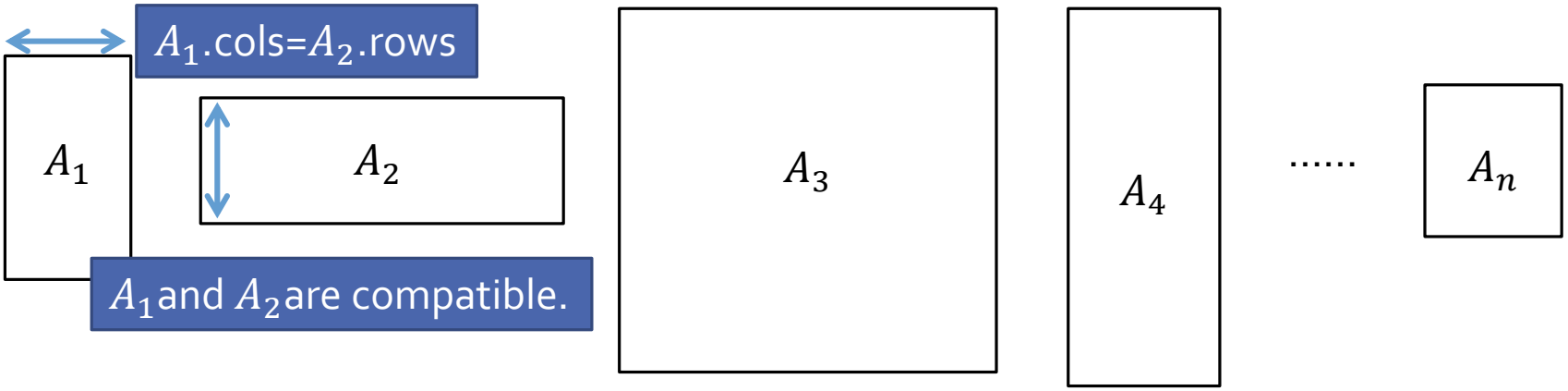
Dynamic Programming II

Michael Tsai

2013/10/10

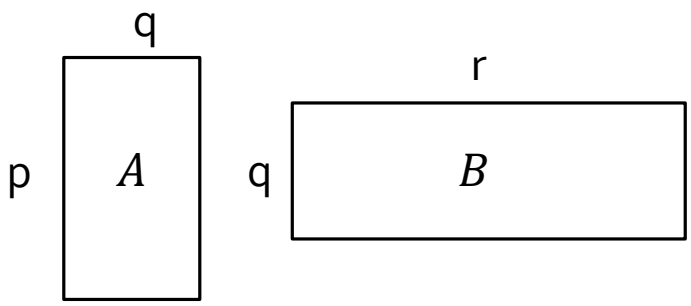
連串矩陣相乘問題

題目: 求$A_1A_2 \ldots A_n$矩陣相乘之解.



- Matrix multiplication is associative.
- $(((A_1A_2)A_3)A_4)$
- $((A_1(A_2A_3))A_4)$
- $((A_1A_2)(A_3A_4))$
- $(A_1((A_2A_3)A_4))$
- $\left(A_1\left(A_2(A_3A_4)\right)\right)$
- 以上算出來答案都一樣

連串矩陣相乘問題

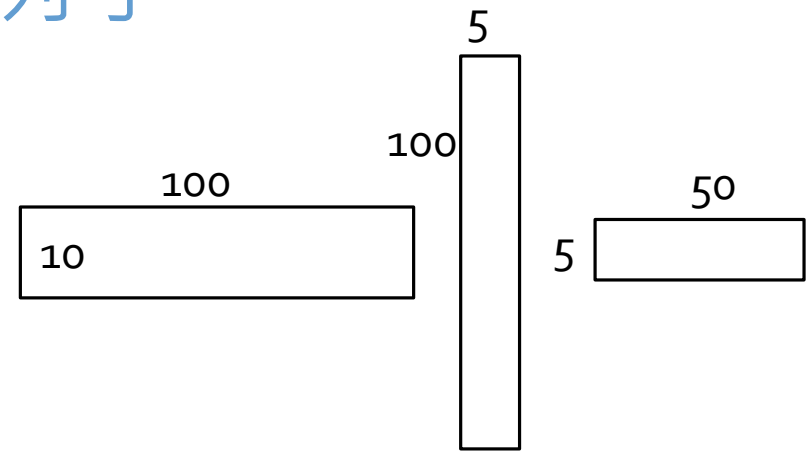



```
Matrix-Multiply(A,B)
if A.columns != B.rows
      error "incompatible dimensions"
else let C be a new A.rows x B.cols matrix
      for i=1 to A.rows                    p
            for j=1 to B.cols              r
                  c_ij=0
                  for k=1 to A.cols        q
                        c_ij = c_ij + a_ik · b_kj
      return C
```

