer crashes, network crashes, software or hardware failure, temporary Canvas failure, email failure, will NOT be accepted as justification for late submissions. If you want to minimize chances of a late submission, aim to submit early.

Group Work

This is an individual assignment.

StackOverflow.com & Similar Sites

Use of stackoverflow.com and other sites is explicitly allowed (industry researchers and academic labs use these sites frequently). However, for this course you must include a comment in your code with the link to the page you referenced whenever these sites influence your own code writing. For example, when writing this homework assignment I forgot how to insert code into LaTeXdocuments and recalled how to after visiting stackoverflow.com. If I were submitting this as an assignment, I would want to include a comment like the below example in my code submission:

```
%When writing this homework assignment, I did not recall how to
%insert code in a nice looking way into LaTeX documents,
%so I referred to this page on stackoverflow for help:
%https://stackoverflow.com/questions/3175105
\usepackage{minted}
\begin{minted}[mathescape, linenos]{python}
Code To Insert in \LaTeX...
```

It is academic dishonesty to copy code from sites like stackoverflow without attribution like this, but is fine as long as you include attribution.

1 Substitution Matrices [10 points]

Recall that in class, we utilized a simplified substitution matrix when working alignment problems:

	A	G	T	C
A	1	-1	-1	-1
G	-1	1	-1	-1
T	-1	-1	1	-1
C	-1	-1	-1	1

Suppose that transition mutations ($A \longleftrightarrow G$ and $T \longleftrightarrow C$) are less common than tranversions ($A \longleftrightarrow T$, $A \longleftrightarrow C$, $G \longleftrightarrow T$, and $G \longleftrightarrow C$). Suggest a substitution matrix that reflects this.

2 Global Alignment [20 points]

Recall in class that we conduct global alignment with the Needleman-Wunsch algorithm. Implement the Needleman-Wunsch algorithm in a single python file called UTAIDNumber_NW.py that contains the following first two lines:

```python
class Solution:
    def global_alignment(self, sequence_A: str, sequence_B:str,
    ↪  substitution: dict, gap: int ) -> [tuple]:
```

Recall that to solve the Needleman-Wunsch algorithm, we construct a 2D matrix D, fill it it according to a recurrence relation, and then traverse it to calculate alignments. In the case of the Needleman-Wunsch algorithm, $D_{0,0} = 0$ and our recurence relation is that $D_{i,j} = \max \begin{cases} D_{i-1,j-1} + S_{x_i,y_i} \\ D_{i-1,j} + g \\ D_{i,j-1} + g \end{cases}$ where S is a substitution matrix represented by nested dictionaries and g is a linear gap penalty. Your implementation of the global_alignment function should take in sequence_A and sequence_B as the strings to be aligned (assume that these strings only contain the chars "acgt" for problem 2), a gap penalty, and a substitution matrix and then returns the list of tuples representing possible alignments. Your code will be tested with a variety of inputs by the grading script. If there are multiple alignments, please return the list of tuples. If there is only one alignment, please return a list of tuples of length 1.

2.1 An example

Suppose that we are aligning the strings "gata" and "ctag" with a match score of 1, a mismatch score of -1, and a gap score of -2. S would be represented by the following table:

As the problem specification says that S needs to be a nested dictionary, we will represent this as:

	a	g	t	c
a	1	-1	-1	-1
g	-1	1	-1	-1
t	-1	-1	1	-1
c	-1	-1	-1	1

```python
d = {'a': {'a':1,'t':-1,'c':-1,'g':-1}, 't':
↪  {'a':-1,'t':1,'c':-1,'g':-1}, 'c':
↪  {'a':-1,'t':-1,'c':1,'g':-1}, 'g':
↪  {'a':-1,'t':-1,'c':-1,'g':1}}
```

Our code should generate a 2d matrix D:

		C	T	A	C
	0	-2	-4	-6	-8
G	-2	-1 ↖	-3 ←↖	-5 ←↖	-7 ←↖
A	-4	-3 ↖↑	-2 ↖	-2 ↖	-4 ←
T	-6	-5 ↖↑	-2 ↖	-3 ↖	-3 ↖
A	-8	-7 ↖↑	-4 ↑	-1 ↖	-3 ←

