DAA PRESENTATION

KNAPSACK PROBLEM

Shreyash, Sonali, Kimaya, Sneha, Sanjali

WHAT WILL BE COVERED ?

1. What is Dynamic Programming
2. Knapsack Problem
3. Approaches to Knapsack problem
4. 0/1 Knapsack Problem
5. Example 1
6. Example 2
7. Conclusion ---> Comparison

CONCEPT OF DYNAMIC PROGRAMMING

- Dynamic programming to solve optimisation problems
- Two main properties of a problem suggest that the given problem can be solved using Dynamic Programming.
- These properties are overlapping sub-problems and optimal substructure.
- Dynamic Programming algorithm solves each sub-problem just once and then saves its answer in a table, thereby avoiding the work of re-computing the answer every time

KNAPSACK PROBLEM

- The knapsack problem is an optimization problem used to illustrate both problem and solution.
- It derives its name from a scenario where one is constrained in the number of items that can be placed inside a fixed-size knapsack.
- Given a set of items with specific weights and values, the aim is to get as much value into the knapsack as possible given the weight constraint of the knapsack.
- The problem can be tackled using various approaches: brute force, top-down with memoization and bottom-up are all potentially viable approaches to take.
- The latter two approaches (top-down with memoization and bottom-up) make use of Dynamic Programming

APPROACHES TO KNAPSACK PROBLEM

There are two ways to approach Knapsack problem:

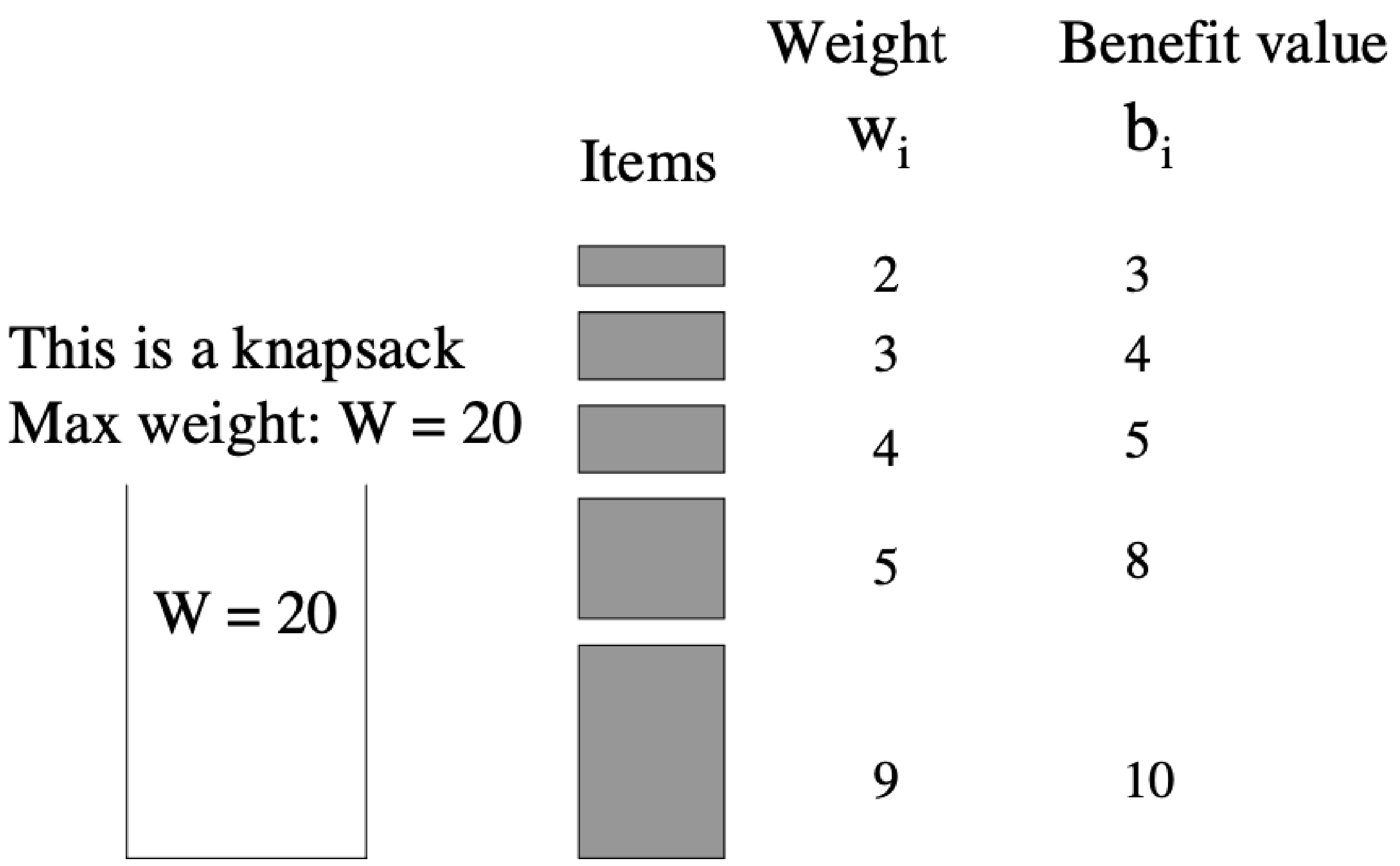
1. Dynamic Approach(0/1 Knapsack): In this case, items are not divisible, i.e., you either take an item or not.
2. Greedy Approach (Fractional Knapsack): In this case, you can take any fraction of an item.

