CSCI 3104 Fall 2022
Instructors: Prof. Grochow and Chandra Kanth Nagesh

Problem Set 9

Due Date ..... November 7, 2022
Name ..... **Tyler Huynh**
Student ID **109603994**
Collaborators ..... **N/A**

Contents

Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Useful links and references on LaTeXcan be found here on Canvas.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.

- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section Honor Code). Failure to do so will result in your assignment not being graded.

Honor Code (Make Sure to Virtually Sign the Honor Pledge)

Problem HC. On my honor, my submission reflects the following:

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

In the specified region below, clearly indicate that you have upheld the Honor Code. Then type your name.

Honor Code. Tyler Huynh □

1 Standard 23: Dynamic Programming: Using Recurrences to Solve

Consider the KNAPSACK problem from lecture. Fill in the dynamic programming table for the following input: $[(1,3),(2,3),(2,2),(3,3),(4,5)], W = 7$. Here, each pair (v_i, w_i) denotes an item with value v_i and weight w_i. What is the value of the optimal knapsack choice? (You **do not** need to fill in back-pointers.)

Answer. I will first specify what the items, weight, and values are for our input:

Items: 1, 2 ,3, 4, 5
Values: 1, 2, 2, 3, 4
Weights: 3, 3, 2, 3, 5

I will now fill in our table to find he most optimal knapsack code, as follows:

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
(2, 2) 1	0	0	2	2	2	2	2	2
(1, 3) 2	0	0	2	2	2	3	3	3
(2, 3) 3	0	0	2	2	2	4	4	4
(3, 3) 4	0	0	2	3	3	5	5	5
(4, 5) 5	0	0	2	3	3	5	5	6

From the above we can see that the most optimal knapsack choice would be 6. □

2 Standard 24: Backtracking to find Solutions

Consider the KNAPSACK problem, with input $[(5,3),(6,4),(3,3),(7,5),(5,3)], W = 9$. Here, each pair (v_i, w_i) denotes an item with value v_i and weight w_i. The following is the dynamic programming table for the optimum value. From the table, **clearly indicate the steps you take to backtrace to find the optimum knapsack.** Be sure to include the steps at which an item was *not* chosen, as well as those at which it was chosen. That is, the number of steps your algorithm takes should be the same as the length of the list (5) (or maybe 6, depending on how you handle the last step).

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
(5,3) 1	0	0	0	5	5	5	5	5	5	5
(6,4) 2	0	0	0	5	6	6	6	11	11	11
(3, 3) 3	0	0	0	5	6	6	8	11	11	11
(7, 5) 4	0	0	0	5	6	7	8	11	12	13
(5, 3) 5	0	0	0	5	6	7	10	11	12	13

Answer. The steps to backtrack to find the optimum knapsack will be:

1. = The first step we would take would be to compare item 5 at column 9
2. = The second step we would take would be to compare item 4 at column
3. = The third step we would take would be to compare item 3 at column 4
4. = The fourth step we would take would be to compare item 2 at column
5. = The fifth step we would take would be to compare item 1 at column 0

From the above we have finished backtracking.

We can see that the most optimum knapsack would be:
13 where we take item 4 with value and weight (7, 5) and item 2 with value and weight (6, 4). □

3 Standard 25: Design a Dynamic Programming Algorithm (Synthesis Standard)

Problem 1. Recall from class that if x is a string, then a *subseequence* of x is a string of characters that occur in the same order as they do in x, but not necessarily contiguously. For example, abc is a subsequence of $aebeec$, but bca is not.

The Longest Reflective Subsequence problem is defined as follows.

- Input: A string x with characters from a finite alphabet Σ
- Output: A subsequence y of x such that for all $i \in \{1, \ldots, \ell = len(y)\}$, $y_i = y_{\ell - i + 1}$, that is as long as possible.

Examples.

1. Any subsequence of length 0 (the empty string) or length 1 (a single character) is reflective.
2. In $abasd; klfjasd; lfkjba$, the first two and last two characters together form a reflective subsequence $abba$. There are also several reflective subsequences of length 6, for example, $abjjba$, $abffba$, $abddba$, $ab;;ba$, $abkkba$. The longest are of length 9, for example $abkfdfkba$.
3. In 12**3**4567**898**53**5**, the longest reflective subsequence is 3589853, which we've put in **bold** in the original string.

The goal of this problem is to design a dynamic programming algorithm to solve the Longest Reflective Subsequence problem.

3.1 Problem 25(a)

(a) Let $L[i, j]$ denote the length of the longest reflective subsequence of $x[i, \ldots, j]$. Write down a mathematical recurrence for $L[i, j]$. Clearly justify each case.

Answer. For this problem we can see that the subproblem will be to denote the length of the longest reflective subsequence of $x[i, \ldots, j]$. From this I will write down a recurrence for $L[i, j]$, such that:

$$L[i,j] = \begin{cases} i = 0 & : if (i+j) \neq a_i \vee b_j \\ j = 0 & : if (i+j) \neq a_i \vee b_j \\ L[i-1, j] & : if (i+j) = a_i \\ L[i, j-1] & : if (i+j) = b_j \\ L[i-1, j] + L[i, j-1] & \end{cases}$$

□

3.2 Problem 25(b)

(b) Clearly describe how to construct and fill in the lookup table. For the cell $L[i, j]$, clearly describe the sub-cases we consider (e.g., using a dependency diagram), which optimal sub-case we select, and any relevant pointers that should be included in the table of back-pointers.

Answer. □

3.3 Problem 25(c)

(c) Work through an example of your algorithm using the input string $x =$ `uzwfzbu`. Clearly show how to recover an optimal solution by backtracing. You may hand-draw your table(s), but your explanation must be typed.

Answer. □