Homework 2.
Due: Thursday, September 19, 2019 before 3pm EDT 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...

See next page for homework problems.

DP Homework

Problem 1 [DPV] Problem 6.8 – Longest common substring

(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 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.

(Faster (and correct) in asymptotic $O(\cdot)$ notation is worth more credit.)

(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 x1, x2, . . . , xn, 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 Maximum Product

The input to the problem is a string $Z = z_1 z_2 \dots z_n$ where each $z_i \in \{1, 2, \dots, 9\}$ and an integer k where $0 \leq k < n$. An example string is $Z = 8473817$, which is of length $n = 7$. We want to insert k multiplication operators $\times$ into the string so that the mathematical result of the expression is the largest possible. There are $n-1$ possible locations for the operators, namely, after the i-th character where $i = 1, \dots, n-1$. For example, for input $Z = 21322$ and $k = 2$, then one possible way to insert the $\times$ operators is: $2 \times 1 \times 322 = 644$, another possibility is $21 \times 3 \times 22 = 1386$.

Design a dynamic programming to **output the maximum product** obtainable from inserting exactly k multiplication operators $\times$ into the string. You can assume that all the multiplication operations in your algorithm take O(1) time.

(Faster (and correct) algorithm in $O(\cdot)$ notation is worth more credit.)

(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.