CSCI 3104 Spring 2023
Instructors: Chandra Kanth Nagesh and Prof. Ryan Layer

Quiz 21 Standard 21 – DP – Identify Precise Subproblems

Due Date	TODO
Name	**Blake Raphael**
Student ID	**109752312**
Quiz Code (enter in Canvas to get access to the LaTeX template)	**RTYZB**

Contents

Instructions

- You may either type your work using this template, or you may handwrite your work and embed it as an image in this template. **If you choose to handwrite your work, the image must be legible, and oriented so that we do not have to rotate our screens to grade your work.** We have included some helpful LaTeX commands for including and rotating images commented out near the end of the LaTeX template.
- You should submit your work through the **class Gradescope 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 **may not collaborate with other students**. **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, Discord, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section). Failure to do so will result in your assignment not being graded.

Honor Code (Make Sure to Virtually Sign)

Problem HC.

- 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.

Agreed (I agree to the above, Blake Raphael). □

21 Standard 21: Dynamic Programming - Identify Precise Subproblems

Problem 21. Consider the following problem:

> **Input:** A list $[v_1, \ldots, v_n]$ of values (each v_i is a positive number)
> **Output:** A subset $S \subseteq \{1, \ldots, n\}$ such that (1) no two adjacent items are in S (that is, S cannot contain both i and $i+1$), and (2) maximizing $\sum_{i \in S} v_i$, subject to (1).

For example, if the input is $[7, 4, 5]$, an optimal solution S is $\{1, 3\}$, with value $7 + 5 = 12$; if we had included item 2, then we could not include items 1 nor 3 because of the no-two-adjacent-items constraint (which would have resulted in a smaller value, namely 4).

Suppose you are going to solve this problem by dynamic programming; this can be done with a **one-dimensional** table T. **Clearly define** what subproblems $T[i]$ corresponds to, and which other indices j need to be considered when determining the value of $T[i]$. **Fill in the blanks**

Answer. We define $T[i]$ to be the highest optimal value achievable using a subset of the items $\{1, \ldots, n\}$. In the recurrence we'd have to try two cases which are if the values are adjacent and if they are not adjacent. If case 1 we add the largest value to S, then we recursively use addition to iterate our index, since it can't include adjacents. If case 2 then we add the values to S, then we recurse on a *max* function to iterate the index that will give us both the maximum value and no adjacents.

□

Problem 22. Consider the following problem:

> **Input:** Two strings s and t, with their lengths $n = |s|$ and $m = |t|$.
> **Output:** We return *true* if s is a **subsequence** of t, or $false$ otherwise.

For example, if the input is $s =$ "*agdc*", $t =$ "*ahbgdc*", the output will be *true*. Since, "*agdc*" occurs in t.

Suppose you are going to solve this problem by dynamic programming; **(1)** Answer what kind of dynamic programming table are we going to use to solve the problem. **(2)** Clearly define what subproblems corresponds to.

Answer.

1. We are going to be using a two- dimensional table $DP[s, t]$
2. We define $DP[s, t]$ to be the table containing the optimal value achievable using a comparison of s and t to see if either is a subset of the other. In the recurrence we'd have to try two cases which are if a letter $s[i] \in t$ or if a letter $t[i] \in s$ and if a letter $s[i] \notin t$ or if a letter $t[i] \notin s$. If case 1 we will return true and add the letter to our substring. If case 2 then we return false and then iterate to the next letter until we hit $s[n]$ or $t[m]$.

□