CS 330 – Longest Common Subsequence

deadline: Friday 11/22/19 at 11:59 pm

In this exercise you will implement a dynamic programming algorithm for finding the longest subsequence common to two strings. This algorithm was covered in the lecture notes (on Piazza: "Lectures 16-17 Dynamic Programming I and II"). Your code should run in $\mathcal{O}(n^2)$ time, where n is the length of the sequences. We've set rough upper bounds on execution time; if your code exceeds these, you'll see the limit and your code's running time.

Section 6.6 of *Algorithm Design* covers the problem of finding a minumum-cost alignment of the sequences, where there may be different costs for aligning various characters. If we take the following parameters:

$$\delta = \frac{1}{2},\ \alpha_{xy} = \begin{cases} 0 & \text{if } x = y \\ \infty & \text{otherwise} \end{cases},$$

the cost of the optimal alignment is n minus the length of the longest common subsequence (when the strings are the same length). These parameters force us to only match identical letters, and $\delta = 1/2$ counts the unmatched letters since there must be associated gaps in each string. It may be simpler to solve for the longest common subsequence directly, though.

Submission Format

On Gradescope submit a file (either `longest_common_subsequence.py` or `longest_common_subsequence.java`, names must match exactly) that reads a text file `input` and creates a text file `output` containing your solution. You may submit other files containing additional functions and classes. The autograder will put your code in the same directory as `input` and call it from the command line. Your code must create and write `output` in that directory.

Input

The file `input` will specify two strings, each n characters long. The characters will be drawn from the 26 uppercase English letters: A, B, C, ..., Z. The first line will contain n, written in base 10. The next two lines will each contain one of the strings.

We have uploaded files corresponding to the visible test cases on Gradescope. They may help you test and debug locally. Note that passing these test cases does not guarantee your code is correct!

Output

Write a text file `output` with the longest common subsequence on a single line. Write the **explicit string** you found, not the length.

Example

The following example comes from the slides and is padded with a "Q" to make the strings equal length. There are multiple possible longest common subsequences, any of which is acceptable.

`input:`

```
7
ABCBDAB
BDCABAQ
```

```
output:

BCAB
```