Dynamic Programming - 2

lecturer : double
Credit by baluteshih, lawfung

行事曆

第二階段		
1	5/06	Dynamic programming (2)
2	5/13	Computational Geometry
3	5/20	[北] String / [竹] ? / [南] Dynamic programming (3)
4	5/27	Graph(1)
5	6/03	[北] Randomize / [竹] ? / [南] Segment Tree
6	6/10	Graph (2)
	6/17	端午節補課, 放假
7	6/24	Team Contest
8	7/01	Exam

影片 Q&A

- 有問題嗎？

背包問題

- 影片中大家學了好多背包問題的變形
- 不用每個都死記，理解概念才是學習要領
  - 說穿了，每個都是可以獨立思考後得出的 DP 問題

- 這堂課我們會帶大家看更多變形
  - 讓你們沒辦法全部死記(O

背包問題

- 先講好一些常用符號

- w 是重量， v 是價值
- 重量上限是 W
- 物品有 N 種

背包問題

- 背包問題的原形有三種，讓我們一一來複習！
- 1. 0/1 背包問題
- 2. 無限背包問題
- 3. 有限背包問題

1. 0/1 背包問題

每個東西只有一個，取或不取

1. 0/1 背包問題

$$f(n,m) = \max(f(n-1,m),\ f(n-1,m-w_i)+v_i)$$

1. 0/1 背包問題

$f(n,m) = \max(f(n-1,m),\ f(n-1,m-w_i)+v_i)$

可以滾動陣列嗎？

1. 0/1 背包問題

$f(n,m) = \max(f(n-1,m), f(n-1,m-w_i)+v_i)$

可以滾動陣列

```
for i = 1...N :
    for j = 0...W :
        f[i%2][j] = max(f[(i-1)%2][j],
                        f[(i-1)%2][j-w[i]]+v[i])
```

1. 0/1 背包問題

$f(n,m) = \max(f(n-1,m), f(n-1,m-w_i)+v_i)$

可以滾動陣列，可以壓成一維陣列嗎？

1. 0/1 背包問題

$$f(n,m) = \max(f(n-1,m),\ f(n-1,m-w_i)+v_i)$$

可以滾動陣列，甚至壓成一維陣列

```
for i = 1...N :
    for j = W...0 :
        f[j] = max(f[j], f[j-w[i]]+v[i])
```

Sprout

2. 無限背包問題

每個東西有無限多個，愛取幾個就取幾個

2. 無限背包問題

$f(n,m) = \max(f(n-1,m),\ f(n,m-w_i)+v_i)$

可以滾動陣列，或是甚至壓成一維陣列

```
for i = 1...N :
    for j = 0...W :
        f[j] = max(f[j], f[j-w[i]]+v[i])
```

2. 無限背包問題

這是正確的 code 嗎？

```
for j = 0...W :
   for i = 1...N :
      f[j] = max(f[j], f[j-w[i]]+v[i])
```

2. 無限背包問題

這是正確的 code 嗎？

```
for j = 0...W :
   for i = 1...N :
        f[j] = max(f[j], f[j-w[i]]+v[i])
```

是

3. 有限背包問題

第 i 個東西有 t_i 個，也就是可以取 0~t_i 個

3. 有限背包問題

$f(n,m) = \max(f(n-1,m-k*w_i)+k*v_i)$, for $0 \le k \le t_i$

可以滾動陣列，或是甚至壓成一維陣列

```
for i = 1...N :
    for j = W...0 :
        for k = 0...t[i]:
            f[j] = max(f[j], f[j-k*w[i]]+k*v[i])
```

複雜度 $O(N*\max\{t_i\}*W)$

或是精確一點來說， $O(\mathrm{sum}\{t_i\}*W)$

物品拆分

我們可以做點變化：把第 i 個物品當做 t_i 個不同的物品，但有一樣的 w_i 和 v_i，然後做一般的背包問題。

但是複雜度沒有變： $O(\text{sum}\{t_i\}*W)$

想想看

- 要組合出 0~N 的數字，需要把 N 拆成 N 個 1 嗎？
- 用多少個數字可以組合出 0~8 所有數字呢？
  是 8 個嗎？
- 最少可以拆成幾個呢？ 為什麼？

物品拆分

將 t_i 個物品拆成

1 個、2 個、4 個、....、2^h 個、$t_i-(2^{h+1}-1)$ 個

其中 h 是滿足 $(2^{h+1}-1) \le t_i$ 的最大整數。

如此一來 t_i 個物品就可以拆成 $\log(t_i)$ 種不同物品就好！

然後新的背包就會有 $\mathrm{sum}\{\log(t_i)\}$ 個物品，每個物品選或不選，

變回 0/1 背包問題了！

複雜度 : $O(N*\log(\max\{t_i\})*W)$

3. 有限背包問題

N’ 是新的物品數量(至多 N*log(max{t_i}))

```
for i = 1...N’ :
    for j = W...0 :
        f[j] = max(f[j], f[j-w[i]]+v[i])
```

