CS 401
Solutions of Assignment #3

Problem 1: Design a dynamic programming algorithm for the following problem:

> *given an amount n and unlimited quantities of each of the coins $d_1, d_2, \ldots, d_m$, find the smallest number of coins that add up to n or indicate that the problem does not have a solution.*

Your algorithm should run in $O(mn)$ time. If you cannot get $O(mn)$ running time, provide the best possible running time you can get.

Solution:

Let $C(x)$ be the minimum number of coins needed to achieve a sum of x. Our goal is to determine $C(n)$. We initialize $C(x) = \infty$ for all $1 \leq x \leq n$. We then compute the entries in increasing order of x starting at $x = 1$ using the recurrence

$$C(x) = 1 + \min_{1 \leq i \leq m} \{C(x - d_i)\}$$

with the understanding that $1 + \infty$ is same as ∞. Computing each entry C takes $O(m)$ time, and thus the time to compute is $C(n)$ is $O(mn)$. If $C(n) = \infty$ then there is no solution to the problem.

Problem 2 Consider the following **more general version** of the Knapsack problem. There are p **groups of objects** $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_p$ and a knapsack capacity W. Each object x has a *weight* w_x and a *value* v_x. Our goal is to select a subset of objects such that:

- the total weights of selected objects is at most W,
- **at most one object is selected from any group**, and
- the total value of the selected objectsis *maximized*.

Suppose that n is the **total number of objects in all the groups** and V is the *maximum* value of any object, *i.e.,* $V = \max_{x \text{ is an object}} v_x$. Give an $O(nW)$ time algorithm for this general Knapsack problem. Explain why your algorithm is correct and analyze the running time of your algorithm.

Hint: Do a dynamic programming with increasing Knapsack capacity.

Solution:

This is very similar to the Knapsack problem seen in the class except that we have to consider groups of objects with the restriction that at most one object from any group may be selected. Let

$K(w, j)$ denote the optimal solution when the total weight of all selected objects is at most w and only objects from groups $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_j$ are considered. The recurrences are:

$$K(w,j) = \begin{cases} 0 & \text{if } w = 0 \text{ or } j = 0 \\ \max\left\{ K(w, j-1), \max\limits_{x \in \mathcal{O}_j} \{K(\max\{0, w - w_x\}, j-1) + v_x\} \right\} & \text{otherwise (if } w > 0 \text{ or } j > 0) \end{cases}$$

The time taken to compute $K(w, j)$ is $O(|\mathcal{O}_j|)$. Thus, the total time taken is $O\left(W \sum_{j=1}^{p} |\mathcal{O}_j|\right) = O(nW)$.