CSCI3220 Algorithms for Bioinformatics 2018-2019 First Term
The Chinese University of Hong Kong

Assignment 1

Due date: 7 October 2017 (Sun) 23:59 Full mark: 100
Expected time spent: 3-6 hours

Aims:
1. Get familiar with concepts related to sequence alignment
2. Practice using dynamic programming to solve sequence alignment problems optimally
3. Establish scientific rigor by cross-checking your results using multiple means

Description:
In this assignment, you will first use dynamic programming to solve optimal sequence alignment problems, both manually and by writing a computer program. You will then study ways to skip some entries in the dynamic programming table that would not affect the final results. Finally, you will go through some steps of FASTA on some example sequences.

The bonus part is for those who enjoy challenges. The maximum score you can get from this assignment is capped at 100, but doing this bonus part would give you a better chance of getting a high score.

Questions:

In Questions **1-2**, we will use a scoring scheme of having **+2** scores for a match, **-1** score for a mismatch, and **-2** scores for an indel **without** applying affine gap penalty.

1. Fill in the following dynamic programming table to perform optimal **global** alignment between the sequences *r*=GTACC and *s*=CCTCC based on the scoring scheme stated above. Draw **all arrows** that lead to the score of each cell (i.e., only the "red arrows"). You can use the PowerPoint table template in the file CSCI3220_2018Fall_Assignment1_template.pptx if you want. Report the **optimal alignment score** and **all alignments** with this optimal score. When showing an alignment, indicate the names of the sequences clearly. (20%)

r \ s	C	C	T	C	C	Ø
G	1	3	2	-2	-6	-10
T	0	2	4	0	-4	-8
A	-1	1	3	2	-2	-6
C	-2	0	2	4	0	-4
C	-6	-4	-2	0	2	-2
Ø	-10	-8	-6	-4	-2	0

Answer:

Optimal alignment score = 1.

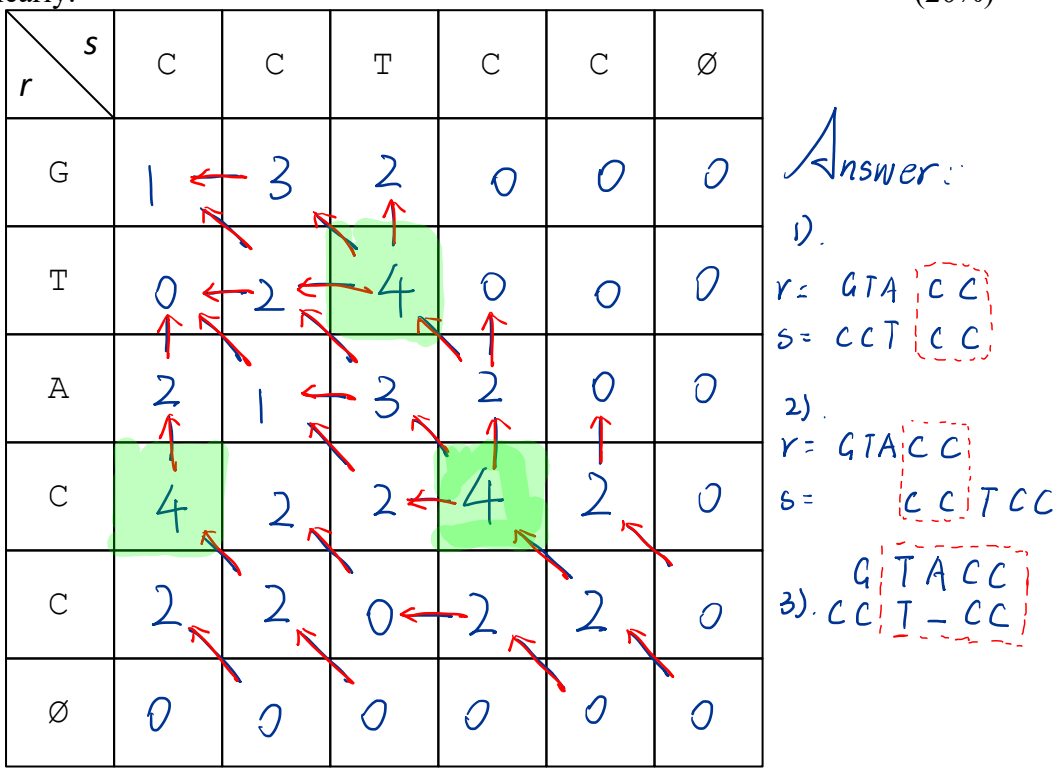
All alignment:

1). r: _GTACC
s: CCT_CC

2). r: G_TACC
s: CCT_CC

3). r: GTACC
s: CCTCC

2. Repeat Question 1 but perform a local alignment instead of a global alignment. When showing an alignment, indicate the sequence names **and the starting and ending positions of the aligned regions** clearly. (20%)



3. Write a computer program in C, C++, Java or Python for performing optimal pairwise **global** alignments of DNA sequences using dynamic programming, **without** affine gap penalty. Your program should take the following inputs from stdin (**not** from a file) in the specified order, each occupying a different line:

   i. Score for a match (a positive integer)
   ii. Score for a mismatch (a negative integer)
   iii. Score for an indel (a negative integer)
   iv. First DNA sequence (a text string)
   v. Second DNA sequence (a text string)

   You do not need to check for errors in the inputs.