其實有 O(NW) 的做法，以後的影片會教到

4. 二維背包問題

每個東西有重量(w_i), 價錢(c_i), 跟價值(v_i)
你除了要滿足 $\text{sum}\{w_i\}\le W$, 還要滿足 $\text{sum}\{c_i\}\le C$
一樣問最大價值？

4. 二維背包問題

每個東西有重量(w_i), 價錢(c_i), 跟價值(v_i)
你除了要滿足 $\mathrm{sum}\{w_i\} \le W$, 還要滿足 $\mathrm{sum}\{c_i\} \le C$
一樣問最大價值？

開狀態 f[N][W][C] 即可, 轉移為
$f(n,w,c) = \max(f(n-1,w,c),\ f(n-1,w-w_i,c-c_i)+v_i)$
實作上一模一樣！

5. 分組背包問題

有 T 組物品，每個物品跟以往一樣有重量，價值。
但是每組物品裡面最多只能選一個。

請問最大價值？

5. 分組背包問題

```
for i = 1...T :
    for j = W...0 :
        for k = 1...size(i):
            f[j] = max(f[j], f[j-w[i][k]]+v[i][k])
```

複雜度 $O(sum\{size_i\}*W)$

5. 分組背包問題

```
for i = 1...T :
    for k = 1...size(i):
        for j = W...0 :
            f[j] = max(f[j], f[j-w[i][k]]+v[i][k])
```

這是正確的 code 嗎？

5. 分組背包問題

這是正確的 code 嗎？

```
for i = 1...T :
    for k = 1...size(i):
        for j = W...0 :
            f[j] = max(f[j], f[j-w[i][k]]+v[i][k])
```

不是

其實這是單純的 0/1 背包

6.背包合併

有兩堆物品 A、B，他們已經各自做好了背包 DP 儲存在 h、k 陣列，你想要把兩堆物品合起來一起考慮，然後重做一個背包 DP

```
for i = 0...W :
    for j = 0...i :
      f[i] = max(f[i], h[j]+k[i-j])
```

複雜度 $O(W^2)$

7. 背包問題的變化

(1)求最大價值的方法總數
(2)求最大價值的一組方案
(3)求最大價值的字典序最小的一組方案
(4)求次大價值的解 / 第 K 大價值的解

7. 背包問題的變化

(1)求最大價值的方法總數

```
for i = 1...N :
    for j = W...0 :
        if f[j] < f[j-w[i]] + v[i] :
            f[j] = f[j-w[i]] + v[i]
            g[j] = g[j-w[i]]
        else if f[j] == f[j-w[i]] + v[i] :
            g[j] += g[j-w[i]]
```

7. 背包問題的變化

(2)求最大價值的一組方案

```
for i = 1...N :
   for j = W...0 :
      if f[i-1][j] < f[i-1][j-w[i]] + v[i] :
         f[i][j] = f[i-1][j-w[i]]+v[i]
         g[i][j] = 1
      else:
         f[i][j] = f[i-1][j]
         g[i][j] = 0
```

7. 背包問題的變化

(2)求最大價值的一組方案

```
for i = 1...N :
    for j = W...0 :
        if f[i-1][j] < f[i-1][j-w[i]] + v[i] :
            f[i][j] = f[i-1][j-w[i]] + v[i]
            g[i][j] = 1
        else:
            f[i][j] = f[i-1][j]
            g[i][j] = 0
```

倒著沿著 g[i][j] 隨意的走回去

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案

遇到求解、又要求字典序的問題，通常會有兩種做法：

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案

遇到求解、又要求字典序的問題，通常會有兩種做法：

1. 直接在狀態存著「最小字典序」的方案
2. 「從頭」開始貪心的取

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案(方法一)

```
for i = 1...N :
    for j = W...0 :
        if f[j] < f[j-w[i]] + v[i] :
            f[j] = f[j-w[i]] + v[i]
            g[j] = g[j-w[i]] ∪ i
        else if f[j] == f[j-w[i]] + v[i] :
            g[j] = min(g[j], g[j-w[i]] ∪ i)
```

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案(方法一)

```
for i = 1...N :
    for j = W...0 :
        if f[j] < f[j-w[i]] + v[i] :
            f[j] = f[j-w[i]] + v[i]
            g[j] = g[j-w[i]] ∪ i
        else if f[j] == f[j-w[i]] + v[i] :
            g[j] = min(g[j], g[j-w[i]] ∪ i)
```

複雜度會多一個 N！

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案(方法二)

- 第一個物品可以取多小的？

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案(方法二)

- 第一個物品可以取多小的？

- 窮舉第一個物品，假設窮舉到 i，最佳解是 ans
- 檢查「i+1~n」能不能用容量「W-w[i]」湊出「ans-v[i]」

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案(方法二)

- 第一個物品可以取多小的？

- 窮舉第一個物品，假設窮舉到 i，最佳解是 ans
- 檢查「i+1~N」能不能用容量「W-w[i]」湊出「ans-v[i]」
- 能的話，取 i 進來當答案
- 遞迴求「i+1~N」、容量「W-w[i]」、最佳解「ans-v[i]」的最小字典序

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案(方法二)