0/1 KNAPSACK PROBLEM

1. Given a knapsack with maximum capacity W, and a set S consisting of n items
2. Each item i has some weight wi and benefit value bi (all wi , bi and W are integer values)

Problem: How to pack the knapsack to achieve maximum total value of packed items?

0-1 Knapsack problem: a picture



- Problem, in other words, is to find

$$\max \sum_{i \in T} b_i \text{ subject to } \sum_{i \in T} w_i \leq W$$

- The problem is called a "*0-1*" problem, because each item must be entirely accepted or rejected.
- In the "*Fractional Knapsack Problem*," we can take fractions of items.

COMPARISON BETWEEN THE APPROACHES OF KNAPSACK PROBLEM

1.In the dynamic approach, you can either take the object or not. Whereas in greedy approach, you can take fraction of every object.

2. In dynamic approach, in each progression we always choose an object which is optimal and in greedy approach we calculate the ration value/weight of each item and sort them in descending order to choose the items.

3. The dynamic approach is slower than the greedy approach.

SOLVED EXAMPLE 1

Recursive Formula

$$B[k,w] = \begin{cases} B[k-1,w] & \text{if } w_k > w \\ \max\{B[k-1,w],\ B[k-1,w-w_k]+b_k\} & \text{else} \end{cases}$$

- The best subset of S_k that has the total weight w, either contains item k or not.
- First case: $w_k > w$. Item k can't be part of the solution, since if it was, the total weight would be $> w$, which is unacceptable.
- Second case: $w_k \le w$. Then the item k can be in the solution, and we choose the case with greater value.

Example

Let's run our algorithm on the following data:

n = 4 (# of elements)
W = 5 (max weight)
Elements (weight, benefit):
(2,3), (3,4), (4,5), (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1						
2						
3						
4						

for w = 0 to W
    B[0,w] = 0

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

for i = 1 to n
    B[i,0] = 0

Example (4)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0				
2	0					
3	0					
4	0					

i=1
b_i=3
w_i=2
w=1
w-w_i =-1

if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else **B[i,w] = B[i-1,w]** // w_i > w

Example (5)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	**3**			
2	0					
3	0					
4	0					

i=1
b_i=3
w_i=2
w=2
w-w_i =0

if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w] // w_i > w

Example (6)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	**3**		
2	0					
3	0					
4	0					

i=1
b_i=3
w_i=2
w=3
w-w_i =1

if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w] // w_i > w

Example (7)

Items:
- 1: (2,3)
- 2: (3,4)
- 3: (4,5)
- 4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	**3**	
2	0					
3	0					
4	0					

$i=1$, $b_i=3$, $w_i=2$, $w=4$, $w-w_i=2$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```


Example (8)

Items:
- 1: (2,3)
- 2: (3,4)
- 3: (4,5)
- 4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	**3**
2	0					
3	0					
4	0					

$i=1$, $b_i=3$, $w_i=2$, $w=5$, $w-w_i=3$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```

Example (9)

Items:
- 1: (2,3)
- 2: (3,4)
- 3: (4,5)
- 4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	**0**				
3	0					
4	0					

$i=2$, $b_i=4$, $w_i=3$, $w=1$, $w-w_i=-2$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```


Example (10)

Items:
- 1: (2,3)
- 2: (3,4)
- 3: (4,5)
- 4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	**3**			
3	0					
4	0					

$i=2$, $b_i=4$, $w_i=3$, $w=2$, $w-w_i=-1$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```

Example (11)

Items:
- 1: (2,3)
- 2: (3,4)
- 3: (4,5)
- 4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	**4**		
3	0					
4	0					

$i=2$, $b_i=4$, $w_i=3$, $w=3$, $w-w_i=0$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```


