SC2001 Project 3

Dynamic Programming

Group 3

Gao Xin Yue U2322378A
Tio Sher Min U2320324H
Lee Pei Xin U2320668K

TABLE OF CONTENTS

01
Recursive definition

02
Subproblem graph of P(14)

03
Dynamic Programming

04
Compare P(14) with different weights

01
Recursive definition

Recursive Definition of P(C)

C: capacity

i = 0, 1, … , n-1

Base Case:

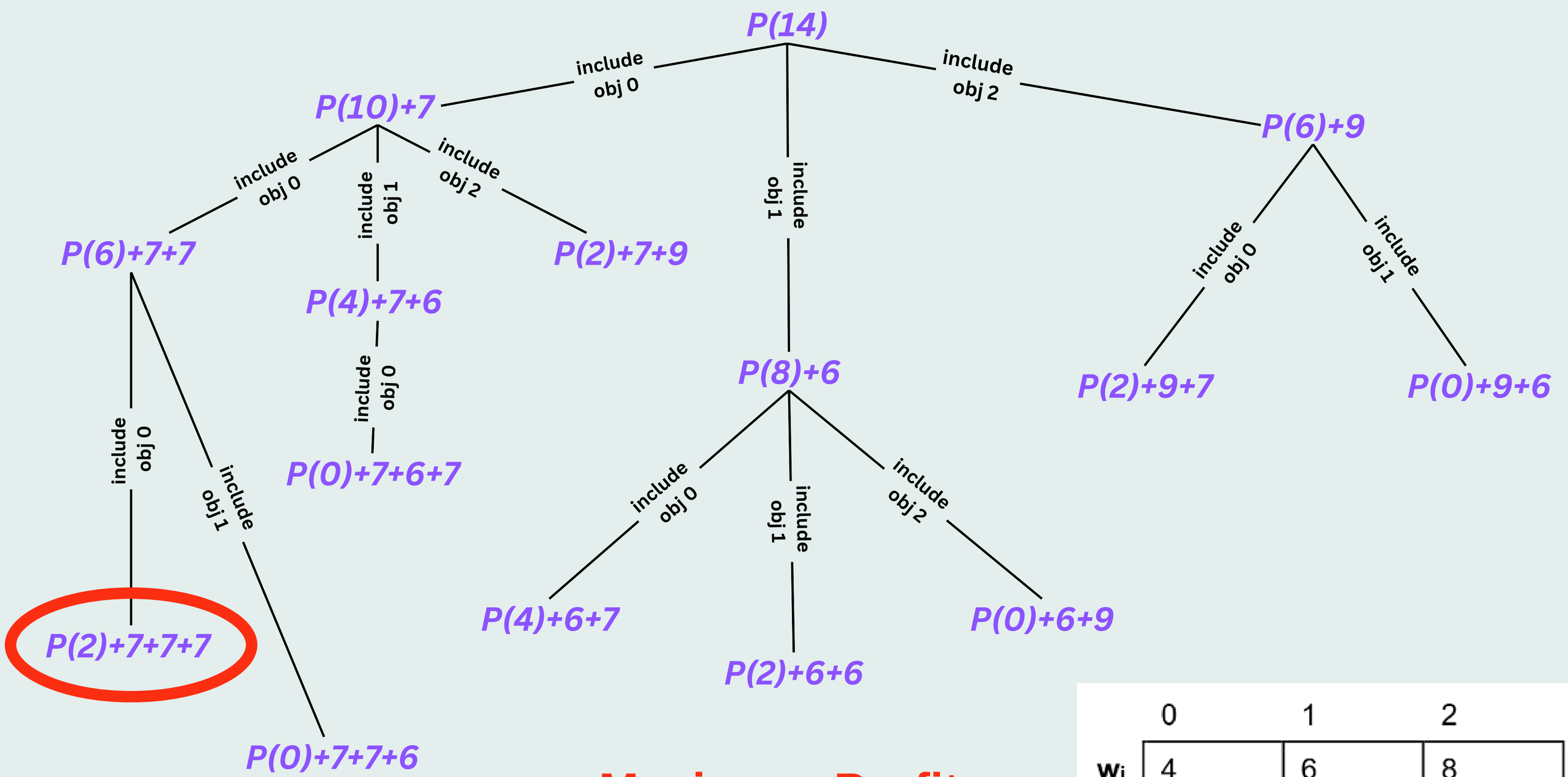
P(C)= 0 when C=0 or n=0

$$P(C) = \max(P(C), P(C-w_i) + p_i)$$

***ith* item is excluded**

***ith* item is included**

02
Subproblem graph of P(14)



Maximum Profit:
7+7+7 = 21

	0	1	2
w_i	4	6	8
p_i	7	6	9

03
Dynamic Programming

Dynamic Programming with Bottom-Up Approach:

