CS3000: Algorithms & Data — Summer 2023 — Laney Strange

Homework 2 - Long
Due Tuesday May 23 @ 9pm Gradescope

Name: Laney Strange
Collaborators: the Knights

- Put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This assignment is due Tuesday May 23 @ 9pm Gradescope. You may submit up to 48 hours late for no penalty, but expect a delay in grading.
- Show ALL your work, even if the problem doesn't specify it.
- You'll have an opportunity to resubmit two homeworks (one long, one short) at the end of the semester.
- Solutions must be typeset, preferably in LaTeX. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.
- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- If you get stuck on a homework problem, come by office hours or post on Piazza! We recommend you spend about 30 minutes trying to figure out a problem, and then ask for help. We'll be happy to clarify material from class and algorithm concepts, but we will not give out solutions or confirm your answers are correct.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class, is strictly forbidden.

Problem 1. *Mergesort (4 points)*

The pseudocode for the Merge step of Mergesort is at https://course.ccs.neu.edu/cs3000/resources/Mergesort_Pse
Copy it below, and modify the code so that, when you reach the end of either subarray, L or R, the while loop ends. Then, simply copy over the remaining elements (from the subarray we *haven't* reached the end of) into A. You'll need to modify the condition of the while loop, and then add some extra code after the while loop too. (If you'd like to make other changes too, such as in how the L and R arrays are created, you're welcome to but not required.)

Solution:

Problem 2. *Quicksort V2 (4 + 2 + 2 + 2 = 10 points)*

The version of Quicksort we've covered assumes you choose one pivot q and place everything smaller (or equal) to its left and everything larger to its right.

The Java version[1] of Quicksort picks two pivots q_1 and q_2 where $q_1 \leq q_2$. We partition the array into three parts: (1) all the elements less than or equal to q_1, (2) all the elements greater than q_1 and less than or equal to q_2, and (3) all the elements greater than q_2.

(a) Give the pseudocode for this new version of QUICKSORT – just the Quicksort procedure. You can assume the correct PARTITION procedure exists and returns the locations of the two pivots.

Solution:

(b) In the original version, we chose the pivot $q = A[r]$ and partition returns its final location. In this version, we'll choose $q_1 = A[p], q_2 = A[r]$. What would the following array look like after the first call to PARTITION? $< 7, 6, 12, 8, 10 >$

Solution:

(c) What would PARTITION return after being called that first time?

Solution:

(d) Give a recurrence for the best-case run-time of this new version of Quicksort and state its bound.

Solution:

[1] Implemented in JDK 7+ for sorting an array of primitives of length 47 or more.

Problem 3. *Dynamic Programming 1 (4 + 2 + 2 + 4 = 12 points)*

This problem refers to the DP algorithm we covered in class to find a Longest Common Subsequence.

(a) Draw the c table that would result from calling LCS-LENGTH on the sequences $X =< N, E, N, T, H, O >, Y =< E, N, O, E, N, H >$. We've started the outline for you, below.

	y_j	E	N	O	E	N	H
x_i	0	0	0	0	0	0	0
N	0						
E	0						
N	0						
T	0						
H	0						
O	0						

Solution:

(b) What is the length of the LCS for the X and Y in Part A? What is its value?

Solution:

(c) What are the 4 row/column pairs in your c table that indicate the letters of the LCS?

Solution:

(d) Write the pseudocode for an algorithm like PRINT-LCS, but one that doesn't use the b table. Instead, your algorithm should use the c table and sequences $X =< x_1, x_2, ..., x_m >$ and $Y =< y_1, y_2, ..., y_n >$ (it can also have additional parameters if you need them).

Solution:

Problem 4. *Dynamic Programming 2 (2 + 2 + 2 + 2 + 4 = 12 points)*

A closely related problem to Longest Common Subsequence is the Shortest Common Supersequence. In this version, we still start out with two sequences, X, Y. Our goal is to find Z, a shortest sequence possible with both X and Y as subsequences. In this version, we still start out with two sequences, X,Y .

For example, given $X =< A, C, B, C >$ and $Y =< C, D, E >$, the shortest common super sequence would be $< A, C, B, C, D, E >$ (length 6).

(a) If X is an empty sequence, and Y is any sequence with length n, what is the length of an SCS?

Solution:

(b) Given $X =< N, E, N >, Y =< E, N, O$, what is a shortest common supersequence? What is its length?

Solution:

(c) Suppose we add the same element to the end of X and the end of Y, so we have $X =< N, E, N, E >, Y =< E, N, O, E >$. What is the SCS now and what is its length?

Solution:

(d) Suppose we add two different elements to the end of X and the end of Y, so we have $X =< N, E, N, E, H >, Y =< E, N, O, E, N >$. What is the SCS now and what is its length?

Solution:

(e) Given the following recursive formula, fill in the table below for an SCS between $X =< N, E, N, E, H >, Y =< E, N, O, E, N >$. We use $c[i, j]$ to represent the length of an SCS of $X_i =< x_1, x2, ..., x_i >$ and $Y_j =< y_1, y_2, ..., y_j >$.

$c[i, j] = ...$

- j if $i = 0$
- i if $j = 0$
- $c[i-1, j-1] + 1$ if $x_i == y_j$
- $\min(c[i-1, j], c[i, j-1]) + 1$ if $x_i \neq y_j$

	y_j	E	N	O	E	N
x_i	0	1	2	3	4	5
N	1					
E	2					
N	3					
E	4					
H	5					

Solution: