Homework 1.
Due: Monday, September, 5 2022 before 8:00am via Gradescope.

[DPV] Practice Dynamic Programming Problems

Suggested reading: Chapter 6 of the book.

[DPV] Problem 6.1 – Maximum sum
A *contiguous subsequence* of a list S is a subsequence made up of consecutive elements of S...
[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 $x = x_1x_2 \dots x_n$ and $y = y_1y_2 \dots y_m$ we wish to find the length of their *longest common substrings...*
[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...

(Jumping frog)
The official pet of 6515 is a frog named René, who live in a nice pond at GeorgiaTech. The pond has n rocks in a row numbered from 1 to n, and René is trained to jump from rock i to rock $i + 1$ or rock number $i + 4$, where $1 \leq i \leq n - 1$. Find the number of ways René can go from rock 1 to rock n. Two ways are considered different if they jump on different subsets of rocks.

Example: for $n = 6$ there are three ways to get from rock 1 to rock 6. Those are:
$1 \to 2 \to 3 \to 4 \to 5 \to 6$,
$1 \to 2 \to 6$, and
$1 \to 5 \to 6$.

See next page for homework problems.

Problem 1 (Jumping frog II)

(*Read the last practice problem before you attempt this one! It is not necessary to solve the first one to do this one.*)

The current team of 6515 decided the pond was getting boring for René and decided to make it a bit nicer. We set a n by n grid of stones! Starting from (i, j), René learned to jump to $(i+1, j+2)$ or $(i+2, j+1)$. Find the number of ways to go from the top left corner $(1, 1)$ to the bottom right (n, n). (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 2 (electoral colleges)

In this problem, we want to determine the set of states with the smallest total population that can provide the votes to win the electoral college. Formally, the problem is the following: You are given a list of n states along with their population p_i, and the number of electoral votes v_i, for $1 \leq i \leq n$. Also, you are given Z, the number of electoral votes needed to win. All electoral votes of a state go to a single candidate. Our goal is to find a set of states S with the smallest total population that has at least Z electoral votes in total. You only have to output the total population of the set S, you do not need to output the set itself.

Example: if $n = 5$, populations are $P = [200, 100, 30, 700, 250]$, electoral votes are $V = [5, 1, 2, 7, 6]$ and $Z = 12$, then the solution is 480 since $480 = 200 + 30 + 250$ and states $1, 3, 5$ have $5 + 2 + 6 = 13$ electoral votes. Note in this example: $p_2 > p_3$ but $v_2 < v_3$, this might occur, but shouldn't affect your algorithm. Design a dynamic programming algorithm to solve this problem. (Faster (and correct) algorithm in big-O 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.