1. **2D DP Array**: dp[i][j] stores the maximum profit for capacity j using items up to index i.
2. **Initialize DP Array**: Each row represents an item and each column represents a capacity from 0 to C.
3. **Outer Loop**:

- **iterates over each capacity** j from 1 to C.

4. **Inner Loop:**

- iterates over each item i.
- Excluding the item: If we exclude the item, dp[i][capacity] is set to dp[i - 1] [capacity] if i > 0, otherwise it's 0 (base case).
- Including the item: If weights[i] <= capacity, the code calculates the profit by including the item, **allowing reuse by taking dp[i][capacity - weights[i]] + profits[i]**

5. **Final Solution**:

- dp[n-1][C] holds the maximum profit for capacity C using all items.

Dynamic Programming with Bottom-Up Approach:

```python
def max_profit(C, weights, profits):
    n = len(weights)
    # Initialize a 2D DP array where rows are items and columns are capacities
    dp = [[0] * (C + 1) for _ in range(n)]

    # Fill the DP table
    for capacity in range(1, C + 1):
        for i in range(n):
            # Option 1: Exclude the item
            dp[i][capacity] = dp[i - 1][capacity] if i > 0 else 0

            # Option 2: Include the item (if capacity allows)
            if weights[i] <= capacity:
                dp[i][capacity] = max(dp[i][capacity], dp[i][capacity - weights[i]] + profits[i])

    # The maximum profit with the given capacity and all items is in dp[n-1][C]
    return dp[n - 1][C]
```

The code **allows item reuse** by **repeatedly updating dp[i][capacity] with dp[i][capacity-weights[i]] + profits[i] whenever an item fits.**

Time Complexity

Outer Loop (Capacities):

The outer loop iterates over each capacity from 1 to C, so it runs C times.

Inner Loop (Items):

For each capacity, the inner loop iterates over all n items.

In Each Iteration:

Within the inner loop, the program checks two options (excluding and including the item) and performs a constant-time update of the dp array.
Overall Time Complexity:

Since the outer loop runs C times and the inner loop runs n times for each capacity, the total time complexity is: **O(n×C)**

Space Complexity

DP Array:

The 2D dp array has n rows (one for each item) and C + 1 columns (one for each capacity from 0 to C), resulting in a space complexity of: O(n×C)

vs Recursive Solution (Without Memoization)

- try each item at every capacity level, making recursive calls for each item that can fit within the remaining capacity.
- The recursion continues until the remaining capacity is exhausted (or until there are no items left to consider).

Time Complexity

For each capacity C, we have n choices (one for each item) if it fits. For each choice, the function calls itself recursively with a smaller capacity (C - weight[i]).

Branching Factor: **At each step, the function can branch out n times (one for each item), and it can potentially reach down to a depth of C** (if each call reduces capacity by 1 in the worst case).

Worst-Case Time Complexity:**O(n^C), which is exponential**.

Space Complexity

Recursive Call Stack: The depth of the recursive calls depends on the capacity C.

worst case: each call reduces the capacity by the smallest weight, leading to a maximum depth of C.

-> O(C)