- 第一個物品可以取多小的？

- 窮舉第一個物品，假設窮舉到 i，最佳解是 ans
- 檢查「i+1~N」能不能用容量「W-w[i]」湊出「ans-v[i]」
- 能的話，取 i 進來當答案
- 遞迴求「i+1~N」、容量「W-w[i]」、最佳解「ans-v[i]」的最小字典序
- 我們需要什麼？

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案(方法二)

- 第一個物品可以取多小的？

- 窮舉第一個物品，假設窮舉到 i，最佳解是 ans
- 檢查「i+1~N」能不能用容量「W-w[i]」湊出「ans-v[i]」
- 能的話，取 i 進來當答案
- 遞迴求「i+1~N」、容量「W-w[i]」、最佳解「ans-v[i]」的最小字典序
- 我們需要什麼？
- 反過來的背包 DP 表格

Sprout

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案(方法二)

```
// dp[i][j]:= i~n 的物品湊出 j 的最佳解
ans = max(dp[1][0]~dp[1][W])
for i = 1...N :
    if max(dp[i+1][0]~dp[i+1][W-w[i]]) == ans - v[i]:
        把 i 加進答案
        W -= w[i]
        ans -= v[i]
        continue
```

7. 背包問題的變化

(3)求最大價值的字典序最小一組方案(方法二)

```
// dp[i][j]:= i~n 的物品湊出 j 的最佳解
ans = max(dp[1][0]~dp[1][W])
for i = 1...N :
   if max(dp[i+1][0]~dp[i+1][W-w[i]]) == ans - v[i]:
      把 i 加進答案
      W -= w[i]
      ans -= v[i]
      continue
```

複雜度不變！

Sprout

7. 背包問題的變化

(4)求第 K 大價值

```
//f[i][j] becomes a sorted vector
for i = 1...N :
    for j = W...0 :
        for k = 0...K-1:
            vec.push_back(f[i-1][j][k])
            vec.push_back(f[i-1][j-w[i]][k]+v[i])
        sort(vec)
        f[i][j] = vec[0...K-1]
```

8. 分數背包

- 每個物品都可以只取部分
- 貪心那次手寫作業應該寫過了

Recall

- 用重量做為狀態
  - 空間複雜度:O(W)
  - 時間複雜度:O(NW)
  - 限制:W 不能太大
- 用價值做為狀態
  - 空間複雜度:O(V)
  - 時間複雜度:O(NV)
  - 限制:V 不能太大
  - (注意到 V 是所有物品價值總和)

9. V 跟 W 都很大的背包

- 那如果 V 和 W 都很大怎麼辦
- 例如 $V \le 10^9$, $W \le 10^9$
- 但是 $N \le 20$

9. V 跟 W 都很大的背包

- 不就枚舉就好
- 複雜度 $O(2^N)$

9. V 跟 W 都很大的背包

- 那如果 $N \leq 40$ ？

9. V 跟 W 都很大的背包

- 那如果 N ≤ 40 ？
- 拆一半？

9. V 跟 W 都很大的背包

- 那如果 N ≤ 40 ？
- 拆一半？

- 分兩半各自暴力枚舉所有可能
- 答案肯定是來自兩半中各選一種可能！

9. V 跟 W 都很大的背包

- 那如果 N ≤ 40 ？
- 拆一半？

- 分兩半各自暴力枚舉所有可能
- 答案肯定是來自兩半中各選一種可能！
- 窮舉其中一半，另一半找「能配對的最佳解」

9. V 跟 W 都很大的背包

- 那如果 N ≤ 40 ？
- 拆一半？

- 分兩半各自暴力枚舉所有可能
- 答案肯定是來自兩半中各選一種可能！
- 窮舉其中一半，另一半找「能配對的最佳解」

- 這招我們通常稱作「折半枚舉」
- 或是叫做 meet in the middle
- 複雜度 $O(N2^{N/2})$

換零錢問題

- 回想計算方法數的換零錢問題
- N 種硬幣，湊出 M 元的方法數有幾種？
  - f(n,m) = f(n-1,m) + f(n-1,m-c[n])

換零錢問題

- N 種硬幣 $c_1, c_2, \ldots, c_n$
- 求對於所有長度 k 的區間，湊出 M 的方法數有幾種？
- 要求 O(NM)

可回溯換零錢問題

- n 種硬幣 $c_1, c_2, \ldots, c_n$
- 求對於所有長度 k 的區間，湊出 M 的方法數有幾種？
- 要求 O(NM)

- 觀察 1：交換硬幣的順序不影響答案

可回溯換零錢問題

- n 種硬幣 $c_1, c_2, \ldots, c_n$
- 求對於所有長度 k 的區間，湊出 M 的方法數有幾種？
- 要求 O(NM)

- 觀察 1:交換硬幣的順序不影響答案
- 觀察 2:計數 DP 可以倒著做回去

可回溯換零錢問題

- n 種硬幣 $c_1, c_2, \ldots, c_n$
- 求對於所有長度 k 的區間，湊出 M 的方法數有幾種？
- 要求 $O(NM)$

- 觀察 1：交換硬幣的順序不影響答案
- 觀察 2：計數 DP 可以倒著做回去
- 酷似爬行法的維護區間的 DP 值？