Example Project 3
Group 1

Lim Li Ping Joey, U2321331C
Vinodhithaa, U2322818J
Murugan Rithika U2323063E

Problem definition

- **Unbounded Knapsack Problem:**
  - Given:
    - Knapsack capacity C(positive integer).
    - n types of objects, each with:
      - **Weight:** w_i
      - **Profit:** p_i
  - Each object can be added unlimited times.
  - **Goal:** Maximize the total profit in the knapsack without exceeding its capacity.

Recursive Formula for Maximum Profit P(C)

Definition of P(C):

- P(C) is the maximum profit achievable with knapsack capacity C.

Recursive Formula:

$$P(C)= \max_{0\le i\le n} \{P(C-w_i)+p_i\} \text{ if } C>0$$

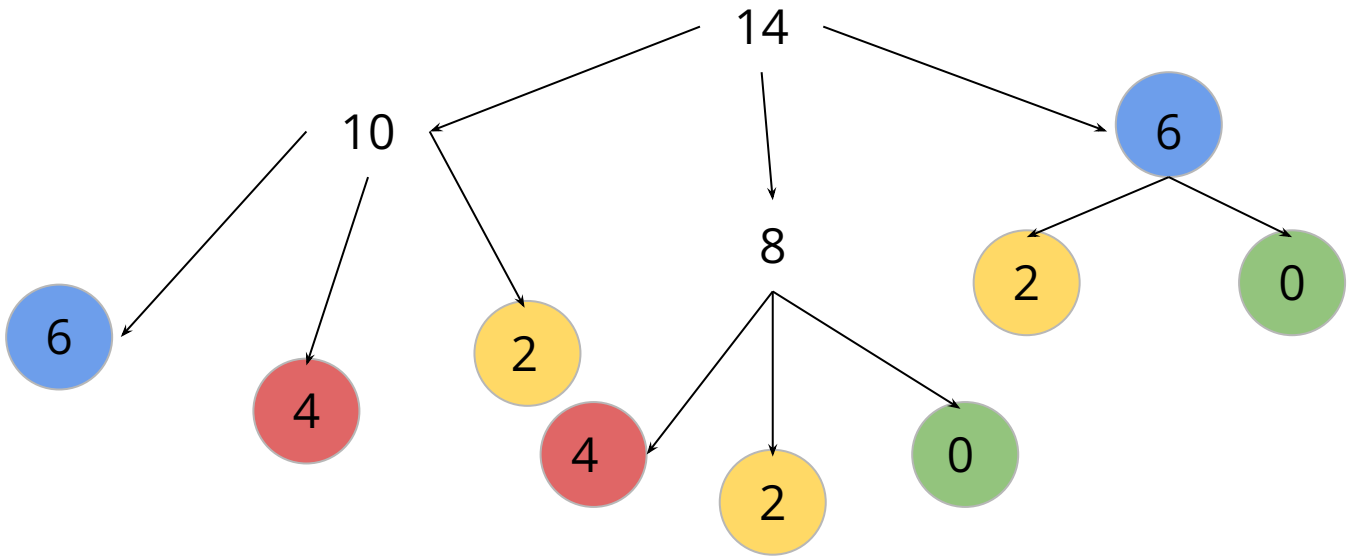
represents the maximum profit achievable with a reduced capacity $C-w_i$

To find the maximum profit achievable by choosing values for i

Base Case: When C=0 then P(C)=0

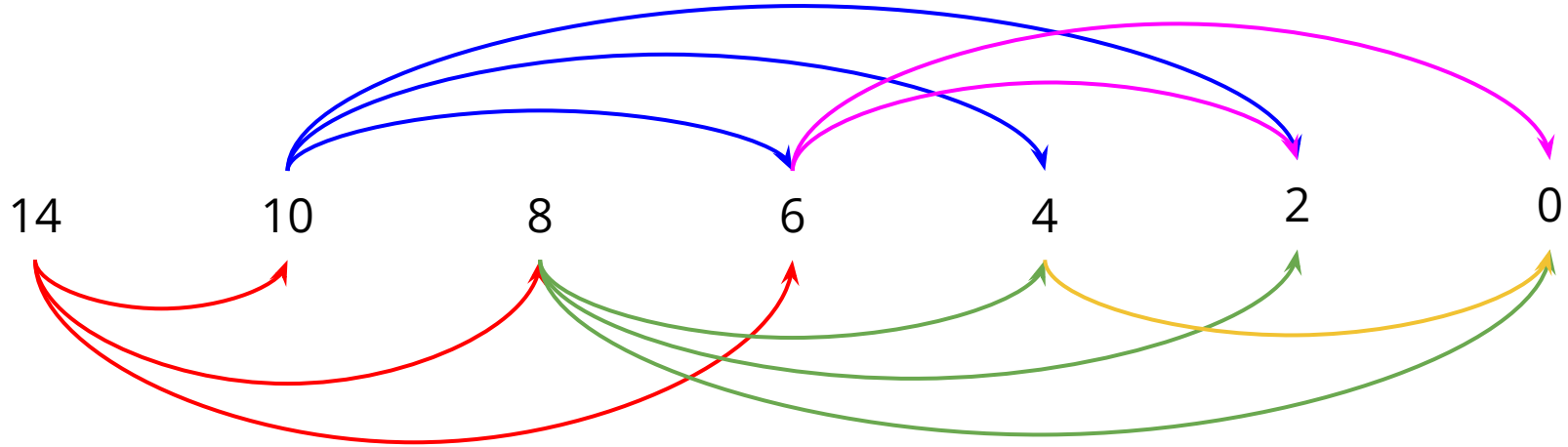
p_i is the profit of including one object of type i in the knapsack.

