CSCI-570 Fall 2021
Analysis of Algorithms
Final Project

Due: Dec 8, 2021

I. Guidelines

The project is related to the implementation of the two different solutions provided in chapter 6 of the Kleinberg textbook for the *Sequence Alignment* problem. You can work on this project in groups of no more than 3 people.

II. Project Description

Implement the basic Dynamic Programming solution to the *Sequence Alignment* problem. Run the test set provided and show your results.

A. Algorithm Description:

Suppose we are given two strings X and Y, where X consists of the sequence of symbols $x_1, x_2 \ldots x_m$ and Y consists of the sequence of symbols $y_1, y_2 \ldots y_n$. Consider the sets $\{1, 2, \ldots, m\}$ and $\{1, 2, \ldots, n\}$ as representing the different positions in the strings X and Y, and consider a matching of these sets; recall that a matching is a set of ordered pairs with the property that each item occurs in at most one pair. We say that a matching M of these two sets is an alignment if there are no *"crossing"* pairs: if $(i, j), (i', j') \in M$ and $i < i'$, then $j < j'$. Intuitively, an alignment gives a way of lining up the two strings, by telling us which pairs of positions will be lined up with one another.

Our definition of similarity will be based on finding the optimal alignment between X and Y, according to the following criteria. Suppose M is a given alignment between X and Y:

1. First, there is a parameter $\delta_e > 0$ that defines a gap penalty. For each position of X or Y that is not matched in M — it is a gap — we incur a cost of δ.
2. Second, for each pair of letters p, q in our alphabet, there is a mismatch cost of α_{pq} for lining up p with q. Thus, for each $(i, j) \in M$, we pay the appropriate mismatch

cost $\alpha_{x_i y_j}$ for lining up x_i with y_j. One generally assumes that $\alpha_{pp} = 0$ for each letter *p*—there is no mismatch cost to line up a letter with another copy of itself—although this will not be necessary in anything that follows.

3. The cost of M is the sum of its gap and mismatch costs, and we seek an alignment of minimum cost.

B. Input string Generator:

The input to the program would be a text file, *input.txt* containing the following information:

1. First base string
2. Next *j* lines would consist of indices corresponding the after which the previous string to be added to the cumulative string
3. Second base string
4. Next *k* lines would consist of where the base string to be added to the cumulative string

This information would help generate 2 strings from the original 2 base strings. This file could be used as an input to your program and your program could use the base strings and the rules to generate the actual strings. Also note that the numbers *j* and *k* corresponds to the first and the second string respectively. Make sure you validate the length of the first and the second string to be 2^j*`str`$_1$`.length` and 2^k*`str`$_2$`.length`. Please note that the base strings need not have to be of equal length and similarly, *j* need not be equal to *k*.

```
ACTG
3
6
1
TACG
1
2
9
```

Using the above numbers, the generated strings would be
`ACACTGACTACTGACTGGTGACTACTGACTGG` and
`TATTATACGCTATTATACGCGACGCGGACGCG`

Following is the step by step process on how the above strings are generated.

```
ACTG
ACTGACTG
ACTGACTACTGACTGG
ACACTGACTACTGACTGGTGACTACTGACTGG

TACG
TATACGCG
TATTATACGCGACGCG
TATTATACGCTATTATACGCGACGCGGACGCG
```

C. Values for Delta and Alphas

Values for α's are as follows. δ_e is equal to 30.

	A	C	G	T
A	0	110	48	94
C	110	0	118	48
G	48	118	0	110
T	94	48	110	0

D. Programming/Scripting Languages:

Following are the list of languages which could be used:

1. C
2. C++
3. Java
4. Python
5. Python3

E. Bounds:

Bounds on the length of the base strings and the values of *m* and *n*, along with the zip file will be released on 17 November 2021, i.e. 3 weeks before the due date.

III. Goals

Following are the goals to achieve for your project

A. Your program should take input:

1. 2 strings that need to be aligned, should be generated from the string generator mentioned above.
2. Gap penalty *(δ_e)*.
3. Mismatch penalty *(α_{pq})*.

B. Your solution should output *output.txt* file containing the following information at the respective lines:

1. The first 50 elements and the last 50 elements of the actual alignment.
2. The time it took to complete the entire solution.
3. Memory required.

C. Implement the memory efficient version of this solution and repeat the tests in Part B.

D. Plot the results of Part A and Part B:

1. Single plot of *CPU time* vs *problem size* for the two solutions.
2. Single plot of *Memory usage* vs *problem size* for the two solutions.

IV. Notes and Hints

A. You will be provided a zipfile which has a folder containing some sample test cases and corresponding output text files containing the correct answers. Please note that we will be having another set of test cases (that are not released) which will be used for testing your program/script.

B. You should provide us with a zipfile containing your programs/scripts along with plots and a summary file.

C. The name of your program/script, folder and all the other meta-data files should have the USC IDs (not email ids) of everyone in your group separated by underscore, as follows:
    1. Zip filename: *1234567890_1234567890_1234567890.zip,* of the original foldername *1234567890_1234567890_1234567890.*
    2. Program filename: *1234567890_1234567890_1234567890_basic.py* **and** *1234567890_1234567890_1234567890_efficient.py* (if using Python).
    3. Plots: *CPUPlot.png* and *MemoryPlot.png* (you can use .jpg as well).
    4. Summaries: *Summary.txt.*

D. Your programs/script should take input as *input.txt* file as an argument and output an *output.txt* file.

E. Summarize your results and include any insights or observations drawn from your results in *Summary.txt*

F. You also need to state each group members contribution to the project, e.g.coding, testing, report preparation, etc. Do that in *Summary.txt.*

V. Grading

A. Basic Algorithm: **25** points.

B. If your program outputs a .txt file having all the 3 lines with correct answers: **15** points.

C. Memory efficient version: **30** points.

D. Plots, analysis of results, insights and observations: **30** points.