CS 101 Fall 2021 - Quiz 10-B

12/21/2021 - 25 Minutes **Name:** **ID number:**

NOTE: Please write down the subproblem, recurrence equation and the time complexity. Breifly explain why.

Problem 1 Transform String (5 pts)

Given two strings s1 and s2, find out the minimum operations so that transform s1 to s2.

You can use 3 operations as following:

1. insert
2. delete
3. replace

For example,$s1$"$hors$", $s2 =$ "$rose$", $theoutputis3$, "$hors$"$->(replace)$"$rors$"$->(delete)$"ros"$->(insert)$"$rose$".(The order of the operations is not important)

Subprolem: dp[i][j] as the minimum operations transforming s1[1..i] to s2[1..j]

Bellman Equationi: $dp[i][j] = \begin{cases} dp[i-1][j-1], & s1[i] = s2[j] \\ \min(dp[i-1][j], dp[i][j-1], dp[i-1][j-1]) + 1, & s1[i] \neq s2[j] \end{cases}$

base case: dp[0][0...n],dp[0...m][0]

Problem 2 (5 pts)

Gezi Wang and his friends are going mountain climbing! There are K people, and each of them carries a bag. These packages have the same capacity W. There are N types of items and each type i has a given weight w_i and value v_i. In Gezi Wang's view, a reasonable backpack arrangement would look like this:

1.There can only be at most one item of each type in each bag, but the item of the same type can exist in different bags.

2.The total weight of items in each person's backpack is **exactly the same as the capacity of the bag.**

3.Any two people can't have exactly the same lists of things in their bags.

Assume there are plenty of lists to satisfy the requirement. Find an efficient algorithm to determine the maximum total value of all items in all bags, given the above requirements.

Subprolems: dp[i][j][m] as the mth large total value of the all k bags that capacity j can be filled exactly right considering the first i types of items. That means for each i and j, dp[i][j][1]... dp[i][j][k] are in descending order, which are the total value of all items in all k bags .

So we have the following recurrence equation:

$dp[i][j][1]...dp[i][j][k] = first_k(dp[i-1][j][1], .., dp[i-1][j][k], dp[i-1][j-w_i][1] + v_i, ..., dp[i-1][j-w_i][k] + v_i)$

The operation $first_k$ is the same as the operation in merge sort because dp[i][j][1]... dp[i][j][k] are in descending order, but we just need save the first k numbers.

Because the total weight of items in each person's bag is exactly the same as the capacity of the bag, so at first let $dp[0][0][1..k] = 0$ and the others is negative infinity.

The time complexity is $O(WNK)$.