主要花費時間在這邊!

共花費pqr次乘法的時間

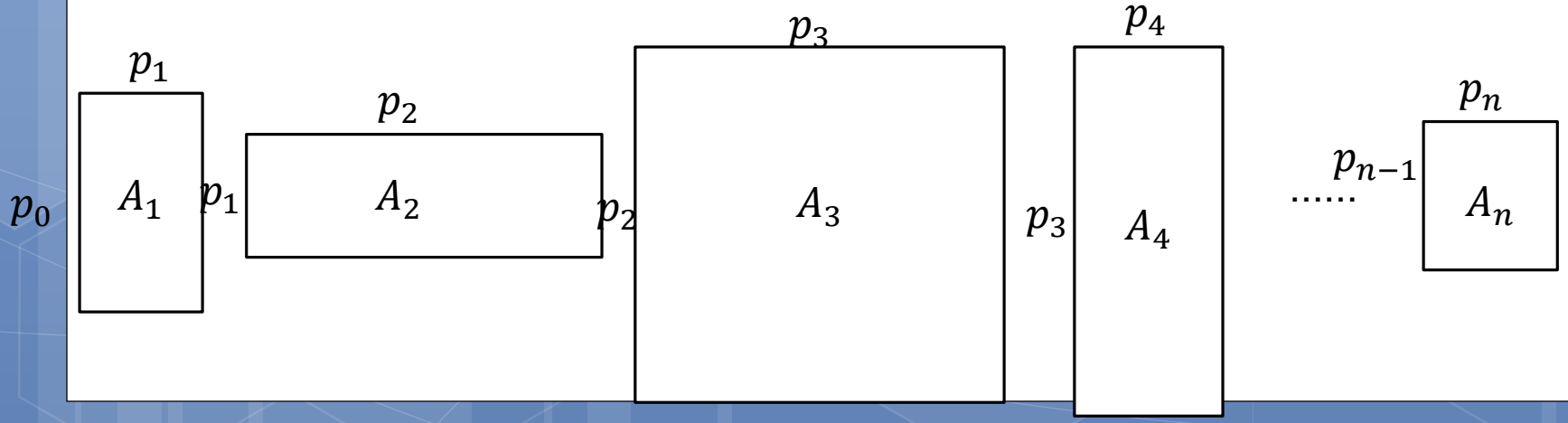
看一個例子



- $((A_1A_2)A_3)$花費之乘法數目=$10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$
- $(A_1(A_2A_3))$花費之乘法數目=$100 \times 5 \times 50 + 10 \times 100 \times 50 = 75000$
- 差十倍!!

連串矩陣相乘問題-正式版

- 給一連串的矩陣$\langle A_1, A_2, \ldots, A_n \rangle$, 其中矩陣$A_i$的大小為$p_{i-1} \times p_i \ (i = 1, 2, \ldots, n)$, 找出一種乘法可以使計算時的乘法數目最少
- 沒有真的要算結果, 而只是找出能最快算出結果的方法.
- 因為算"怎麼算比較快"多花的時間, 比"用爛方法直接算"多花的時間少很多



暴力法有多暴力

- 全部到底有幾種算法呢?
- P(n): 代表n個矩陣相乘共有幾種算法
- 用遞迴定義:

$$P(n) = \begin{cases} 1 & \text{if } n = 1, \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 2. \end{cases}$$

假設先這樣分:$(A_1 A_2 \ldots A_k)(A_{k+1} A_{k+2} \ldots A_n)$

- P(n)之解為Catalan numbers, $\Omega\left(\frac{4^n}{n^{\frac{3}{2}}}\right)$, or is also $\Omega(2^n)$

所以不要暴力了.

