Homework #2

Due Time: 2023/04/28 00:00

Contact TAs: algota@noj.tw

Introduction and Rules

1. The judge system is located at https://noj.tw or https://v2.noj.tw, please submit your code by the deadline.
2. Homework Rule
3. Can I refer to resources from the Internet or other sources that are not from textbooks or lecture slides?
    - Yes, but you must include a reference source, such as a website or book title and page number, and attach it as a comment at the top of the code.
    - Although you can refer to external resources, please write your own code after the reference.
    - Remember to specify the references; otherwise we'll view it as cheating.
4. The Non-Programming part is no need to hand in. Just for practicing!

Programming Part

Please go to Normal Online Judge to read the programming problem.

Non-Programming Part

1. Warm up

Knapsack problem

ok

item	weight	value
1	2	4
2	3	2
3	1	2
4	4	3

有一個大包包，最大的負重是6

The capacity of the knapsack is 6, which means that the sum of the weights of items taken must be ≤ 6. Find the maximum possible sum of the values of items that fit into the knapsack.

Note that all kinds of the items are having exactly one. (How if there are unlimited number of instance of each item?)

(1).

Write down the state transition (recursive relation), and tabulate (draw a table) with the given items by using the transition you wrote.

Please label the meaning of the column and the row in your table.

Game of Stone

ok

There are N piles of stones $P_1,\ P_2,\ \ldots,\ P_N$, placed in a row, where N is even and P_i is the number of stones in $i-th$ pile. You play a game against an opponent by alternating turns. In each turn, a player has to select either the first pile or the last pile from the residual piles, take them away from the row, and obtains the number of the stones in that pile.

Determine the maximum possible amount of stones you can definitely obtain if you move first. Absolutely, both of the opponent and you will take the best strategy to win the game.

Define $dp(i, j)$ = the maximum possible amount of stones the player who moved first can get in the interval $P_i,\ P_{i+1},\ \ldots,\ P_j$.

(1).

Assuming you move first, taking a pile from $P_i,\ P_{i+1},\ ...,\ P_j$, then your maximum earnings is $dp(i,j)$.

Now in this round, if you take P_i (the first pile), fill in the blank "(A)" in the following equation.

$$dp(i,j) = P_i + \underline{\qquad}(A)\underline{\qquad}$$

On the contrary, if you take P_j (the last pile), what is the "(B)" blank should be.

$$dp(i,j) = \underline{\qquad}(B)\underline{\qquad} + P_j$$

Briefly explain your answer.

(2).

Follow the previous problem, write down the state transition, and tabulate with the following instance.

$$P_1,\ ...,\ P_6 = 2, 8, 3, 7, 5, 3$$

Build a table, and fill it with $dp(i,j)$ in cell (i,j).

Hint : the approach is similar to the Matrix-chain Multiplication problem.

2. LIS

Do you remember what LCS is? Now, let's learn about LIS (Longest Increasing Subsequence), which is often compared to LCS.

Given a sequence below:

$$s_1 = 1, 4, 2, 9, 10, 6$$

LIS (Longest Increasing Subsequence) is a classical problem which seeks to find the longest subsequence of a given sequence that is strictly increasing.

For the array above, the answer is 4 (1, 4, 9, 10 or 1, 2, 9, 10).

(1).

Show that how to find a LIS solution in a sequence using a naive approach. What is the time complexity of this algorithm?

✓ **(2).**

Given a sequence and an LIS solution, how to find the new LIS solution after appending a new element to the sequence?

Design an algorithm that can solve this problem in $O(N)$ time complexity.

Hint: Can you append the new element to the current LIS? If not, can you append it to other increasing subsequences?

(3).

We have learned that dynamic programming can solve problem by breaking it down into simpler sub-problems and recursively finding the optimal solutions to the sub-problems.

Can we solve LIS by dynamic programming? If so, what is the transition function for finding the LIS of a sequence? dp可以用來做LIS

Design an algorithm that can solve this problem in $O(N^2)$ time complexity.

Hint: Thinking about the meaning of $dp[i]$ can help inspire the transition function.

(4).

In fact, you can design an $O(nlogn)$ algorithm for LIS without using dynamic programming.

Design an algorithm that can solve this problem in $O(nlogn)$ time complexity.

3. LCS and LIS

Given two sequence $s_1,\ s_2$, both of them have N distinct positive numbers.

For instance:

$$s_1 = 1,\ 2,\ 3,\ 4$$

$$s_2 = 1,\ 3,\ 2,\ 4$$

(1).

Show that how to obtain the length of the Longest Common Subsequence of the given $s_1,\ s_2$ in $O(N \times N)$ time complexity, and analyze the space complexity of your method. $\rightarrow n^2$

For the above $s_1,\ s_2$, the answer is 3 (1, 2, 4 or 1, 3, 4). LCS

(2). 當我們在計算某一行的時候，我們只需要保留前一行的結果，所以只需要O(n)的空間。

So far in this course, we always want to make our algorithm more faster. However, space might also be expensive in some situation.

LCS

Design an algorithm that can do LCS problem in $O(N)$ space complexity.

Hint: 當你在推演 DP 表格的答案時，你真的需要 $N \times N$ 的空間來儲存資訊嗎？有沒有哪些是可以節省下來的？

(3).

The LIS approach also solves the LCS problem. Show that how to obtain the length of LCS of given s_1, s_2 by LIS approach.