Example (12)

Items:
- 1: (2,3)
- 2: (3,4)
- 3: (4,5)
- 4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	**4**	
3	0					
4	0					

$i=2$, $b_i=4$, $w_i=3$, $w=4$, $w-w_i=1$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```

Example (13)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	**7**
3	0					
4	0					

$i=2$
$b_i=4$
$w_i=3$
$w=5$
$w-w_i=2$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```


Example (14)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	**0**	**3**	**4**		
4	0					

$i=3$
$b_i=5$
$w_i=4$
$w= 1..3$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```

Example (15)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	**5**	
4	0					

$i=3$
$b_i=5$
$w_i=4$
$w= 4$
$w- w_i=0$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```


Example (16)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	**7**
4	0					

$i=3$
$b_i=5$
$w_i=4$
$w= 5$
$w- w_i=1$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```


Example (17)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	**0**	**3**	**4**	**5**	

$i=4$
$b_i=6$
$w_i=5$
$w= 1..4$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```

Example (18)

Items:
1: (2,3
2: (3,4
3: (4,5
4: (5,6

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	**7**

$i=4$
$b_i=6$
$w_i=5$
$w= 5$
$w- w_i=0$

```
if w_i <= w // item i can be part of the solution
    if b_i + B[i-1,w-w_i] > B[i-1,w]
        B[i,w] = b_i + B[i-1,w- w_i]
    else
        B[i,w] = B[i-1,w]
else B[i,w] = B[i-1,w]  // w_i > w
```

Finding the Items (2)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

$i=4$
$k=5$
$b_i=6$
$w_i=5$
$B[i,k] = 7$
$B[i-1,k] = 7$

i=n, k=W
while i,k > 0
  if $B[i,k] \neq B[i-1,k]$ then
    mark the i^{th} item as in the knapsack
    $i = i-1$, $k = k-w_i$
  else
    $i = i-1$

Finding the Items (3)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

$i=3$
$k=5$
$b_i=6$
$w_i=4$
$B[i,k] = 7$
$B[i-1,k] = 7$

i=n, k=W
while i,k > 0
  if $B[i,k] \neq B[i-1,k]$ then
    mark the i^{th} item as in the knapsack
    $i = i-1$, $k = k-w_i$
  else
    $i = i-1$

Finding the Items (4)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

$i=2$
$k=5$
$b_i=4$
$w_i=3$
$B[i,k] = 7$
$B[i-1,k] = 3$
$k - w_i = 2$

i=n, k=W
while i,k > 0
  if $B[i,k] \neq B[i-1,k]$ then
    mark the i^{th} item as in the knapsack
    $i = i-1$, $k = k-w_i$
  else
    $i = i-1$

Finding the Items (5)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

$i=1$
$k=2$
$b_i=3$
$w_i=2$
$B[i,k] = 3$
$B[i-1,k] = 0$
$k - w_i = 0$

i=n, k=W
while i,k > 0
  if $B[i,k] \neq B[i-1,k]$ then
    mark the i^{th} item as in the knapsack
    $i = i-1$, $k = k-w_i$
  else
    $i = i-1$

Finding the Items (6)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

$i=0$
$k=0$

The optimal knapsack should conta
{1, 2}

i=n, k=W
while i,k > 0
  if $B[i,k] \neq B[i-1,k]$ then
    mark the n^{th} item as in the knapsack
    $i = i-1$, $k = k-w_i$
  else
    $i = i-1$

Finding the Items (7)

Items:
1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

The optimal knapsack should contai
{1, 2}

i=n, k=W
while i,k > 0
  if $B[i,k] \neq B[i-1,k]$ then
    mark the n^{th} item as in the knapsack
    $i = i-1$, $k = k-w_i$
  else
    $i = i-1$

EXAMPLE 2

Given:

Weights = {3,4,5,6}
Profits = {2,3,4,1}
Total weight of bag = 8 kg

$$B[k,w] = \begin{cases} B[k-1,w] & \text{if } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k]+b_k\} & \text{else} \end{cases}$$

Where:

B: Matrix
k: No. of items
w: Total weight
w_k: weight of k_{th} item
b_k: benefit value of k_{th} item/given profit of k_{th} item

b_k	W_k	k \ w	0	1	2	3	4	5	6	7	8
		0									
2	3	1									
3	4	2									
4	5	3									
1	6	4									

b_k	W_k	k \ w	0	1	2	3	4	5	6	7	8
		0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
2	3	1	**0**								
3	4	2	**0**								
4	5	3	**0**								
1	6	4	**0**								

b_k	W_k	k \ w	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
2	3	1	0	**0**	**0**						
3	4	2	0								
4	5	3	0								
1	6	4	0								

		k \ w	0	1	2	3	4	5	6	7	8
b_k	W_k	0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	**2**					
3	4	2	0								
4	5	3	0								
1	6	4	0								