   Your program should output to the screen (i.e., stdout) the **highest alignment score** and **all optimal alignments** in the following format:

   <Best alignment score><line break>
   [<line break>
   <first sequence with gaps filled<line break>
   <second sequence with gaps filled<line break>]

   The part in square brackets repeats for each optimal alignment. During the traceback, whenever a table entry has multiple incoming arrows, the horizontal one should be the first to traceback,

followed by the diagonal one, and finally the vertical one. **The resulting optimal alignments in the output should also follow this order.**

The non-comment portion of your program is expected to contain no more than 150 lines of code.

Here is a screen shot when a sample Java program called DP was run on an example from the lecture notes:

```
>java DP
1
-1
-1
ATGCGT
ACGGCGT
3

A_TGCGT
ACGGCGT

AT_GCGT
ACGGCGT

ATG_CGT
ACGGCGT
```

Your program will be graded based on i) whether it can be compiled/run successfully, ii) whether it follows the input/output formats as specified above, iii) its accuracy on a number of test cases and iv) whether the program is well-documented with appropriate comments added to explain the meaning of the code. In view of the large size of our class, for easy grading, you may get zero score if your program cannot be compiled or it does not conform to the input/output formats specified.

Do not forget to use your program to check your answer to Question 1, and the examples in the lecture notes and tutorial notes. (30%)

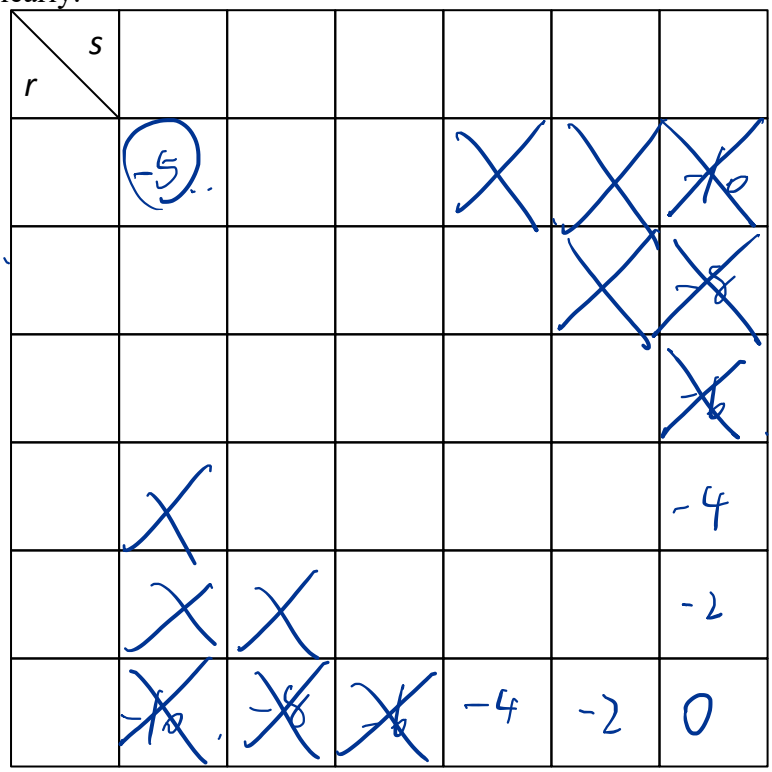
4. In Question 1, we found the optimal global alignment(s) by filling in the whole dynamic programming table. However, given the specific scoring scheme we considered (+2 for a match, -1 for a mismatch and -2 for an indel), it is actually not necessary to fill in the whole table.

   (a) List all the entries in the dynamic programming table that can never be involved in an optimal global alignment between any two length-5 sequences given the above scoring scheme. Explain why these entries can never be involved in an optimal global alignment. (Hint: Consider what score each table entry needs to achieve in order to be involved in an optimal alignment, and whether the entry can actually achieve this score.) (14%)

   (b) Based on your answer in Part a, explain how the dynamic programming algorithm can be modified to skip the table entries that can never be involved in an optimal global alignment, assuming that the table is still a 6×6 two-dimensional array. Give as much specific detail as possible. (6%)

   (c) [Optional] Now we generalize the results in Part a to two sequences of any lengths $|r|=n$ and $|s|=m \geq n$, and any reasonable scoring scheme without affine gap penalty. Specifically,

2. Repeat Question 1 but perform a local alignment instead of a global alignment. When showing an alignment, indicate the sequence names **and the starting and ending positions of the aligned regions** clearly. (20%)



(b).
1). Calculate the worst score = 5x mismatch score.