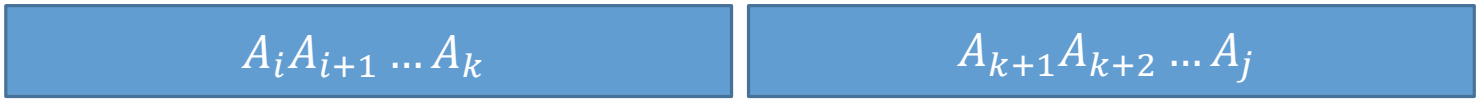
- 使用dynamic programming
- 正規步驟:

1. 找出最佳解的"結構"
2. 使用遞迴來定義最佳解的花費
3. 計算最佳解的花費
4. 使用已經計算的資訊來構築最佳解

找出最佳解的”結構”

$i \le j$

在k切一刀

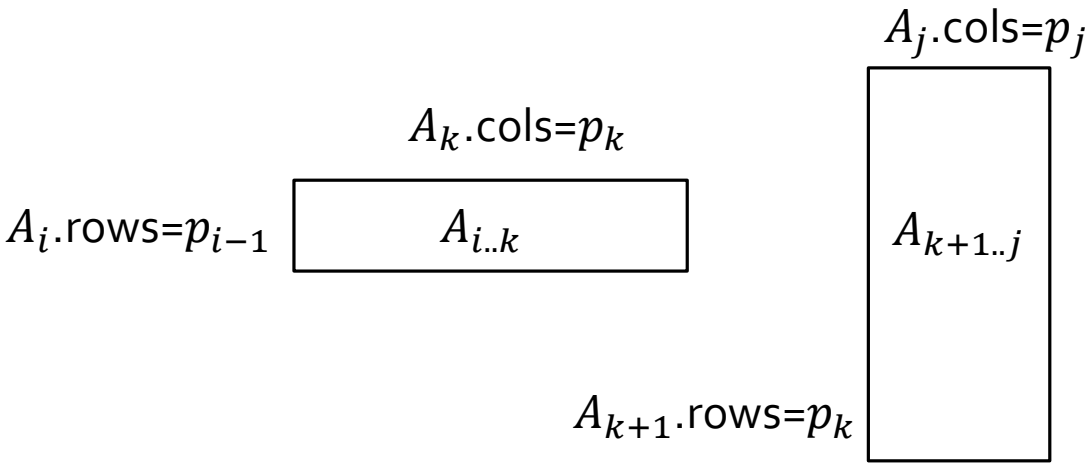


$i \le k < j$

- 總花費=$A_i..A_k$的花費+$A_{k+1}..A_j$的花費+把$A_i..A_k$和$A_{k+1}..A_j$乘起來的花費
- 最佳解的結構: 假設有$A_i..A_j$的最佳解, 此一方法為在k切一刀. 則在此$A_i..A_j$最佳解中, $A_i..A_k$的相乘方法一定是$A_i..A_k$的最佳相乘方法
- 假設$A_i..A_k$不是最佳解, 則我們可以在$A_i..A_j$中把$A_i..A_k$換成更好的方法, 則可以找到一個更好的$A_i..A_j$的相乘方法→矛盾.
- 子問題的最佳解可以導出大問題的最佳解!
- 最後結論: 分成兩個子問題, 並嘗試所有可以切分的地方(k值)

使用遞迴來定義最佳解的花費

- 遞迴定義:使用子問題最佳解的cost來定義大問題最佳解的cost
- 定義: $m[i,j]$為$A_i..A_j$所需花的最少乘法數
- $A_{i..k}A_{k+1..j}$所花乘法數=$p_{i-1}p_kp_j$



使用遞迴來定義最佳解的花費

- $m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_k p_j$
- 但是我們不知道最佳解中的k的值
- 因此我們必須找所有$k = i, i+1, \ldots, j-1$
- 最後版本:

$$m[i,j] = \begin{cases} 0 & \text{if } i = j, \\ \min\limits_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1}p_k p_j\} & \text{if } i < j. \end{cases}$$

計算最佳解的花費

- 純recursive解法: exponential time
- 使用前面教的Bottom-up填表方法: 每個不同的subprogram僅需解1次
- 有幾個不同的問題?
- $1 \le i \le j \le n$, 幾個i和j的組合?
- 答案: $\binom{n}{2} + \mathrm{n} = \Theta(n^2)$

$i \neq j$ $i = j$

計算最佳解的花費

- 如何決定填表的順序呢? (這次有i,j兩個變數)

$$m[i,j] = \begin{cases} 0 & \text{if } i = j, \\ \min\limits_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1}p_k p_j\} & \text{if } i < j. \end{cases}$$

$k - i + 1$個矩陣相乘　　$j - k$個矩陣相乘

都小於$j - i + 1$個

我們可以把j-i+1(也就是要相乘的matrix個數)
當作problem size的定義

計算最佳解的花費

```
n=p.length-1
let m[1..n,1..n] and s[1..n-1,2..n] be new tables
for i=1 to n
      m[i,i]=0
for l=2 to n
      for i=1 to n-l+1
            j=i+l-1
            m[i,j]=∞
            for k=i to j-1
                  q=m[i,k]+m[k+1,j]+p_{i-1} p_k p_j
                  if q<m[i,j]
                        m[i,j]=q
                        s[i,j]=k
return m and s
```

邊界條件先設好.

大problem的解只會用到小problem的解.因此慢慢往上長.

把同樣problem size的所有i,j組合都依序做過

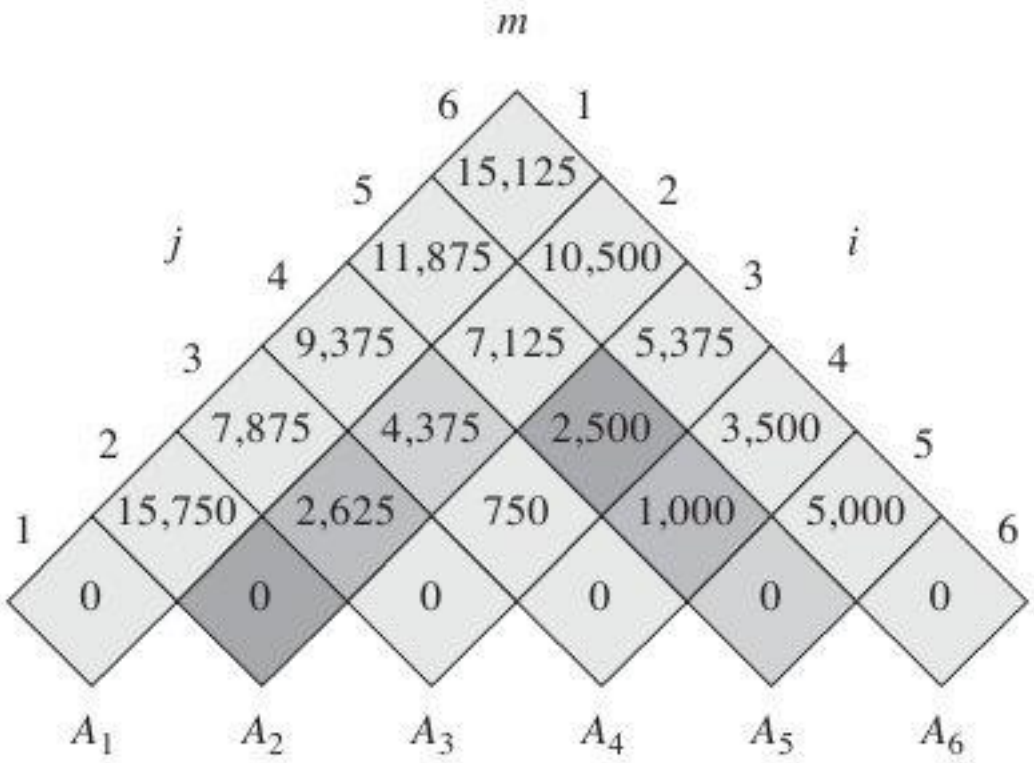
使用遞迴式找出最佳切點k

$\Theta(n^3)$

計算最佳解的花費

- 用一個例子來trace code比較快

Matrix	A_1	A_2	A_3	A_4	A_5	A_6
Dimension	30 x 35	35 x 15	15 x 5	5 x 10	10 x 20	20 x 25