Since w_k<=w,

Formula: $B[k,w] = \max\{B[k-1, w], (B[k-1, w-w_k]+b_k)\}$

Here, k=1,w=3, w_k=3, b_k=2

$$
\begin{aligned}
B[1,3] &= \max\{B[1-1, 3], (B[1-1, 3-3]+2)\}\\
&=\max\{B[0,3], (B[0,0] + 2)\}\\
&=\max\{0, (0+2)\}\\
&=\max\{0,2\}\\
&=2
\end{aligned}
$$

		k \ w	0	1	2	3	4	5	6	7	8
b_k	W_k	0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	**2**				
3	4	2	0								
4	5	3	0								
1	6	4	0								

Since w_k<=w,

Formula: $B[k,w] = \max\{B[k-1, w], (B[k-1, w-w_k]+b_k)\}$

Here, k=1,w=4, w_k=3, b_k=2

$$
\begin{aligned}
B[1,4] &= \max\{B[1-1, 4], (B[1-1, 4-3]+2)\}\\
&=\max\{B[0,4], (B[0,1] + 2)\}\\
&=\max\{0, (0+2)\}\\
&=\max\{0,2\}\\
&=2
\end{aligned}
$$

		k \ w	0	1	2	3	4	5	6	7	8
b_k	W_k	**0**	0	0	0	0	0	0	0	0	0
2	3	**1**	0	0	0	2	**2**	**2**	**2**	**2**	**2**
3	4	**2**	0								
4	5	**3**	0								
1	6	**4**	0								

		k \ w	0	1	2	3	4	5	6	7	8
b_k	W_k	**0**	0	0	0	0	0	0	0	0	0
2	3	**1**	0	0	0	2	2	2	2	2	2
3	4	**2**	0	**0**							
4	5	**3**	0								
1	6	**4**	0								

Since $w_k > w$,

Formula: $B[k,w] = B[k-1\ ,\ w]$

Here, $k=2, w=1, w_k=4$

$B[2,1] = B[2-1,1]$

$= B[1,1]$

$= 0$

		w → / k ↓	0	1	2	3	4	5	6	7	8
b_k	W_k	**0**	0	0	0	0	0	0	0	0	0
2	3	**1**	0	0	0	2	2	2	2	2	2
3	4	**2**	0	**0**	**0**	**2**					
4	5	**3**	0								
1	6	**4**	0								

Since w_k>w,

Formula: B[k,w] = B[k-1 , w]

Here, k=2,w=2,w_k=4

B[2,2] = B[2-1,2]
=B[1,2]
=0

Since w_k>w,

Formula: B[k,w] = B[k-1 , w]

Here, k=2,w=3,w_k=4

B[2,3] = B[2-1,3]
=B[1,3]
=2

		w → / k ↓	0	1	2	3	4	5	6	7	8
b_k	W_k	**0**	0	0	0	0	0	0	0	0	0
2	3	**1**	0	0	0	2	2	2	2	2	2
3	4	**2**	0	0	0	2	**3**				
4	5	**3**	0								
1	6	**4**	0								

Since w_k<=w,

Formula: B[k,w] = max{B[k-1 , w], (B[k-1 , w-w_k]+b_k)}

Here, k=2,w=4, w_k=4, b_k=3

B[2,4] = max{B[2-1 , 4], (B[2-1 , 4-4]+3)}
=max{B[1,4] , (B[1,0] + 3)}
=max{2, (0+3)}
=max{2,3}
=3

		k ↓ w →	0	1	2	3	4	5	6	7	8
b_k	W_k	**0**	0	0	0	0	0	0	0	0	0
2	3	**1**	0	0	0	2	2	2	2	2	2
3	4	**2**	0	0	0	2	3	**3**	**3**	**5**	**5**
4	5	**3**	0								
1	6	**4**	0								

Since $w_k<=w$,

Formula: $B[k,w] = \max\{B[k-1 , w], (B[k-1 , w-w_k]+b_k)\}$

Here, $k=2, w=7, w_k=4, b_k=3$

$$
\begin{aligned}
B[2,7] &= \max\{B[2-1 , 7], (B[2-1 , 7-4]+3)\} \\
&=\max\{B[1,7] , (B[1,3] + 3)\} \\
&=\max\{2, (2+3)\} \\
&=\max\{2,5\} \\
&=5
\end{aligned}
$$

		k ↓ w →	0	1	2	3	4	5	6	7	8
b_k	W_k	**0**	0	0	0	0	0	0	0	0	0
2	3	**1**	0	0	0	2	2	2	2	2	2
3	4	**2**	0	0	0	2	3	3	3	5	5
4	5	**3**	0	**0**	**0**	**2**	**3**				
1	6	**4**	0								