Recurrence Tree



	0	1	2
w_i	4	6	8
p_i	7	6	9

Subproblem Graph



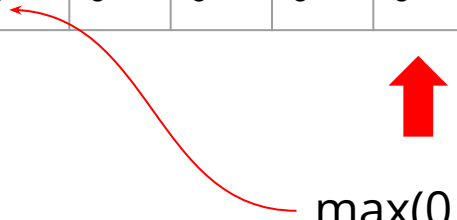
	0	1	2
w_i	4	6	8
p_i	7	6	9

Base Case:
When C=0, P(C)=0

DP Algorithm

	0	1	2
w_i	4	6	8
p_i	7	6	9

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DP Algorithm

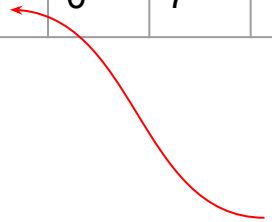
	0	1	2
w_i	4	6	8
p_i	7	6	9

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	7	0	0	0	0	0	0	0	0	0	0

DP Algorithm

	0	1	2
w_i	4	6	8
p_i	7	6	9

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	7	7	0	0	0	0	0	0	0	0	0

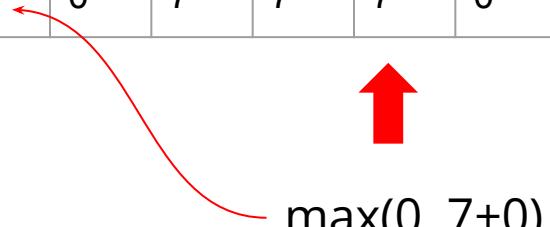


max(0, 7+0)

DP Algorithm

	0	1	2
w_i	4	6	8
p_i	7	6	9

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	7	7	7	0	0	0	0	0	0	0	0



DP Algorithm

	0	1	2
w_i	4	6	8
p_i	7	6	9

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	7	7	7	0	0	0	0	0	0	0	0

DP Algorithm

```python
def unbounded_knapsack_1d(C, weights, values):
    # init DP table
    profits = [0] * (C+1)

    for capacity in range(1, C+1):
        print("For Capacity " + str(capacity))
        print(str(profits) + " [Initial]")
        for i in range(len(weights)):
            # for each capacity, take the max possible out of all values
            if weights[i] <= capacity:
                # dont choose
                exclude = profits[capacity]
                # choose
                include = profits[capacity - weights[i]] + values[i]
                profits[capacity] = max(include, exclude)
                print(str(profits), end="")
                print(f" W[{i}]: {weights[i]}, P[{i}]: {values[i]}")
        print()
    return profits[C]
```

Results

	0	1	2
w_i	4	6	8
p_i	7	6	9

P(14)

```
w = [4,6,8]
p = [7,6,9]
C = 14
n = len(w)
answer = unbounded_knapsack_1d(C, w, p)
print(answer)
```

```
For Capacity 1
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]

For Capacity 2
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]

For Capacity 3
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]
```

Results

	0	1	2
w_i	4	6	8
p_i	7	6	9

```
w = [4,6,8]
p = [7,6,9]
C = 14
n = len(w)
answer = unbounded_knapsack_1d(C, w, p)
print(answer)
```

max(0, 7+7)
max(14, 6+0)
max(14, 9+0)

```
For Capacity 4
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0] W[0]: 4, P[0]: 7

For Capacity 5
[0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 0] W[0]: 4, P[0]: 7

For Capacity 6
[0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0] W[1]: 6, P[1]: 6

For Capacity 7
[0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 0, 0, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 0, 0, 0, 0, 0] W[1]: 6, P[1]: 6

For Capacity 8
[0, 0, 0, 0, 7, 7, 7, 7, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 0, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 0, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 0, 0, 0, 0] W[2]: 8, P[2]: 9

For Capacity 9
[0, 0, 0, 0, 7, 7, 7, 7, 14, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 0, 0, 0] W[2]: 8, P[2]: 9
```

```
For Capacity 10
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] W[2]: 8, P[2]: 9

For Capacity 11
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 0, 0] W[2]: 8, P[2]: 9

For Capacity 12
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 0] W[2]: 8, P[2]: 9

For Capacity 13
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21] W[2]: 8, P[2]: 9

For Capacity 14
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 21] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 21] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 21] W[2]: 8, P[2]: 9

21
```

