

0-1 Knapsack problem

Knapsack problem

กำหนดให้มีวัตถุหรือสิ่งของหลายๆ ชิ้น ($\text{Item}[1..n]$) ซึ่งสิ่งของแต่ละชิ้นมีน้ำหนัก ($\text{Weight}[1..n]$) และมีมูลค่า ($\text{Value}[1..n]$) อยู่

เราต้องการเอาสิ่งของต่างๆ ใส่ถุงเป้ โดยให้ได้ผลรวมราคาของสิ่งของในเป้มีค่ามากที่สุด ในขณะที่ผลรวมน้ำหนักของสิ่งของที่ถุงเป้รับได้จะต้องไม่เกินค่าบางค่า (W)

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: ดังนั้น

เราสามารถเลือกหยิบหรือไม่หยิบ

ปัญหานี้แก้ได้ด้วย Dynamic programming

แบบที่ 2 สิ่งของแบ่งย่อยได้: ดังนั้น

เราสามารถเลือกส่วนของสิ่งของมาได้

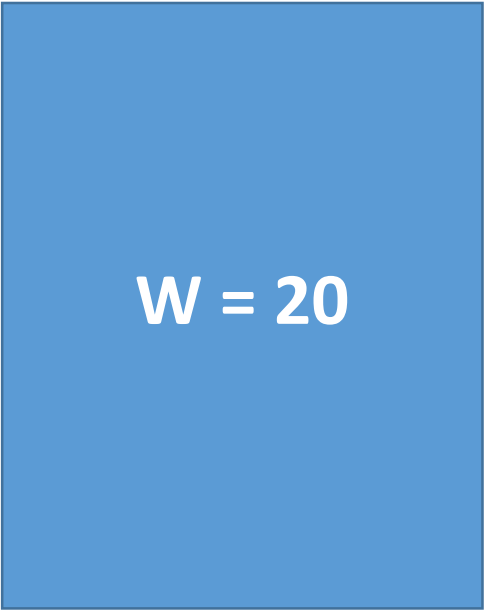


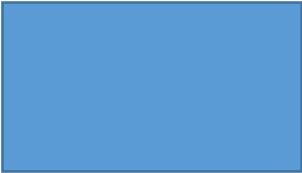


ปัญหานี้แก้ได้ด้วย Greedy algorithm (ทำได้ยังไง?)

0-1 Knapsack problem

กำหนดให้ ถุงผ้าที่มีความจุ W และเซต S ที่ประกอบด้วย
สิ่งของ n ชิ้น โดยสิ่งของแต่ละชิ้น (ชิ้นที่ i) จะมีน้ำหนัก w_i
และมีมูลค่า b_i (กำหนดให้ทุก ๆ w_i, b_i และ W เป็นจำนวนเต็ม)

ปัญหา จะบรรจุสิ่งของลงถุงผ้าอย่างไรเพื่อให้สิ่งของที่ถูก
บรรจุมีผลรวมมูลค่ามากที่สุด

0-1 Knapsack problem

	Weight		Benefit value	
	Item	w_i	b_i	
knapsack Max weight: $W = 20$ 		2	3	
		3	4	
		4	5	
		5	8	
		9	10	

Brute-force algorithm

- หากแก้ปัญหานี้แบบตรงๆ

เนื่องจากมีสิ่งของ n ชิ้น เพราะฉะนั้นจะมีรูปแบบที่เป็นไปได้ 2^n แบบ (สิ่งของหนึ่งชิ้นมี 2 ทางเลือกคือ เลือกเอาและเลือกไม่เอา)

เราจะลองทุกๆ รูปแบบนี้และหาแบบที่ได้มูลค่ารวมที่มากที่สุดโดยที่มีน้ำหนักรวมไม่เกิน W

