Assignment 6: Dynamic Programming
CS 758/858, Fall 2024
Due at **11:30pm on Wed, Oct 2**

Implementation

The skeleton code on the course web page is the start of a sequence alignment program (like BLAST for DNA or `diff` for text). Two files, each containing a string, are given on the command line. You must complete the program by finding the best alignment of the two strings using dynamic programming. We provide a simple enumeration algorithm that scales poorly, code to print the number of matching pairs in the best alignment, and code to print the alignment.

Testing

On the course web page, we supply skeleton code, a test harness, and some sample input files. Most of the programs we distribute in this class will tell you their command-line arguments if you run them with the `--help` option.

`align` is the program you are to complete. It takes three arguments on the command line, in the following order: solving algorithm, input file one, and input file two. Using the given solving algorithm, it will compute an alignment of the data in files one and two, and then print the appropriate output on `stdout`.

```
./align dfs one two
```

will perform an alignment using depth first search on files `one` and `two`, while

```
./align dyn one two
```

should compute the same alignment using dynamic programming.

`align-harness` runs your program, checks its output, and optionally displays a plot of the performance. For example:

```
align-harness -a dfs -a dyn -m 10000 -i 1000 -n 5
```

will run your alignment program with files containing random characters. Your program will be run 5 times on each file size where the file sizes are between 1,000 and 10,000 characters each and increase in increments of 1,000 for both the depth-first-search baseline as well as your dynamic programming solver. If the `-d` option is passed to the harness then a plot will be displayed to the screen showing a best fit quadratic curve to the performance of your alignment program. If you are running the harness on a system without an X display then you may use the `-o <file>` option to output the plot to a file instead (`<file>` should end in ".pdf",".ps" or ".png").

Be aware that, while the harness checks that you have aligned the strings legally, it does not verify that your alignment is the optimal one. There is also an unconfirmed report of problems if there are multiple optimal solutions (as always, let us know if you run into problems).

Written Problems

1. Briefly list any parts of your program which are not fully working. Include transcripts or plots showing the successes or failures. Is there anything else that we should know when evaluating your implementation work?
2. Exercise 14.3–2 in CLRS.
3. Exercise 14.3–5 from CLRS.

4. (Those in 858 only) Exercise 14.4–4 in CLRS.
5. (Those in 858 only) Part a of problem 14–10 in CLRS.
6. Parts b and c of problem 14–10 in CLRS. (You may assume part a.)
7. What suggestions do you have for improving this assignment in the future?

Submission

Electronically submit your work using the script on agate (eg, `~cs758/scripts/sub758 6-undergrad your-asn6-dir`).

Evaluation

In addition to correctness, your work will be evaluated on clarity and efficiency.
Tentative breakdown:

3 DP implementation (2 points for 858)

7 written problems (8 points for 858)