Suggested reading

Chapter 6 [DPV].

Practice Problems (<u>do not turn in</u>)

[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 6.8 – Longest common substring

Given two strings x = [1...n] and y = [1...m] we wish to find the length of their longest common substrings...

[DPV] Problem 6.18 – Making change II

Consider the following variation on the change-making problem (Exercise 6.17): you are given denominations $x_1, x_2, \ldots, x_n, \ldots$

[DPV] Problem 6.19 – Making change k

Given an unlimited supply of coins of denominations $x_1, x_2, \ldots, x_n$, we wish to make change for a value v using at most k coins...

[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

Graded Problem

A thief is planning on burglarizing some subset of n consecutive houses in a neighborhood. The houses are labeled 1, 2, . . . , n and the thief will address them sequentially. The thief has an estimate of the profit to be earned from burglarizing each house $p_i,\ i = 1\ \dots\ n$, where $p_i > 0$. To avoid detection, he decides that he will never burglarize two adjacent houses, meaning that if he burglarizes 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 table $p = [p_1, p_2, \dots, p_n]$. Your output should be the maximum profit the thief can get. You do not have to return the subset of houses the thief has to burglarize to achieve the maximum.

Please answer the following parts:

1. Define the entries of your table in words. E.g. T(i) or T(i, j) is ...
2. State a recurrence for the entries of your table in terms of smaller subproblems. Don't forget your base case(s).
3. Write pseudocode for your algorithm to solve this problem.
4. State and analyze the running time of your algorithm.

Faster (in asymptotic Big O notation) and correct solutions are worth more credit.

Submit your typed solution as a PDF in the Assignments section in Canvas. Hand-written submissions will receive a penalty.