ดังนั้นใช้เวลาในการทำงาน $O(2^n)$

คำถาม : จะทำได้ดีกว่านี้ไหม

คำตอบ : ได้ ใช้ dynamic programming

ดังนั้นเราต้องระบุ subproblem ให้ได้

Dynamic programming

1. นิยาม subproblem

หากกำหนดให้ชื่อสิ่งของเป็นหมายเลข 1, 2, ..., n แล้ว
เราจะนิยาม subproblem จะเป็น

S_k = คำตอบที่ดีที่สุดในการเลือกสิ่งของหมายเลขที่ 1
จนถึงหมายเลขที่ k

เหมาะสมไหม

คำถาม: เราสามารถอธิบายคำตอบสุดท้าย S_n ได้ด้วย S_k หรือไม่

คำตอบ: ทำไม่ได้

Dynamic programming

1. นิยาม subproblem

ตัวอย่างกำหนดสิ่งของ 5 ชิ้น แต่ละชิ้นค่าน้ำหนักและมูลค่าดังตาราง และค่า Max weight ของถุงเป้เป็น 20

- หากพิจารณาครบ 5 ชิ้น จะได้

S_5 = คำตอบที่ดีที่สุดในการเลือกสิ่งของหมายเลข
ที่ 1 จนถึงหมายเลขที่ 5 นั่นคือจะได้ถุงเป้มี
น้ำหนักรวม 20 และมูลค่ารวม 26

$W_1=2$	$W_3=4$	$W_4=5$	$W_5=9$
$b_1=3$	$b_3=5$	$b_4=8$	$b_5=10$

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

- คำถาม S_4 เป็นส่วนหนึ่งของคำตอบของ S_5 หรือไม่

Dynamic programming

1. นิยาม subproblem

พิจารณา S_4 : น้ำหนักรวม 14, มูลค่ารวม: 20

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

พิจารณา S_5 : น้ำหนักรวม 20, มูลค่ารวม: 26

$W_1=2$	$W_3=4$	$W_4=5$	$W_5=9$
$b_1=3$	$b_3=5$	$b_4=8$	$b_5=10$

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

พบว่าคำตอบของ S_4 ไม่ได้เป็นส่วนหนึ่งของ S_5

- กรณีเลือกสิ่งของชิ้นที่ 5 จะไม่ได้ใช้คำตอบที่ดีที่สุด S_4 โดยตรงแต่ต้องพิจารณากรณีที่ถูกรับได้มูลค่ารวมสูงสุดแต่น้ำหนักรวมไม่เกิน 20 ดังนั้นจะต้องพิจารณาจากน้ำหนักรวมที่ถูกรับได้ด้วย

Dynamic programming

เราพบว่าคำตอบของ S_4 ไม่ได้เป็นส่วนหนึ่งของคำตอบของ S_5

แสดงว่าการนิยาม S_k ยังไม่เพียงพอ แล้วทำอย่างไรดี

เราพบว่าจริงๆ แล้วมี parameter 2 ตัวที่ต้องคำนึงถึง

ดังนั้นเราต้องเพิ่ม parameter อีกหนึ่งตัว คือ w ซึ่งแทนน้ำหนักจริงของแต่ละ subset ของสิ่งของ

นิยามใหม่ subproblem จะเป็นการคำนวณ $B[k,w]$

Dynamic programming

2. หา recurrence ของ subproblem

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

หมายความว่า ชั้นเซตของสิ่งของที่ดีที่สุดของ S_k ที่มีน้ำหนักรวม w คือ

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

นั่นคือชั้นเซตที่ดีที่สุดของ S_k มีน้ำหนักรวม w มีการเลือกหรือไม่เลือกสิ่งของ k

กรณีแรก $w_k > w$ สิ่งของ k ไม่สามารถเป็นส่วนหนึ่งของคำตอบได้

กรณีที่สอง $w_k \leq w$ สิ่งของ k สามารถเป็นส่วนหนึ่งของคำตอบได้

และเราจะเลือกกรณีที่ให้ค่ามากที่สุด

Dynamic programming

3. หา base case

1. หากมีสิ่งของแต่ถุงเป้มีขนาดเป็น 0 แสดงว่าเลือกได้ไหม,มูลค่ารวม=?
2. หากไม่มีของแต่ถุงเป้มีขนาดต่าง ๆ แสดงว่าเลือกได้ไหม,มูลค่ารวม=?

