CS6013: Advanced Data Structures and Algorithms IIT Hyderabad, Fall 2020

Programming Assignment 3 (Sequence Alignment)

- Out on: **1st Dec 2020**. Due Date: **11:59pm, 10th Dec 2020**.
- The total marks for this assignment is 50 (scaled down to 10 for final weightage).
- **You may refer to internet sources like Wikipedia for clarifications (since the problems are well-known) but the written explanation in the report and implementation should be your own.**
- For submission and upload instructions, see the section after the questions.
- The sequence alignment problem and algorithm description is given in Section 6.6 of the Kleinberg-Tardos book.

---

1 Sequence Alignment

1.1 Problem Description

In this assignment, we will look at an application of Dynamic Programming to align DNA sequences. In the global DNA sequence alignment problem, you are given as input two DNA sequences S_1, S_2, which are strings using the four characters `A, C, T, G`. The strings are of lengths m, n, where possibly $m \neq n$. Given two DNA sequences, the goal is to compute the best possible (minimum) alignment cost between them. We will soon specify how to compute the alignment cost between two sequences.

Application: Why find the alignment cost? Suppose you want to find the role played by a DNA sequence you find in an organism. Instead of carrying out experiment after experiment trying to determine the function, we can use prior information to our advantage: there is usually a good chance that it is a variant of a known gene in a previously studied organism. DNA sequences and their functions in many species have been studied and published in publicly available databases. Thus, a more efficient way would be to compare our sequence for a match with a sequence in an already studied organism. However, DNA strands are subject to minor random mutations from organism to organism, so we cannot expect that we will find an exact match. Nevertheless, two sequences that are more or less similar can be expected to perform the same function. We try to align our (query) sequence with these target sequences, and compute an alignment cost; finding a low alignment cost with a target sequence would give us a reasonable guess about the role played by the query sequence in our organism.

Alignment and scoring:

Consider the target and query sequences (our first sequence will always be called the *target* and second the *query*): `AGGGCCT, TGGCA` respectively. Note that these are of different lengths, so one natural way to pair them would be:

```
A G G G C C T
T G G _ C _ T
```

Here, _ is called a *gap*, that can be inserted anywhere in either string, to make the rest of sequences align better. There is a penalty $s(x, y)$ for matching character `x` to character `y`. Consider the following scoring function:

$$s(x,y) = \begin{cases} 0, & \text{if } x = y \\ 2, & \text{if } x \neq y \\ 1, & \text{if } x = \_\_ \text{ or } y = \_\_ \end{cases}$$

Note that a smaller cost means better alignment. The cost of aligning a character with the gap character is usually denoted by γ, and is called the *gap-penalty*. According to the above scoring rules, the pairing given above has an alignment cost of $2 + 1 + 1 = 4$.

Consider aligning the target string: `AACAGTTACC` and query string: `TAAGGTCA`. Two possible alignments and corresponding scores are given:

```
           A A C A G T T A C C
           T A A G G T C A _ _
  Score:   2+0+2+2+0+0+2+0+1+1 = 10

         A A C A G T T A C C
         T A _ A G G T _ C A
    Score: 2 0 1 0 0 2 0 1 0 2  = 8
```

Thus, the second alignment is preferred. Note we may decide to insert gap characters in both target or query string to minimize the alignment scores, and not just in the string with smaller length.

Now consider another example: two possible alignments of `AGTACG` and `ACATAG`. Both have an alignment cost of 4, which, you can check, is the minimum alignment cost between the two strings.

Alignment 1:

```
A_GTACG
ACATA_G
```

Alignment 2:

```
A__GTACG
ACA_TA_G
```

Problem to solve

Given two sequences, write code to compute the cost and alignment pattern of the *optimal* alignment. The scoring $s(x, y)$ function will be specified as part of the input, along with the target and query sequences. The output should be (each on a new line):

1. The optimal alignment score (a single integer)
2. The pattern with gaps of the target sequence that is matched (without spaces)
3. The pattern with gaps of the query sequence that is matched (without spaces)

Important criterion: As seen in the last example above, there may be *multiple* output patterns possible with the same alignment score. You are to output the alignment pattern in which the matched **query** pattern is *lexicographically the least*, when read from left to right. You should consider the character ordering: `_< A < C < G < T`. The gap character is considered the smallest. e.g. `ACC_TG` is smaller then `ACCT_G`, because `_` is smaller than `T`. This ordering makes sense even for strings of unequal length: `AT_CGG` is smaller than `ATCC`.

In the third example given above, the second alignment should be given as output, and not the first.

1.2 Programming Specifications

Input:

Input is given on standard input.

1. 1st line: length of X (target sequence) and length of Y (query sequence)
2. 2nd line: gap cost and mismatch cost (the latter is the same for every pair of distinct symbols)
3. 3rd line: X sequence (no spaces, using characters `A,C,T,G`)
4. 4th line: Y sequence (no spaces, using characters `A,C,T,G`)

Example Input:

```
6 6
1 2
AGTACG
ACATAG
```

Output

Output will be to standard output. Use underscore _ for the gap character.

Example Output:

```
4
A__GTACG
ACA_TA_G
```

1.3 Programming Language

1. You may use one of C, C++, Java or Python for this.
2. You may use existing standard libraries (e.g: STL in C++) for data structures for this assignment; you need not code it from scratch.
3. Instructions (for programming and submissions) specific to each language will be updated soon. Some generic guidelines are: for C/C++, the programs should compile in gcc with the –std=c++14 switch. Python programs should compile on python3 . **Do not submit python notebooks**. Instructions will be similar to Assignment 2.

2 Submission Guidelines

1. Your submission will be one zip file named `<roll-number>.zip` , where you replace `roll-number` by your roll number (e.g. `cs20mtech11003.zip`), all in small letters. The compressed file should contain the below mentioned files:
   (a) Programming files (please do not submit python notebooks or IDE files). More detailed instructions will follow.
   (b) A description of your solutions in **pdf** format named `report.pdf`. This file should describe the algorithms you use, Pseudo-code is the preferred way to write out algorithmic concepts. If it is not an algorithm described in class, then you have to both describe it and argue that it is correct. You may state and used results proven in class without proof.
   (c) Upload your zip files on canvas. No delays permitted.
2. The preferred way to write your report is using LaTeX(Latex typesetting). However it is okay to submit pdf files not created in latex. **No handwritten files please!**
3. Failure to comply with instructions (filenaming, upload, input/output specifications) will result in your submission not being evaluated (and you being awarded 0 for the assignment). Do not print a single extra character over what is needed for the output.
4. **Plagiarism policy:** If we find a case of plagiarism in your assignment (i.e. copying of code, either from the internet, or from each other, in part or whole), you will be awarded a **zero** and will lead to a **FR** grade for the course in line with the department Plagiarism Policy (https://cse.iith.ac.in/academics/plagiarism-policy.html). Note that we will not distinguish between a person who has copied, or has allowed his/her code to be copied; both will be equally awarded a zero for the submission.

3 Evaluation

You will be marked for the following aspects:

1. Efficiency (time/space)
2. Correctness of Algorithm
3. Code clarity and comments
4. Report presentation