使用已經計算的資訊來構築最佳解

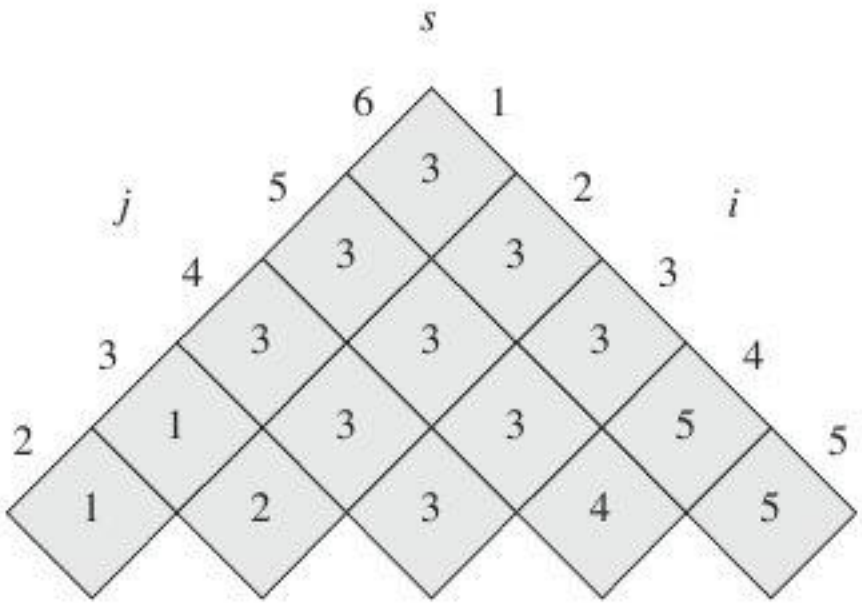
- 前面只印出了花費, 不是真正的解
- 怎麼乘才是最後的解
- 使用s陣列的資訊

使用已經計算的資訊來構築最佳解

```
Print-Optimal-Parens(s,i,j)
if i==j
      print A_i
else
      print "("
      Print-Optimal-Parens(s,i,s[i,j])
      Print-Optimal-Parens(s,s[i,j]+1,j)
      print ")"
```

使用已經計算的資訊來構築最佳解

- 所以該怎麼括?



看了兩個例子以後...

- 問: 一個問題要有什麼要件才能使用dynamic programming?
- 答:

1. Optimal substructure
2. Overlapping Subproblems

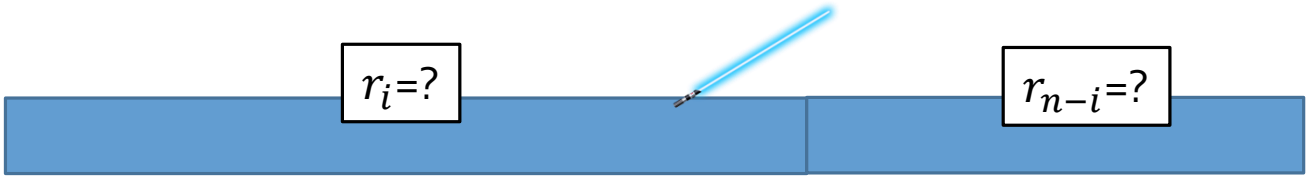
什麼是Optimal substructure?

- Definition: A problem exhibits **optimal substructure** if an optimal solution to the problem contains within it optimal solutions to subproblems.
- 怎麼尋找optimal substructure呢?

怎麼尋找optimal substructure呢?

1. 要得到問題的解答有許多選擇(砍在哪邊, 切在哪邊), 而做這個選擇之後, 我們有一些subproblem要解決.
2. 我們假設對於一個問題, 我們可以找到那個選擇
3. 知道這個選擇以後, 我們找出哪個subproblem可以被拿來應用, 及剩下的問題(沒有對應到subproblem的)怎麼解決
4. 證明大問題的最佳解中可以直接應用(剪下貼上)子問題的最佳解.