Once D is generated, we traverse it by walking back along thee matrix starting at the bottom right to the cell where the best value was calculated from (arrows represent cell path was calculated from, red represents ideal path). As we a traversing D, a diagonal arrow represents a match or mismatch, so the letter of the column and the letter of the row of the origin cell will align. A horizontal or vertical arrow represents an indel. Vertical arrows will align a gap ("-") to the letter of the row (the "side" sequence), horizontal arrows will align a gap to the letter of the column (the "top" sequence). Thus, for the input strings "GATA" and "CTAC" and the paramters for S and g given in this example, your code should return the tuple ("GATA-","C-TAC").

3 Local Alignment [30 points]

Recall from in class that we conduct local alignment with the Smith-Waterman algorithm. Implement the Smith-Waterman algorithm in a single python file called UTAIDNumber_SW.py that contains the following first two lines:

```python
class Solution:
    def local_alignment(self, sequence_A: str, sequence_B:str,
    ↪  substitution: dict, gap: int ) -> [tuple]:
```

Recall that to solve the Smith-Waterman algorithm, we construct a 2D matrix D, fill it it according to a recurrence relation, and then traverse it to calculate alignments. In the case of the Smith-Waterman algorithm, $D_{0,0} = 0$ and our recurence relation is that $D_{i,j} = \max \begin{cases} D_{i-1,j-1} + S_{x_i,y_i} \\ D_{i-1,j} + g \\ D_{i,j-1} + g \\ 0 \end{cases}$ where S is a substitution matrix represented by nested dictionaries and g is a linear gap penalty. Your implementation of the local_alignment function should take in sequence_A and sequence_B as the strings to be aligned (assume that these strings only contain the chars "acgt" for problem 3), a gap penalty, and a substitution matrix and then returns the list of tuples representing possible alignments. Your code will be tested with a variety of inputs by the grading script. Note that your solution to this problem needs to only modify your code from problem 2 to adjust your recurrence relation and traversal logic. If there are multiple alignments, please return the list of tuples. If there is only one alignment, please return a list of tuples of length 1.

4 A Custom Alignment [40 points]

- Please take your first and last name in lowercase and concatenate them (state which values you use in your write up, for example I would take "jacob" and "luber" and concatenate this to "jacobluber").

- Write code to create a custom substitution dict S like used as input for Question 3, but with the difference that rather than have substitution values for "actg" we have substitution values for all 26 characters in the English alphabet. Include code for this problem in the file "UTAIDNumber_CUSTOM.py". Hard coded solutions will not receive credit. Make matches be valued +2 in S. Semi-matches are defined as the set of characters that are present in your concatenated name string. For example, "jacobluber" would yield the set {'o', 'j', 'r', 'c', 'e', 'l', 'a', 'b', 'u'}. Make semi-matches be valued +1 in S. Make mismatches be scored -1 in S. In the file, "UTAIDNumber_S.txt" provide the output from pretty printing your matrix S. An example of pretty printing is below:

```
>>>
↪  S=[['_','a','b','c'],['a',1,2,3],['b',4,5,6],['c',7,8,9]]
>>> np.matrix(S)
matrix([['_', 'a', 'b', 'c'],
        ['a', '1', '2', '3'],
        ['b', '4', '5', '6'],
        ['c', '7', '8', '9']], dtype='<U21')
```

- Run your "local_alignment" function from Problem 3 with the custom S defined by your name, a gap penalty of -2, your concatenated name (i.e. "jacobluber") as the first string, and the pangram "thequickbrownfoxjumpsoverthelazydog" as the second string. Include the output tuple(s) in your write-up.

- For the tuple that you generated immediately above, please pretty print the matrix D after all values are calculated prior to traversal and include this in the file "UTAIDNumber_D.txt"

Difficulty Adjustment

Your answers to this section will be used to adjust the difficulty of future assignments in the class.

- How long did this assignment take you to complete?
- If the assignment took you longer than the 10 hours, which parts were overly difficult?