Since $w_k>w$,

Formula: $B[k,w] = B[k-1 , w]$

Here, $k=3, w=1, w_k=5$

$$
\begin{aligned}
B[3,1] &= B[3-1,1] \\
&=B[2,1] \\
&=0
\end{aligned}
$$

bk	Wk	k ↓ \ w →	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	0	2	3	**4**			
1	6	4	0								

Since $w_k \le w$,

Formula: $B[k,w] = \max\{B[k-1, w], (B[k-1, w-w_k]+b_k)\}$

Here, $k=3, w=5, w_k=5, b_k=4$

$$
\begin{aligned}
B[3,5] &= \max\{B[3-1, 5], (B[3-1, 5-5]+4)\} \\
&= \max\{B[2,5], (B[2,0] + 4)\} \\
&= \max\{3, (0+4)\} \\
&= \max\{3,4\} \\
&= 4
\end{aligned}
$$

bk	Wk	k ↓ \ w →	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	0	2	3	4	**4**	**5**	**6**
1	6	4	0								

Since $w_k \le w$,

Formula: $B[k,w] = \max\{B[k-1, w], (B[k-1, w-w_k]+b_k)\}$

Here, $k=3, w=6, w_k=5, b_k=4$

$$
\begin{aligned}
B[3,6] &= \max\{B[3-1, 6], (B[3-1, 6-5]+4)\} \\
&= \max\{B[2,6], (B[2,1] + 4)\} \\
&= \max\{3, (0+4)\} \\
&= \max\{3,4\} \\
&= 4
\end{aligned}
$$

		k ↓ w →	0	1	2	3	4	5	6	7	8
b_k	W_k	**0**	0	0	0	0	0	0	0	0	0
2	3	**1**	0	0	0	2	2	2	2	2	2
3	4	**2**	0	0	0	2	3	3	3	5	5
4	5	**3**	0	0	0	2	3	4	4	5	6
1	6	**4**	**0**	**0**	**0**	**2**	**3**	**4**			

		k ↓ w →	0	1	2	3	4	5	6	7	8
b_k	W_k	**0**	0	0	0	0	0	0	0	0	0
2	3	**1**	0	0	0	2	2	2	2	2	2
3	4	**2**	0	0	0	2	3	3	3	5	5
4	5	**3**	0	0	0	2	3	4	4	5	6
1	6	**4**	0	0	0	2	3	4	**4**		

Since $w_k <= w$,

Formula: $B[k,w] = \max\{B[k-1 , w], (B[k-1 , w-w_k]+b_k)\}$

Here, $k=4, w=6, w_k=6, b_k=1$

$B[3,5] = \max\{B[4-1 , 6], (B[4-1 , 6-6]+1)\}$
$=\max\{B[3,6] , (B[3,0] + 1)\}$
$=\max\{4, (0+1)\}$
$=\max\{4,1\}$
$=4$

		k ↓ \ w →	0	1	2	3	4	5	6	7	8
b_k	W_k	0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	0	2	3	4	4	5	6
1	6	4	0	0	0	2	3	4	4	**5**	**6**

		k ↓ \ w →	0	1	2	3	4	5	6	7	8
b_k	W_k	0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	0	2	3	4	4	5	6
1	6	4	0	0	0	2	3	4	4	5	**6**

		k ↓ \ w →	0	1	2	3	4	5	6	7	8
b_k	W_k	0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	0	2	3	4	4	5	6
1	6	4	0	0	0	2	3	4	4	5	6

		k ↓ \ w →	0	1	2	3	4	5	6	7	8
P_k	W_k	0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4 ✓	5	3	0	0	0	2	3	4	4	5	6
1	6	4	0	0	0	2	3	4	4	5	6

Total maximum profit = 6
Profit of 3rd item = 4

Therefore,
Remaining profit = 2

Pk	Wk	k \ w	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	0	2	3	4	4	5	6
1	6	4	0	0	0	2	3	4	4	5	6

Pk	Wk	k \ w	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	0	2	3	4	4	5	6
1	6	4	0	0	0	2	3	4	4	5	6

The maximum profit that we have got is: 6

The items that have been selected are: 1st and 3rd items i.e, items with the weight 3kg and 5kg

Weights = {3,4,5,6}
Items selected = {1,0,1,0}