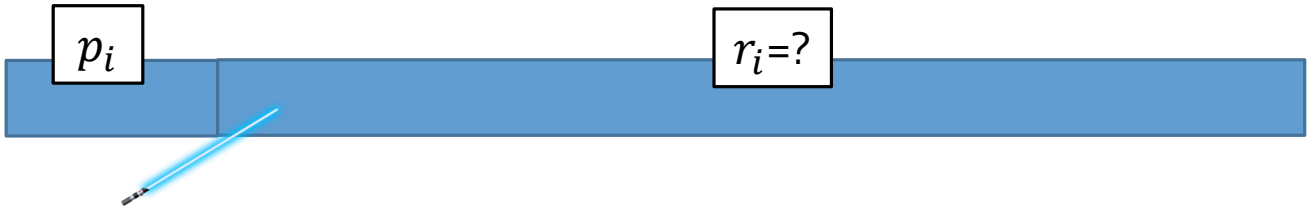
Optimal substructure越簡單越好



versus

這一段砍下來就不再砍
成更小的了(拿去賣)

這一段是subproblem, 找遞迴朋友去解



Optimal substructure越簡單越好

假設把問題定義成$A_1 \dots A_j$就好 (少一個變數)

$1 \le j$

在k切一刀

$A_1 A_{i+1} \dots A_k$ | $A_{k+1} A_{k+2} \dots A_j$

$1 \le k < j$

此為一個子問題

此不為一個子問題!

除非k一直都是j-1, 否則...

Optimal substructure的變化

1. 原始問題的最佳解用了多少個子問題
2. 大問題有多少選擇(選擇用不同的子問題們來獲得最佳解)

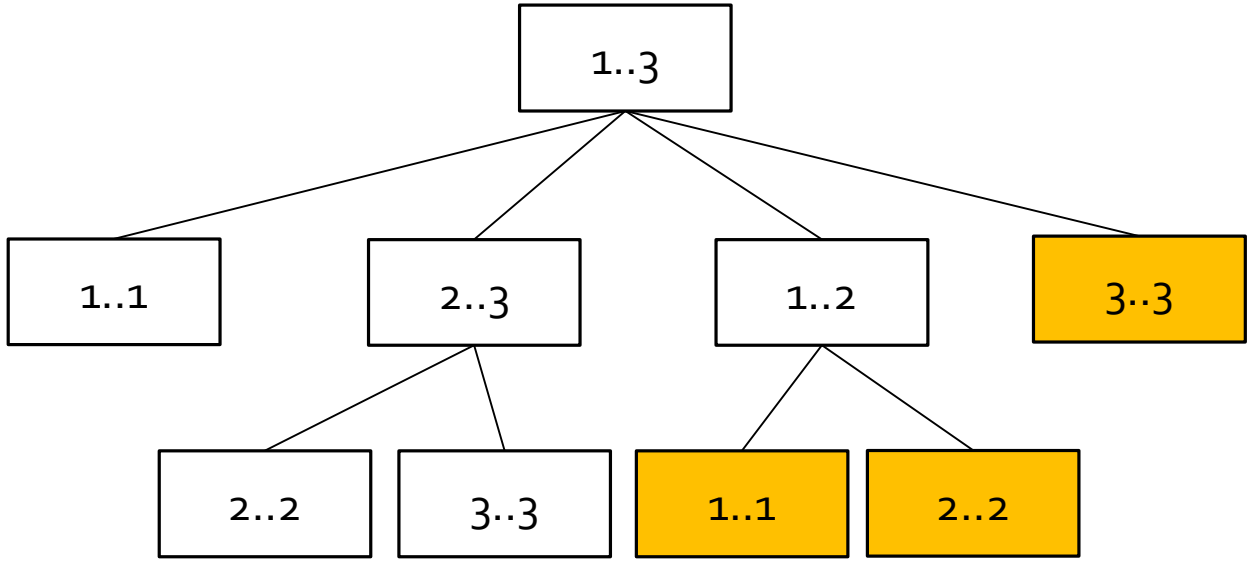
大略來說, 以上兩者決定dynamic programming algorithm的執行時間.

