Homework 2.
Due: Thursday, February 6, 2020 before lecture via Gradescope.

[DPV] Practice Dynamic Programming Problems

Suggested reading: Chapter 6 of the book.

[DPV] Problem 6.4 – Dictionary lookup
You are given a string of n characters s[1...n],which you believe to be a corrupted text document in which all punctuation has vanished...
[DPV] Problem 6.17 – Making-change I
Given an unlimited supply of coins of denominations x1, x2, . . . , xn, we which to make change for a value v...
[DPV] Problem 6.18 – Making change II
Consider the following variation on the change-making problem (Exercise 6.17): you are given denominations x1, x2, . . . , xn, ...
[DPV] Problem 6.20 – Optimal Binary Search Tree
Suppose we know the frequency with which keywords occur in programs of a certain language, for instance ...
[DPV] Problem 6.26 – Alignment
Sequence alignment. When a new gene is discovered, a standard approach to understanding its function is to look through a database of known genes and find close matches...

Longest Common Sub*!?*
Given two strings $X = x_1, x_2, \ldots, x_n$ and $Y = y_1, y_2, \ldots, y_m$ **give a dynamic programming algorithm** to find the **length** k of the longest string $Z = z_1, \ldots, z_k$ where Z appears as a **substring** of X and as a **subsequence** of Y. Recall, a substring is **consecutive** elements.

For example, for the following input:

$$\begin{aligned} X &= a, \mathbf{b}, \mathbf{d}, \mathbf{b}, \mathbf{a}, b, f, g, d \\ Y &= \mathbf{b}, e, t, f, \mathbf{d}, \mathbf{b}, f, \mathbf{a}, f, r \end{aligned}$$

then the answer is 4 (since, b, d, b, a is a substring of X and it is also a subsequence of Y). You do not need to output the actual substring, just its length.
See next page for homework problems.

DP Homework

Problem 1 [DPV] Problem 6.1 – Maximum sum

A *contiguous subsequence* of a list S is a subsequence made up of consecutive elements of S...

(a) Define the entries of your table in words. E.g., $T(i)$ or $T(i, j)$ is ....

(b) State recurrence for entries of table in terms of smaller subproblems.

(c) Write pseudocode for your algorithm to solve this problem.

(d) Analyze the running time of your algorithm.

Problem 2 [DPV] Problem 6.8 – Longest common substring

Given two strings $x = x_1x_2 \dots x_n$ and $y = y_1y_2 \dots y_m$ we wish to find the length of their *longest common substrings...*

(a) Define the entries of your table in words. E.g., $T(i)$ or $T(i, j)$ is ....

(b) State recurrence for entries of table in terms of smaller subproblems.

(c) Write pseudocode for your algorithm to solve this problem.

(d) Analyze the running time of your algorithm.

Problem 3 [DPV] Problem 6.19 – Making change k

Given an unlimited supply of coins of denominations $x_1, x_2, ..., x_n$, we wish to make change for a value v using at most k coins...
(a) Define the entries of your table in words. E.g., $T(i)$ or $T(i, j)$ is ....

(b) State recurrence for entries of table in terms of smaller subproblems.

(c) Write pseudocode for your algorithm to solve this problem.

(d) Analyze the running time of your algorithm.

Problem 4 (The thief's plan)

A thief is planning on burglarizing some subset of n consecutive houses in a neighborhood. The houses are labeled $1, 2, \ldots, n$ and the thief will address then sequentially. The thief has an estimate of the profit to be earned from burglarizing each house $p_i, i = 1 \ldots n$, where $p_i > 0$. To avoid detection, he decides that he will never burglarize two adjacent houses, meaning that if he burglarize house 2, he cannot burglarize house 1 or house 3. Design a dynamic programming algorithm to determine the maximum total profit he can achieve.

Example: In each of the following two neighborhoods, the maximum achievable profit is \$100:
Case 1: $p = [\$20, \$100, \$30]$.
Case 2: $p = [\$40, \$30, \$10, \$60]$.

Your input is the list $[p_1, p_2, \ldots, p_n]$. Your output should be the maximum profit the thief can get. You don't have to return the list of houses the thief has to burglarize to achieve the maximum.

(a) Define the entries of your table in words. E.g., $T(i)$ or $T(i, j)$ is ....

(b) State recurrence for entries of table in terms of smaller subproblems.

(c) Write pseudocode for your algorithm to solve this problem.

(d) Analyze the running time of your algorithm.