กรณีแรกคือ

for $i=1$ to n

$B[i,0]=0$

กรณีที่สองคือ

for $w=0$ to W

$B[0,W]=0$

Dynamic programming : Algorithm

```

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  $w_i \leq w$ 
            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  $B[i, w] = B[i-1, w]$  //  $w_i > w$ 

```

Dynamic programming : Example

$n=4$

$W=5$

สิ่งของ (น้ำหนัก, มูลค่า)

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

Dynamic programming : Example

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

Dynamic programming : Example

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

ใส่ค่า base case

for $i=1$ to n

$B[i,0]=0$

for $w=0$ to W

$B[0,w]=0$

Dynamic programming : Example

$i \backslash W$	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  $w_i \leq w$ 
  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)

Dynamic programming : Example

$i \backslash w$	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  $w_i \leq w$ 
  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)

Dynamic programming : Example

$i \backslash w$	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  $w_i \leq w$ 
  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)

Dynamic programming : Example

$i \backslash w$	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 $w_i \leq w$
 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)

Dynamic programming : Example

$i \backslash w$	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  $w_i \leq w$ 
  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)

Dynamic programming : Example

$i \backslash w$	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  $w_i \leq w$ 
  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)

Dynamic programming : Example

$i \backslash w$	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 $w_i \leq w$
 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)

Dynamic programming : Example

$i \backslash w$	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  $w_i \leq w$ 
  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)

Dynamic programming : Example

$i \backslash w$	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  $w_i \leq w$ 
  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)

Dynamic programming : Example

$i \backslash w$	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 $w_i \leq w$
 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)

Dynamic programming : Practice

$i \backslash w$	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  $w_i \leq w$ 
    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)

Dynamic programming

$i \backslash w$	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  $w_i \leq w$ 
    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)

รู้ได้อย่างไรว่าเราใส่สิ่งของใดลงในเป้บ้าง

ตอนนี้ข้อมูลทุกอย่างอยู่ในตาราง

$B[n, W]$ เป็นค่ามากที่สุดของสิ่งของที่ถูกใส่เข้าไปในถุงเป้

ให้ $i=n$ และ $k=W$

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

$i=i-1$ //Assume the i^{th} item is not in the knapsack

//Could it be in the optimally packed knapsack?

ตัวอย่างการหาว่าเราใส่สิ่งของใดลงในเป้บ้าง

$i \backslash w$	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  $i^{\text{th}}$  item as in the knapsack
        i=i-1, k=k-wi
    else
        i=i-1
  
```

Item(w,b)

1 (2,3)

2 (3,4)

3 (4,5)

4 (5,6)

ตัวอย่างการหาว่าเราใส่สิ่งของใดลงในเป้บ้าง

$i \backslash w$	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  $i^{\text{th}}$  item as in the knapsack
    i=i-1, k=k-wi
  else
    i=i-1
  
```

Item(w,b)

1 (2,3)

2 (3,4)

3 (4,5)

4 (5,6)

ตัวอย่างการหาว่าเราใส่สิ่งของใดลงในเป้บ้าง

$i \backslash w$	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  $i^{\text{th}}$  item as in the knapsack
    i=i-1, k=k-wi
  else
    i=i-1
  
```

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

ตัวอย่างการหาว่าเราใส่สิ่งของใดลงในเป้บ้าง

$i \backslash w$	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  $i^{\text{th}}$  item as in the knapsack
        i=i-1, k=k-wi
    else
        i=i-1
  
```

Item(w,b)

1 (2,3)

2 (3,4)

3 (4,5)

4 (5,6)

ตัวอย่างการหาว่าเราใส่สิ่งของใดลงในเป้บ้าง

$i \backslash w$	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  $i^{\text{th}}$  item as in the knapsack
    i=i-1, k=k-wi
  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}

Practice

- กำหนดให้มีถุงเป็ขนาด 8 หน่วย และมีสิ่งของ 4 ชิ้นที่มีมูลค่าและน้ำหนักดังนี้

Item#	มูลค่า	น้ำหนัก
1	15	1
2	10	5
3	9	3
4	5	4

- จงเลือกสิ่งของเพื่อให้ได้มูลค่ารวมสูงสุดที่มีน้ำหนักรวมไม่เกิน น้ำหนักที่ถุงเป็รับได้และให้บอกน้ำหนักที่ได้ด้วย