(5x-2)x2.

(4x-2)x2 + 2x1

(3x-2)x2+2x2.

3. Write a computer program in C, C++, Java or Python for performing optimal pairwise **global** alignments of DNA sequences using dynamic programming, **without** affine gap penalty. Your program should take the following inputs from stdin (**not** from a file) in the specified order, each occupying a different line:

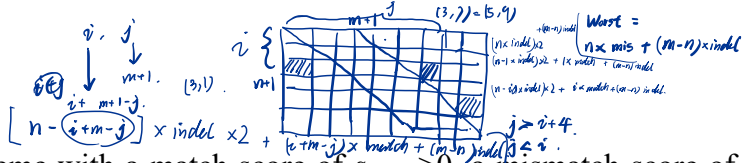
   i. Score for a match (a positive integer)
   ii. Score for a mismatch (a negative integer)
   iii. Score for an indel (a negative integer)
   iv. First DNA sequence (a text string)
   v. Second DNA sequence (a text string)

   You do not need to check for errors in the inputs.

   Your program should output to the screen (i.e., stdout) the **highest alignment score** and **all optimal alignments** in the following format:

   <Best alignment score><line break>
   [<line break>
   <first sequence with gaps filled<line break>
   <second sequence with gaps filled<line break>]

   The part in square brackets repeats for each optimal alignment. During the traceback, whenever a table entry has multiple incoming arrows, the horizontal one should be the first to traceback,



consider a scoring scheme with a match score of $s_{\text{match}}>0$, a mismatch score of $s_{\text{mismatch}}<0$, and an indel score of $s_{\text{indel}}<s_{\text{mismatch}}$. Derive a formula that determines whether the entry on the i-th row and j-th column (first row and first column counted as 1) in the dynamic programming table can never be involved in any optimal global alignment. Don't forget to use the formula you obtain for this general case to verify your answer to the special case in Part a. (bonus 5%)

5. This question is about FASTA.

   (a) Build a lookup table for the 2-mers in sequence r=`ACGCGTCA`. The table should be sorted by the 2-mers. (2%)

   (b) Now you are given another sequence s=`CCGTGCAC`. List all maximal diagonal runs of length 2 or more in the dot plot of r and s by recording the starting and ending positions in r and s without actually making the plot. A diagonal run is maximal if it is not contained by another diagonal run. (3%)

   (c) Suppose `lookup(kmer)` is a function that takes a k-mer as input and returns the list of all its occurrence locations in r as output. Give the pseudocode of an algorithm that can produce the output of Part b using this function. (5%)

Submission:

For the programming question (Question 3), you should first submit your program to our **online judge system**. Please see tutorial notes set 1 for explanations.

For the non-programming questions (Questions 1, 2, 4 and 5), give all your answers in a single file named <ID>_asmt1.<ext>, where <ext> is either doc, docx or pdf. We prefer pdf files because it has better portability, which guarantees for instance the arrows are displayed at the correct locations no matter on which machine the file is opened. If you need to use a doc or docx file, include images for your dynamic programming tables rather than drawing objects (if you use the PowerPoint template, choose "Paste Special" → "Picture (enhanced metafile)" or "PDF").

Both your written and source code files should contain the following header. Contact Kevin before submitting the assignment if you have anything unclear about the guidelines on academic honesty.

```
CSCI3220 2018-19 First Term Assignment 1

I declare that the assignment here submitted is original except for source
material explicitly acknowledged, and that the same or closely related material
has not been previously submitted for another course. I also acknowledge that I
am aware of University policy and regulations on honesty in academic work, and
of the disciplinary guidelines and procedures applicable to breaches of such
policy and regulations, as contained in the following websites.

University Guideline on Academic Honesty:
http://www.cuhk.edu.hk/policy/academichonesty/

Student Name: <fill in your name>
Student ID  : <fill in your ID>
```

Finally, submit **both** your programming source code and your file for the non-programming questions in a single zip package named <ID>_asmt1.zip to **Blackboard**.

Marking Scheme and Notes:

1. Remember to submit your assignment by 23:59pm of the due date. We may not accept late submissions.

2. For both the written part and the program, if you submit multiple times, **ONLY** the content and time-stamp of the **latest** one before the submission deadline will be considered.

University Guideline for Plagiarism

Please pay attention to the university policy and regulations on honesty in academic work, and the disciplinary guidelines and procedures applicable to breaches of such policy and regulations. Details can be found at http://www.cuhk.edu.hk/policy/academichonesty/. With each assignment, students will be required to submit a statement that they are aware of these policies, regulations, guidelines and procedures.