(之前說的Subproblem graphs是另外一種算法)

	多少個子問題	有多少選擇	執行時間
鐵條資源回收	$\Theta(n)$	n	$O(n^2)$
連串矩陣相乘	$\Theta(n^2)$	n-1	$O(n^3)$

複習: Overlapping Subproblems

- 舉個例子: 連串矩陣問題的遞迴樹



橘色的是overlap的部分!

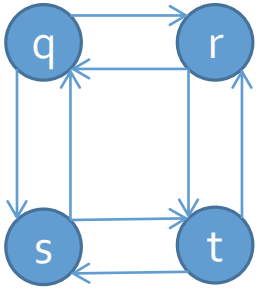
例子：有沒有optimal substructure

- 給一個graph $G = (V, E)$. $u, v \in V$. Edge沒有weight.
- 問題1:找出$u \to v$沒有loop最短路徑.
- 問題2:找出$u \to v$沒有loop最長路徑.

- 問題1有沒有optimal substructure?
- 假設找到u到v的最短路徑p, 則我們可以將其分解為 $u \overset{p_1}{\to} w \overset{p_2}{\to} v$ (w可以是u或v). 則其中p_1一定是u到w的最短路徑.
- 不然的話, 我們可以找到一個p_1'比p_1還短的u到w路徑, 那麼p_1'和p_2組合起來就變成一條比p更短的u到v路徑 (矛盾)

例子：有沒有optimal substructure

- 問題2有沒有optimal substructure?



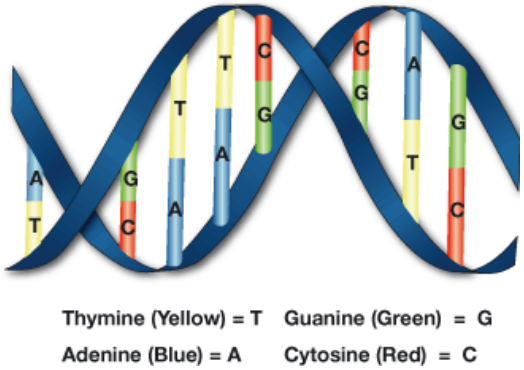
- 沒有! 來舉一個反例.
- q到t的最長路徑: $q \to r \to t$
- 但是q到r的最長路徑為$q \to s \to t \to r$
- 並不是q到t的最長路徑中間的一部分!
- r到t的最長路徑為$r \to q \to s \to t$
- 也不是q到t的最長路徑中間的一部分!

例子：有沒有optimal substructure

- 為什麼問題1和問題2相差這麼多?
- 問題2缺乏"獨立性"(subproblem的解互相之間不會影響)
- $u \xrightarrow{p_1} w \xrightarrow{p_2} v$出現在$p_1$的vertex就不能出現在$p_2$ (否則就會有loop了) →subproblem的解互相影響!

- 問題1有"獨立性"
- 在最短路徑$u \xrightarrow{p_1} w \xrightarrow{p_2} v$中, 出現在$p_1$的vertex本來就不可能出現在$p_2$
- 假設p_1, p_2中除了w以外出現了一個一樣的vertex x. 則可以將最短路徑拆解成$u \xrightarrow{p_{ux}} x \xrightarrow{p_{xw}} w \xrightarrow{p_{wx}} x \xrightarrow{p_{xv}} v$.
- 因為x和w不同, 所以$|p_{xw}| \geq 1, |p_{wx}| \geq 1$. 則$p_{ux}$和$p_{xv}$變成比原本更短的u到v的路徑 (矛盾)

DNA比對問題



- DNA序列可表示為以{A,C,G,T}組合而成的一字串

- $S_1 = ACCGGTCGAGTGCGCGGAAGCCGGCCGAA$
  $S_2 = GTCGTTCGGAATGCCGTTGCTCTGTAAA$

- 比較兩者有多相像??
- 親屬關係?

