Hw6, Algorithm

1 Q1. Knapsack Problem

Given:

- Profits $P = [6, 3, 5, 4, 6]$
- Weights $w = [2, 2, 6, 5, 4]$
- Capacity $c = 10$
- Number of items $n = 5$

Recursive Solution

The recursive solution for the knapsack problem can be defined as:

$$K(n, c) = \begin{cases} 0 & \text{if } n = 0 \text{ or } c = 0 \\ K(n-1, c) & \text{if } w[n-1] > c \\ \max\{K(n-1, c), P[n-1] + K(n-1, c - w[n-1])\} & \text{if } w[n-1] \leq c \end{cases}$$

Base Case

For $n = 0$ or $c = 0$:

$$K(0, c) = 0, \quad \forall c$$

Recursive Case

For $n > 0$ and $c > 0$:

$$
\begin{aligned}
K(5,10) &= \max\{K(4,10), 6 + K(4,6)\} \\
K(4,10) &= \max\{K(3,10), 4 + K(3,5)\} \\
K(3,10) &= K(2,10) \quad (\text{since } w[2] = 6 > 10) \\
K(2,10) &= \max\{K(1,10), 3 + K(1,8)\} \\
K(1,10) &= \max\{K(0,10), 6 + K(0,8)\} = 6 \\
K(1,8) &= 6 \\
K(2,8) &= \max\{K(1,8), 3 + K(1,6)\} = 9 \\
K(3,5) &= \max\{K(2,5), 5 + K(2,0)\} = 5 \\
K(2,5) &= \max\{K(1,5), 3 + K(1,3)\} = 6 \\
K(4,6) &= \max\{K(3,6), 4 + K(3,1)\} \\
K(3,6) &= \max\{K(2,6), 5 + K(2,0)\} = 9 \\
K(2,6) &= \max\{K(1,6), 3 + K(1,4)\} = 9 \\
K(4,10) &= \max\{K(3,10), 4 + 5\} = 10 \\
K(5,10) &= \max\{10, 6 + 9\} = 15
\end{aligned}
$$

Dynamic Programming Solution

Let $DP[i][j] =$ Maximum profit using the first i items and capacity j

$DP[0][j] = 0 \quad \forall j$

$DP[i][0] = 0 \quad \forall i$

For $i = 1$ to n

For $j = 1$ to c

if $w[i-1] \leq j$ then $DP[i][j] = \max(DP[i-1][j], P[i-1] + DP[i-1][j - w[i-1]])$

else $DP[i][j] = DP[i-1][j]$

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	6	6	6	6	6	6	6	6	6
2	0	0	6	6	9	9	9	9	9	9	9
3	0	0	6	6	9	9	9	9	9	9	11
4	0	0	6	6	9	9	9	9	10	10	11
5	0	0	6	6	9	9	9	9	10	10	15

Table 1: DP Table for Knapsack Problem

Thus, the maximum profit for capacity 10 is 15.

2 Q2. 0/1 Knapsack Problem

Let $g(i,x)$ denote the maximum benefit for items $1,\ldots,i$ with a knapsack capacity of x.

(1) Recurrence Relation

The dynamic programming recurrence relation for the 0/1 knapsack problem is:

$$g(i,x) = \begin{cases} 0 & \text{if } i = 0 \text{ or } x = 0 \\ g(i-1,x) & \text{if } w_i > x \\ \max\{g(i-1,x), p_i + g(i-1,x-w_i)\} & \text{if } w_i \le x \end{cases}$$

Where w_i is the weight and p_i is the profit of the i-th item.

(2) Example Calculation

Given:

- Number of items $n = 4$
- Capacity $c = 20$
- Weights $w = [10, 15, 6, 9]$
- Profits $p = [2, 5, 8, 1]$

We need to compute the DP table and backtrack to find the optimal solution.

DP Table

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2
2	0	0	0	0	0	0	0	0	0	2	2	2	2	2	5	5	5	5	5	7	7
3	0	0	0	0	0	0	8	8	8	8	8	8	8	8	8	10	10	10	10	13	13
4	0	0	0	0	0	0	8	8	8	8	8	8	8	8	8	10	10	10	10	13	13

Table 2: DP Table for the Given Example

Backtracking to Find Optimal Items

To find the items that make up the optimal solution, we backtrack from $g(4,20)$.