Code for running the DP algo and printing part 4 results

```python
def max_profit(C, weights, profits):
    n = len(weights)
    # Initialize a 2D DP array where rows are items and columns are capacities
    dp = [[0] * (C + 1) for _ in range(n)]

    # Fill the DP table
    for capacity in range(1, C + 1):
        for i in range(n):
            # Option 1: Exclude the item
            dp[i][capacity] = dp[i - 1][capacity] if i > 0 else 0

            # Option 2: Include the item (if capacity allows)
            if weights[i] <= capacity:
                dp[i][capacity] = max(dp[i][capacity], dp[i][capacity - weights[i]] + profits[i])

    # Print the DP table with labels
    print("DP Table (Items x Capacity):")
    print("      ", " ".join([f"C={j:2}" for j in range(C + 1)]))
    print("     ", "-" * (5 * (C + 1)))

    for i in range(n):
        print(f"Item {i} |", " ".join(f"{dp[i][capacity]:4}" for capacity in range(C + 1)))

    # The maximum profit with the given capacity and all items is in dp[n-1][C]
    return dp[n - 1][C]

# Test with the data
C = 14
weights = [5, 6, 8]
profits = [7, 6, 9]
max_profit = max_profit(C, weights, profits)
print("\nMaximum profit for P(14) with weights [5, 6, 8] and profits [7, 6, 9] is:", max_profit)
```

04
Results using Dynamic Programming

4(a) Results of P(14) with

	0	1	2
w_i	4	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ---------------------------------------------------------------------------
Item 0 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 1 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 2 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21

Maximum profit for P(14) with weights [4, 6, 8] and profits [7, 6, 9] is: 21
```

Maximum Profit = 21

4(a) Results of P(14) with

	0	1	2
w_i	4	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------------
Item 0 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 1 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 2 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21

Maximum profit for P(14) with weights [4, 6, 8] and profits [7, 6, 9] is: 21
```

Maximum Profit = 21

at C=4, item 0
will be placed in the bag
bag = [0]

4(a) Results of P(14) with

	0	1	2
w_i	4	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------------
Item 0 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 1 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 2 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21

Maximum profit for P(14) with weights [4, 6, 8] and profits [7, 6, 9] is: 21
```

Maximum Profit = 21

at C=8, another item 0
will be placed in the bag
bag = [0,0]

4(a) Results of P(14) with

	0	1	2
w_i	4	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------------
Item 0 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 1 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 2 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21

Maximum profit for P(14) with weights [4, 6, 8] and profits [7, 6, 9] is: 21
```

Maximum Profit = 21

at C=10,
if we consider including item 1

bag = [0,0]: profit = 7+7=14
bag = [0,1]: profit = 7+6=13<14

4(a) Results of P(14) with

	0	1	2
w_i	4	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ---------------------------------------------------------------------------
Item 0 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 1 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 2 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21

Maximum profit for P(14) with weights [4, 6, 8] and profits [7, 6, 9] is: 21
```

Maximum Profit = 21

at C=12, another item 0
will be placed in the bag
bag = [0,0, 0]

4(a) Results of P(14) with

	0	1	2
w_i	4	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------------
Item 0 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 1 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 2 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21

Maximum profit for P(14) with weights [4, 6, 8] and profits [7, 6, 9] is: 21
```

Maximum Profit = 21

at C=12,
if we consider including item 2

bag = [0,0, 0]: profit = 7+7+7=21
bag = [0,2]: profit = 7+9=15<21

4(a) Results of P(14) with

	0	1	2
w_i	4	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ---------------------------------------------------------------------------
Item 0 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 1 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 2 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21

Maximum profit for P(14) with weights [4, 6, 8] and profits [7, 6, 9] is: 21
```

Maximum Profit = 21

4(b) Results of P(14) with

	0	1	2
w_i	5	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------------
Item 0 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 1 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 2 |    0    0    0    0    0    7    7    7    9    9   14   14   14   16   16

Maximum profit for P(14) with weights [5, 6, 8] and profits [7, 6, 9] is: 16
```

Maximum Profit = 16

4(b)Results of P(14) with

	0	1	2
w_i	5	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------------
Item 0 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 1 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 2 |    0    0    0    0    0    7    7    7    9    9   14   14   14   16   16

Maximum profit for P(14) with weights [5, 6, 8] and profits [7, 6, 9] is: 16
```

Maximum Profit = 16

at C=5, item 0
will be placed in the bag
bag = [0]

4(b) Results of P(14) with

	0	1	2
w_i	5	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------------
Item 0 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 1 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 2 |    0    0    0    0    0    7    7    7    9    9   14   14   14   16   16

Maximum profit for P(14) with weights [5, 6, 8] and profits [7, 6, 9] is: 16
```

at C=8, only item 0
is in the bag
bag = [0]

