0-1 Knapsack problem

Knapsack problem

เป็นปัญหาที่กำหนดให้มีวัตถุหลายๆ ชิ้น ต้องการเอาของใส่ถุงเป้โดยให้ผลรวมของราคาของวัตถุมีค่ามากที่สุด

วัตถุแต่ละชิ้นมีน้ำหนักและมีราคาอยู่

ผลรวมของน้ำหนักที่ถุงเป้จะรับได้ไม่เกินค่าบางค่า W

ดังนั้นเราจะต้องพิจารณาน้ำหนักพร้อมกับมูลค่าของสินค้าพร้อมๆกัน

Item#	น้ำหนัก	มูลค่า
1	1	8
2	3	6
3	5	5

Knapsack problem

สำหรับปัญหานี้จะมี 2 version

1. 0-1 knapsack problem
2. Fractional knapsack problem (แบ่งของเป็นชิ้นย่อยๆ ได้)

แบบที่ 1 สิ่งของแบ่งย่อยไม่ได้: ดังนั้นเราสามรถเลือกหยิบหรือไม่หยิบ แก้ด้วย Dynamic programming

แบบที่ 2 สิ่งของแบ่งย่อยได้: เราสามารถเลือกส่วนของสิ่งของมาได้ แก้ด้วย Greedy algorithm (ทำได้ยังไง?)

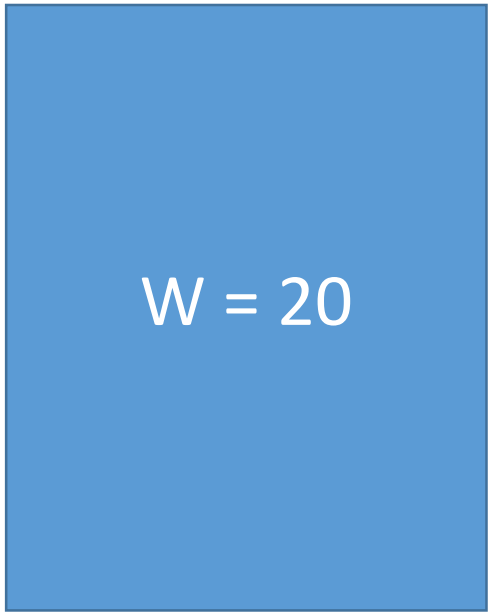
0-1 Knapsack problem

กำหนดให้ ถุงผ้าที่มีความจุ W และเซต S ที่ประกอบด้วยสิ่งของ n ชิ้น สิ่งของแต่ละชิ้น i จะมีน้ำหนัก w_i และมีมูลค่า b_i (กำหนดให้ทุกๆ w_i, b_i และ W เป็นจำนวนเต็ม)

ปัญหา จะบรรจุสิ่งของลงถุงผ้าอย่างไรเพื่อให้สิ่งของที่ถูกบรรจุมีผลรวมมูลค่ามากสุด

0-1 Knapsack problem

knapsack
Max weight: W = 20



Item	Weight w_i	Benefit value b_i
	2	3
	3	4
	4	5
	5	8
	9	10

0-1 Knapsack problem

หากจะแก้ปัญหานี้แบบตรงๆ

เนื่องจากมีสิ่งของ n ชิ้น เพราะฉะนั้นจะมีรูปแบบที่เป็นไปได้ 2^n แบบ

(สิ่งของหนึ่งชิ้นมี 2 ทางเลือกคือ เลือกเอาและเลือกไม่เอา)