- Start at $g(4,20) = 13$
- Since $g(4,20) = g(3,20)$, item 4 is not included.

- Move to $g(3, 20) = 13$
- Since $g(3, 20) \neq g(2, 20)$, item 3 is included.
- Now, $x = 20 - w_3 = 20 - 6 = 14$
- Move to $g(2, 14) = 5$
- Since $g(2, 14) = g(1, 14)$, item 2 is not included.
- Move to $g(1, 14) = 2$
- Since $g(1, 14) \neq g(0, 14)$, item 1 is included.

Thus, the optimal set of items to include are item 1 and item 3.

3 Q3. Task Completion Problem

Given n tasks numbered from 1 to n in topological order, each task i has two ways to complete:

- Method 1: Cost $C_{i,1}$ and Time $T_{i,1}$
- Method 2: Cost $C_{i,2}$ and Time $T_{i,2}$

Define $\text{cost}(i, j)$ as the minimum cost to complete tasks 1 to i within j time units.

The recurrence relation is:

$$\text{cost}(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ \infty & \text{if } i > 0 \text{ and } j < 0 \\ \min\{\text{cost}(i-1, j - T_{i,1}) + C_{i,1}, \text{cost}(i-1, j - T_{i,2}) + C_{i,2}\} & \text{if } i > 0 \text{ and } j \geq 0 \end{cases}$$

Base Cases

$$\text{cost}(0, j) = 0 \quad \forall j \geq 0$$
$$\text{cost}(i, j) = \infty \quad \forall i > 0 \text{ and } j < 0$$

Recursive Case

For $i > 0$ and $j \geq 0$:

$$\text{cost}(i, j) = \min\{\text{cost}(i-1, j - T_{i,1}) + C_{i,1}, \text{cost}(i-1, j - T_{i,2}) + C_{i,2}\}$$

This recurrence relation ensures that for each task i, we consider both methods of completion and choose the one that minimizes the total cost while staying within the given time j.

4 Q4. Maximum Subset Sum Problem

Given $\sum_{i=1}^{n} s_i \geq c$, where $\sum_{i \in J} s_i \leq c$ is an arbitrary positive integer, and J is a subset of {1, 2, ..., n}.

(1) Recurrence Relation

Define max_sum(i, j) as the maximum sum we can achieve using the first i elements without exceeding capacity j.

The recurrence relation is:

$$\text{max\_sum}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{max\_sum}(i-1, j) & \text{if } s_i > j \\ \max\{\text{max\_sum}(i-1, j), s_i + \text{max\_sum}(i-1, j-s_i)\} & \text{if } s_i \leq j \end{cases}$$

(2) Example to Illustrate the Algorithm

Consider $n = 4$, $c = 10$, and $s = [2, 3, 4, 5]$.

Step-by-Step Execution

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	2	2	2	2	2	2	2	2
2	0	0	2	3	3	5	5	5	5	5	5
3	0	0	2	3	4	5	6	7	7	9	9
4	0	0	2	3	4	5	6	7	9	10	12

Table 3: DP Table for Maximum Subset Sum Example

To find the subset that achieves the maximum sum not exceeding c, backtrack from max_sum$(4, 10)$:

- Start at max_sum$(4, 10) = 12$
- Since max_sum$(4, 10) \neq$ max_sum$(3, 10)$, item 4 is included.
- Now, $j = 10 - s_4 = 10 - 5 = 5$
- Move to max_sum$(3, 5) = 5$
- Since max_sum$(3, 5) \neq$ max_sum$(2, 5)$, item 3 is included.
- Now, $j = 5 - s_3 = 5 - 4 = 1$
- Move to max_sum$(2, 1) = 0$, so no more items are included.

Thus, the optimal subset includes items 3 and 4, achieving the sum 9.

Classification error rate(%) for ImageNet-to-ImageNet-C online CTTA task. Gain(%) represents the percentage of improvement in model accuracy compared with the source method.

Average mIoU(%) for the Citiscapes-to-ACDC. DI-A and DS-A represent the Domain-Invariant Adapter and the Domain-Specific Adapters. GNS means the Gradient Non-conflict Solver.