Maximum Profit = 16

at C=8,
consider up till item 2

bag = [0] --> 7
bag = [2] --> 9>7

4(b) Results of P(14) with

	0	1	2
w_i	5	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------------
Item 0 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 1 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 2 |    0    0    0    0    0    7    7    7    9    9   14   14   14   16   16

Maximum profit for P(14) with weights [5, 6, 8] and profits [7, 6, 9] is: 16
```

Maximum Profit = 16

at C=10, another item 0
is added to the bag
bag = [0, 0] --> 7+7=14

at C=10,
consider **up till item 2**

bag = [0, 0] --> 7+7=14
bag = [2] --> 9<14

4(b) Results of P(14) with

	0	1	2
w_i	5	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------------
Item 0 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 1 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 2 |    0    0    0    0    0    7    7    7    9    9   14   14   14   16   16

Maximum profit for P(14) with weights [5, 6, 8] and profits [7, 6, 9] is: 16
```

Maximum Profit = 16

at C=11,
consider **item 1**

bag = [0, 0] --> 7+7=14

bag = [0,1] --> 7 + 6 = 13<14

4(b) Results of P(14) with

	0	1	2
w_i	5	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ---------------------------------------------------------------------------
Item 0 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 1 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 2 |    0    0    0    0    0    7    7    7    9    9   14   14   14   16   16

Maximum profit for P(14) with weights [5, 6, 8] and profits [7, 6, 9] is: 16
```

Maximum Profit = 16

at C=13, another item 0
is added to the bag
bag = [0, 0] --> 7+7=14

at C=13,
consider **up till item 2**

bag = [0, 0] --> 7+7=14
bag = [2, 0] --> 9 + 7 = 16 > 14

4(b)Results of P(14) with

	0	1	2
w_i	5	6	8
p_i	7	6	9

```
DP Table (Items x Capacity):
        C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------------
Item 0 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 1 |    0    0    0    0    0    7    7    7    7    7   14   14   14   14   14
Item 2 |    0    0    0    0    0    7    7    7    9    9   14   14   14   16   16

Maximum profit for P(14) with weights [5, 6, 8] and profits [7, 6, 9] is: 16
```

Maximum Profit = 16

Conclusion

Dynamic Programming improved the overall efficiency

time complexity (DP): O(Cn) vs time complexity(recursive function): O(n^C)

space complexity (DP): O(Cn) using 2D array

to further improve the programming,

we can implement the 1D array, we can improve the **space complexity to O(C)** but it decreases the readability.

using 2D array

```
DP Table (Items x Capacity):
       C= 0 C= 1 C= 2 C= 3 C= 4 C= 5 C= 6 C= 7 C= 8 C= 9 C=10 C=11 C=12 C=13 C=14
      ------------------------------------------------------------------------
Item 0 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 1 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21
Item 2 |    0    0    0    0    7    7    7    7   14   14   14   14   21   21   21

Maximum profit for P(14) with weights [4, 6, 8] and profits [7, 6, 9] is: 21
```

using 1D array

```
(base) gaoxinyue@Gaos-MacBook-Air Desktop % python unbounded_knapsack__Capacities.py
Capacity |   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
-------------------------------------------------------------------
C= 1     |   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
C= 2     |   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
C= 3     |   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
C= 4     |   0   0   0   0   7   0   0   0   0   0   0   0   0   0   0
C= 5     |   0   0   0   0   7   7   0   0   0   0   0   0   0   0   0
C= 6     |   0   0   0   0   7   7   7   0   0   0   0   0   0   0   0
C= 7     |   0   0   0   0   7   7   7   7   0   0   0   0   0   0   0
C= 8     |   0   0   0   0   7   7   7   7  14   0   0   0   0   0   0
C= 9     |   0   0   0   0   7   7   7   7  14  14   0   0   0   0   0
C=10     |   0   0   0   0   7   7   7   7  14  14  14   0   0   0   0
C=11     |   0   0   0   0   7   7   7   7  14  14  14  14   0   0   0
C=12     |   0   0   0   0   7   7   7   7  14  14  14  14  21   0   0
C=13     |   0   0   0   0   7   7   7   7  14  14  14  14  21  21   0
C=14     |   0   0   0   0   7   7   7   7  14  14  14  14  21  21  21

Final DP Array: [0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 21]

Maximum profit for P(14) with weights [4, 6, 8] and profits [7, 6, 9] is: 21
```

THANK YOU