เราจะลองทุกๆ รูปแบบนี้และหามูลค่ารวมที่มากที่สุดที่ไม่เกิน W

ดังนั้นใช้เวลาในการทำงาน $O(2^n)$

Defining a Subproblem

จะทำได้ดีกว่านี้ไหม

ได้ ใช้ dynamic programming

ดังนั้นเราต้องระบุ subproblem ให้ได้

ลองระบุด้วยวิธีนี้

ถ้าเราให้ชื่อสิ่งของเป็นหมายเลข 1, 2, …, n แล้ว subproblem จะเป็น

S_k = คำตอบที่ดีที่สุดในการเลือกสิ่งของหมายเลขที่ 1 จนถึงหมายเลขที่ k

Defining a Subproblem

ลองระบุด้วยวิธีนี้

ถ้าเราให้ชื่อสิ่งของเป็นหมายเลข 1, 2, …, n แล้ว subproblem จะเป็น

S_k = คำตอบที่ดีที่สุดในการเลือกสิ่งของหมายเลขที่ 1 จนถึงหมายเลขที่ k

เหมาะสมไหม

คำถาม: เราสามารถอธิบายคำตอบสุดท้าย S_n ได้ด้วย S_k หรือไม่

คำตอบ: ทำไม่ได้

Defining a Subproblem

W_1=2 b_1=3	W_2=4 b_2=5	W_3=5 b_3=8	W_4=3 b_4=4	

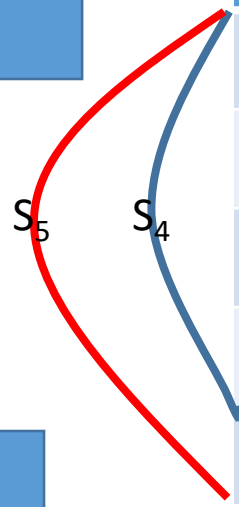
กำหนดให้ Max weight = 20

พิจารณา S_4: น้ำหนักรวม 14, มูลค่ารวม: 20

W_1=2 b_1=3	W_2=4 b_2=5	W_3=5 b_3=8	W_5=9 b_5=10

พิจารณา S_5: น้ำหนักรวม 20, มูลค่ารวม: 26

พบว่าคำตอบของ S_4 ไม่ได้เป็นส่วนหนึ่งของ S5



Item	W_i	B_i
1	2	3
2	3	4
3	4	5
4	5	8
5	9	10

Defining a Subproblem

เราพบว่าคำตอบของ S_4 ไม่ได้เป็นส่วนหนึ่งของคำตอบของ S_5

แสดงว่าการนิยาม S_k ยังเพียงพอ แล้วทำอย่างไรดี

เราพบว่าจริงๆแล้วมี parameter 2 ตัวที่ต้องคำนึงถึง ดังนั้นเราต้องเพิ่ม parameter อีกหนึ่งตัว: w ซึ่งแทนน้ำหนักจริงของแต่ละ subset ของสิ่งของ

นิยามใหม่ subproblem จะเป็นการคำนวณ B[k,w]

Recursive

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

หมายความว่า ซับเซตของสิ่งของที่ดีที่สุดของ S_k ที่มีน้ำหนักรวม w คือ

1. ซับเซตของสิ่งของที่ดีที่สุดของ S_{k-1} ที่มีน้ำหนักรวม w
2. ซับเซตของสิ่งของที่ดีที่สุดของ S_{k-1}ที่มีน้ำหนักรวม w-w_k รวมกับสิ่งของใหม่ k

นั่นคือซับเซตที่ดีที่สุดของ S_k มีน้ำหนักรวม w มีการเลือกหรือไม่เลือกสิ่งของ k

กรณีแรก w_k>w สิ่งของ k ไม่สามารถเป็นส่วนหนึ่งของคำตอบได้

กรณีที่สอง $w_k \leq w$ สิ่งของ k สามารถเป็นส่วนหนึ่งของคำตอบได้และเราจะเลือกกรณีที่ให้ค่ามากที่สุด

Base case

base case

1. หากมีสิ่งของแต่ถุงเป้มีขนาดเป็น 0 แสดงว่าเลือกได้ไหม,มูลค่ารวม=?

2. หากไม่มีของแต่ถุงเป้มีขนาดต่างๆ แสดงว่าเลือกได้ไหม,มูลค่ารวม=?

กรณีแรกคือ

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

กรณีที่สองคือ

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

```
for i=1 to n
      B[i,0]=0
for w=0 to W
      B[0,w]=0
for i=1 to n
      for w=0 to W
            if w_i <= w
                  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
```