Results

	0	1	2
w_i	4	6	8
p_i	7	6	9

```
w = [4,6,8]
p = [7,6,9]
C = 14
n = len(w)
answer = unbounded_knapsack_1d(C, w, p)
print(answer)
```

max(0,7+14)
max(21,6+14)
max(21,9+7)

Solution:
{0,0,0},
p=21

```
For Capacity 4
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0] W[0]: 4, P[0]: 7

For Capacity 5
[0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 0] W[0]: 4, P[0]: 7

For Capacity 6
[0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0] W[1]: 6, P[1]: 6

For Capacity 7
[0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 0, 0, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 0, 0, 0, 0, 0] W[1]: 6, P[1]: 6

For Capacity 8
[0, 0, 0, 0, 7, 7, 7, 7, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 0, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 0, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 0, 0, 0, 0] W[2]: 8, P[2]: 9

For Capacity 9
[0, 0, 0, 0, 7, 7, 7, 7, 14, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 0, 0, 0] W[2]: 8, P[2]: 9
```

```
For Capacity 10
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] W[2]: 8, P[2]: 9

For Capacity 11
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] W[2]: 8, P[2]: 9

For Capacity 12
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 0, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 21, 0, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 21, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 21, 0, 0] W[2]: 8, P[2]: 9

For Capacity 13
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 0, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 0] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 0] W[2]: 8, P[2]: 9

For Capacity 14
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 0] [Initial]
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 21] W[0]: 4, P[0]: 7
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 21] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 7, 7, 7, 7, 14, 14, 14, 14, 21, 21, 21] W[2]: 8, P[2]: 9

21
```

Results

	0	1	2
w_i	5	6	8
p_i	7	6	9

P(14)

```
w = [5,6,8]
p = [7,6,9]
C = 14
n = len(w)
answer = unbounded_knapsack_1d(C, w, p)
print(answer)
```

```
For Capacity 1
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]

For Capacity 2
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]

For Capacity 3
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]

For Capacity 4
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]
```

Results

	0	1	2
w_i	5	6	8
p_i	7	6	9

```
w = [5,6,8]
p = [7,6,9]
C = 14
n = len(w)
answer = unbounded_knapsack_1d(C, w, p)
print(answer)
```

Solution: {0,2}, p=16

```
For Capacity 5
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0] W[0]: 5, P[0]: 7

For Capacity 6
[0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 0] W[0]: 5, P[0]: 7
[0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 0] W[1]: 6, P[1]: 6

For Capacity 7
[0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0] W[0]: 5, P[0]: 7
[0, 0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0] W[1]: 6, P[1]: 6

For Capacity 8
[0, 0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 0, 7, 7, 7, 7, 0, 0, 0, 0, 0] W[0]: 5, P[0]: 7
[0, 0, 0, 0, 0, 7, 7, 7, 7, 0, 0, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 0, 7, 7, 7, 9, 0, 0, 0, 0, 0] W[2]: 8, P[2]: 9

For Capacity 9
[0, 0, 0, 0, 0, 7, 7, 7, 9, 0, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 0, 7, 7, 7, 9, 7, 0, 0, 0, 0] W[0]: 5, P[0]: 7
[0, 0, 0, 0, 0, 7, 7, 7, 9, 7, 0, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 0, 0, 0, 0] W[2]: 8, P[2]: 9

For Capacity 10
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 0, 0, 0] W[0]: 5, P[0]: 7
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 0, 0, 0] W[2]: 8, P[2]: 9
```

```
For Capacity 11
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 0, 0, 0, 0] [Initial]
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 0, 0, 0] W[0]: 5, P[0]: 7
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 0, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 0, 0, 0] W[2]: 8, P[2]: 9

For Capacity 12
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 0, 0, 0] [Initial]
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 0, 0] W[0]: 5, P[0]: 7
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 0, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 0, 0] W[2]: 8, P[2]: 9

For Capacity 13
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 0, 0] [Initial]
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 16, 0] W[0]: 5, P[0]: 7
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 16, 0] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 16, 0] W[2]: 8, P[2]: 9

For Capacity 14
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 16, 0] [Initial]
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 16, 16] W[0]: 5, P[0]: 7
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 16, 16] W[1]: 6, P[1]: 6
[0, 0, 0, 0, 0, 7, 7, 7, 9, 9, 14, 14, 14, 16, 16] W[2]: 8, P[2]: 9

16
```

Summary

0/1 Knapsack	**Unbounded Knapsack**
- Cannot repeat items	- Can repeat items
- More intuitively represented as 2D DP Table, because we need to keep track of which items has already been used	- More intuitively represented in 1D DP Table, because we don't need to keep track of which item has been used
$P(C, j) = \max(P(C, j-1), p_j + P(C-w_j, j-1))$	$P(C)= \max_{0\le i\le n} \{P(C-w_i)+p_i\}$ if $C>0$

Thank
you!