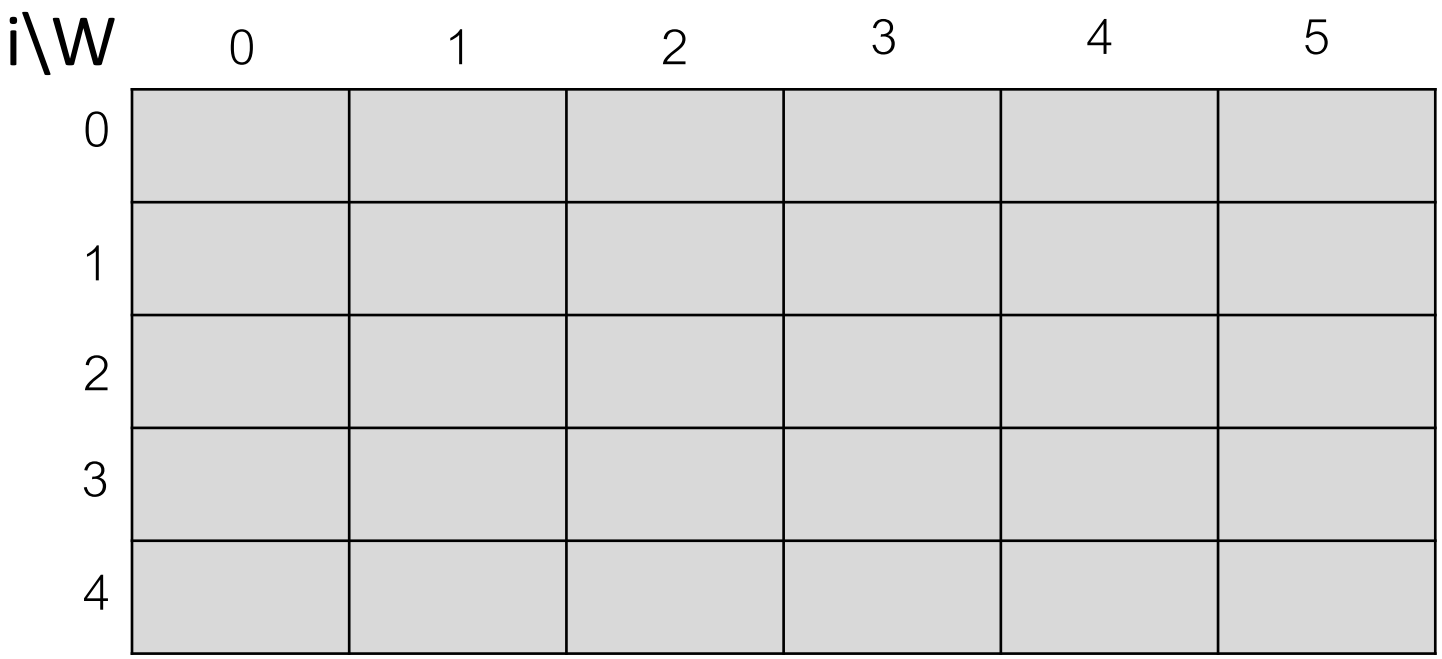
Example

n=4

W=5

สิ่งของ (น้ำหนัก, มูลค่า)

(2,3), (3,4), (4,5), (5,6)


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

Fill basecase

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
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**	**0**				
2	**0**					
3	**0**					
4	**0**					

$i=1$

$b_i=3$

$w_i=2$

$w=1$

$w-w_i=-1$

if $w_i <= w$

if $b_i+B[i-1,w-w_i]>B[i-1,w]$

$B[i,w] = b_i+B[i-1,w-w_i]$

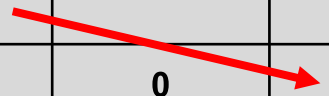
else

$B[i,w] = B[i-1,w]$

else **B[i,w] = B[i-1,w]**

Item(w,b)
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 $w_i <= w$

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 $B[i,w] = B[i-1,w]$

Item(w,b)
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

if b_i+B[i-1,w-w_i]>B[i-1,w]

B[i,w] = b_i+B[i-1,w-w_i]

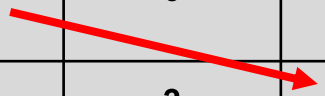
else

B[i,w] = B[i-1,w]

else B[i,w] = B[i-1,w]

Item(w,b)
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 $w_i <= w$

if $b_i+B[i-1,w-w_i]>B[i-1,w]$

$B[i,w] = b_i+B[i-1,w-w_i]$

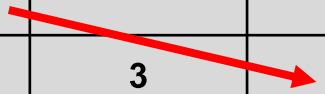
else

$B[i,w] = B[i-1,w]$

else $B[i,w] = B[i-1,w]$

Item(w,b)
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 $w_i <= w$

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 $B[i,w] = B[i-1,w]$

Item(w,b)
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

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]**

Item(w,b)
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 $w_i <= w$

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]**

Item(w,b)
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 $w_i <= w$

if $b_i+B[i-1,w-w_i]>B[i-1,w]$

$B[i,w] = b_i+B[i-1,w-w_i]$

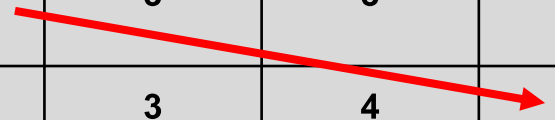
else

$B[i,w] = B[i-1,w]$

else $B[i,w] = B[i-1,w]$

Item(w,b)
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 $w_i <= w$

if $b_i+B[i-1,w-w_i]>B[i-1,w]$

$B[i,w] = b_i+B[i-1,w-w_i]$

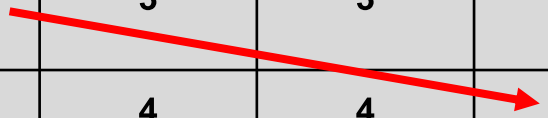
else

$B[i,w] = B[i-1,w]$

else $B[i,w] = B[i-1,w]$

Item(w,b)
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 $w_i <= w$

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 $B[i,w] = B[i-1,w]$

Item(w,b)
1 (2,3)
2 (3,4)
3 (4,5)
4 (5,6)

ลองเติมข้อมูลลงตาราง

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**					

if w_i <= w

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]

Item(w,b)
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

if w_i <= w

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]

Item(w,b)
1 (2,3)
2 (3,4)
3 (4,5)
4 (5,6)

How to find actual knapsack items

ตอนนี้ข้อมูลทุกอย่างอยู่ในตาราง

B[n,W] เป็นค่ามากที่สุดของสิ่งของที่ถูกใส่เข้าไปในถุงเป้

ให้ i=n และ k=W

if B[i,k] != 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 //Assume the ith item is not in the knapsack

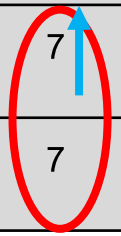
//Could it be in the optimally packed knapsack?

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=n, k=W
while i,k>0
    if B[i,k] != 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
```

Item(w,b)
1 (2,3)
2 (3,4)
3 (4,5)
4 (5,6)

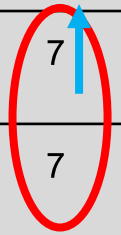
i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7



```
i=n, k=W
while i,k>0
    if B[i,k] != 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
```

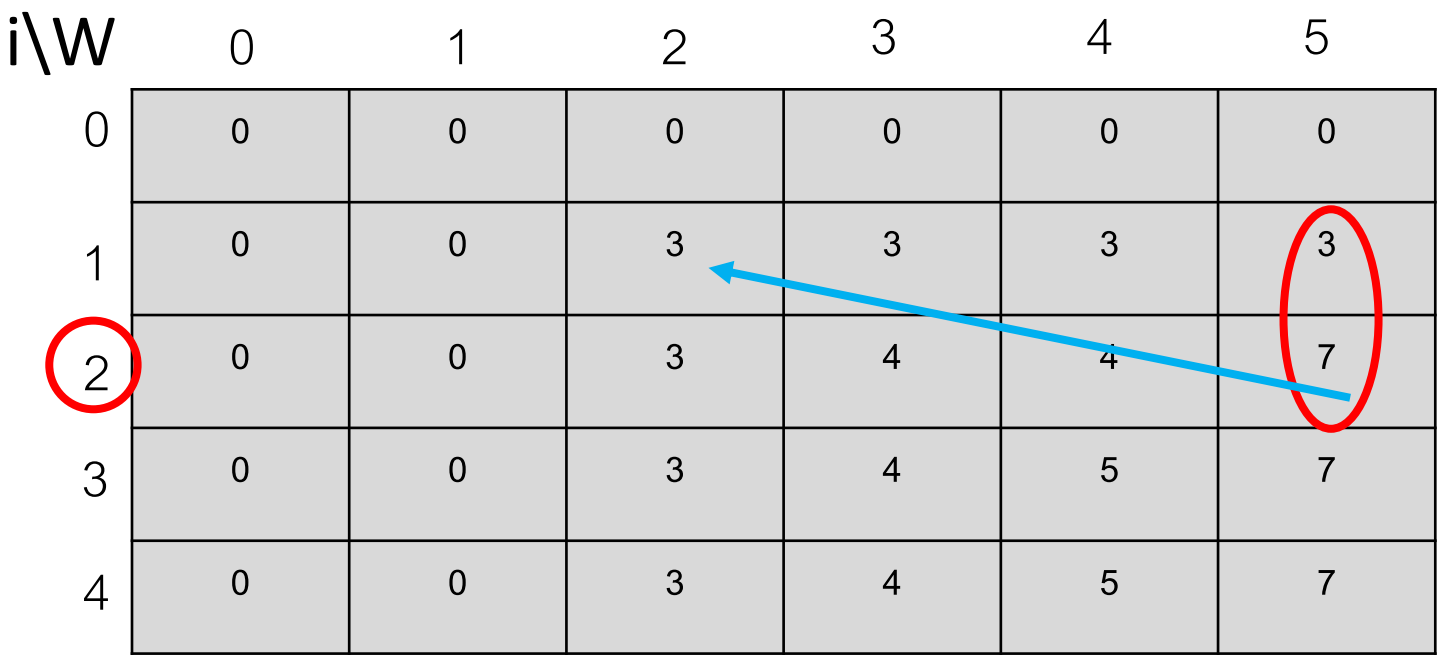
Item(w,b)
1 (2,3)
2 (3,4)
3 (4,5)
4 (5,6)

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



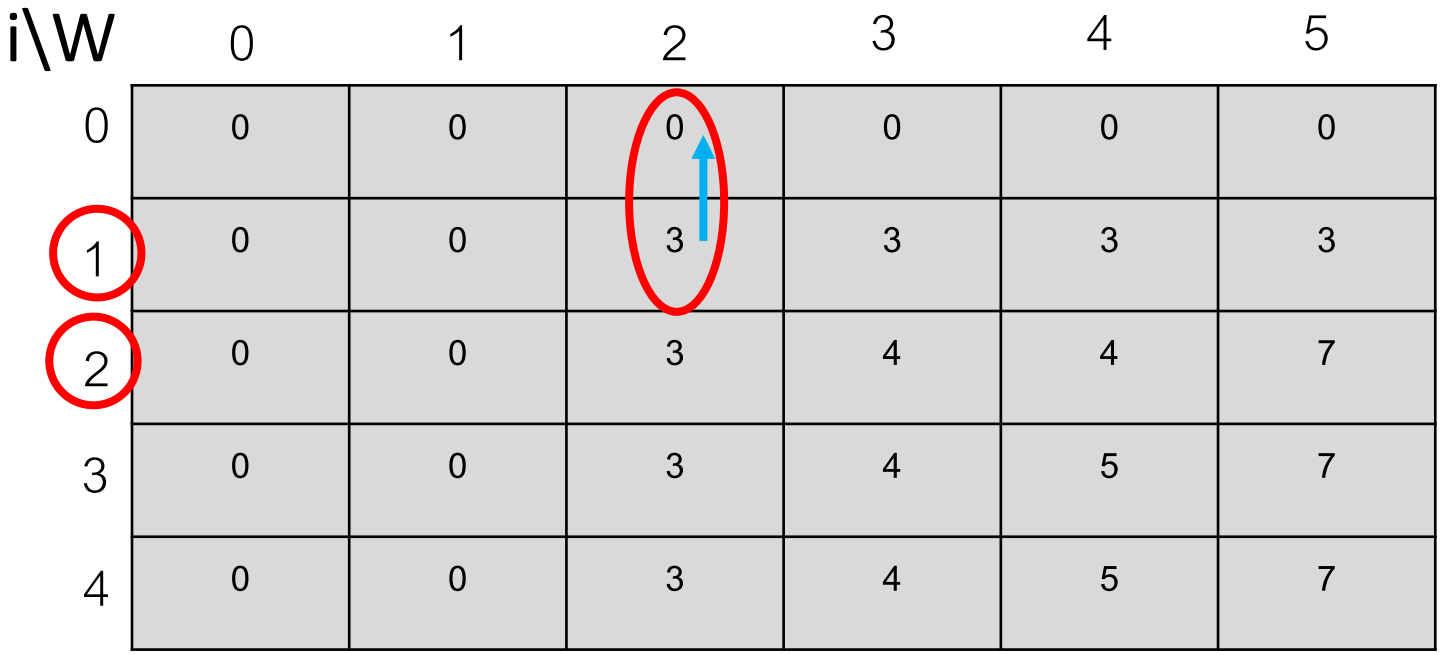
```
i=n, k=W
while i,k>0
    if B[i,k] != B[i-1,k] then
        mark the i^th item as in the knapsack
        i=i-1, k=k-w_i
    else
        i=i-1
```

Item(w,b)
1 (2,3)
2 (3,4)
3 (4,5)
4 (5,6)



i=n, k=W
while i,k>0
    if B[i,k] != 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

Item(w,b)
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=n, k=W
while i,k>0
    if B[i,k] != 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

Item(w,b)
1 (2,3)
2 (3,4)
3 (4,5)
4 (5,6)

The optimal knapsack should be {1,2}