DNA比對問題

- 多相像→找出兩者中都出現的最長子序列→看最長子序列有多長, 越長越相像
- 子序列:
  - 順序相同
  - 但不一定要連續.
- 簡單的例子:
- X=ABCBDAB, Y=BDCABA
- 子序列之一: BCA
- 最長共同子序列: BCBA
- $S_1 = ACCGGTCGAGTGCGCGGAAGCCGGCCGAA$
- $S_2 = GTCGTTCGGAATGCCGTTGCTCTGTAAA$
- 最長共同子序列=?

答案在課本p.391

DNA比對問題→最長共同子序列

- 問題: 給兩字串$X = \langle x_1, x_2, \ldots, x_m \rangle, Y = \langle y_1, y_2, \ldots, y_n \rangle$, 找出最長共同子序列.
- 最長共同子序列=Longest Common Subsequence=LCS
- 問: 暴力法有多暴力?

暴力法有多暴力?

- 找出所有X之子序列, 與Y比較檢驗看看是不是Y的子序列.
- X有幾個子序列?
- 2^m個
- Running time: $\Omega(2^m)$

Dynamic Programming出招

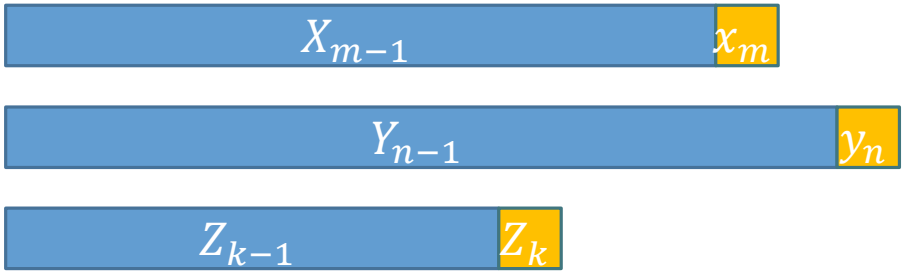
1. 找出Optimal Substructure

- 先來個小定義, 對$X = \langle x_1, x_2, \dots, x_m \rangle$, $X_i = \langle x_1, x_2, \dots, x_i \rangle, 0 \leq i \leq m$

- 先證明以下三個小定理. 給定兩字串$X = \langle x_1, x_2, \dots, x_m \rangle, Y = \langle y_1, y_2, \dots, y_n \rangle$, 及$Z = \langle z_1, z_2, \dots, z_k \rangle$為X和Y的LCS(之一)

1. If $x_m = y_m$, then $z_k = x_m = y_n$ and Z_{k-1} 是X_{m-1}及Y_{n-1}的LCS之一
2. If $x_m \neq y_n$, then $z_k \neq x_m$表示Z是X_{m-1}及Y_n的LCS之一
3. If $x_m \neq y_n$, then $z_k \neq y_n$表示Z是X_m及Y_{n-1}的LCS之一

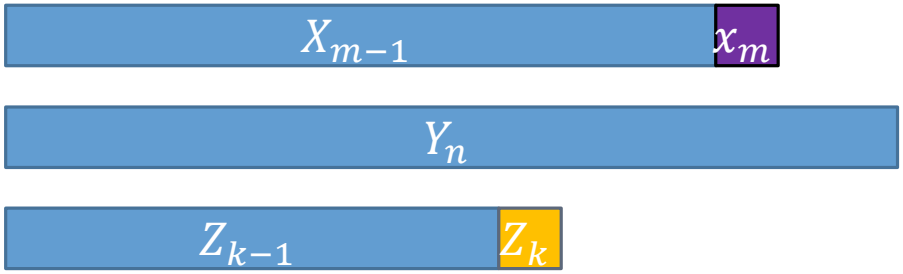
1. If $x_m = y_n$, then (1) $z_k = x_m = y_n$ and (2) Z_{k-1} 是X_{m-1}及Y_{n-1}的LCS之一



(1) Z最後一個字元一定是$x_m(= y_n)$, 否則可以把x_m加到Z的最後面成為比LCS更長的CS (矛盾)

(2)Z_{k-1}一定是X_{m-1}和Y_{n-1}的LCS. 假設不是, 則可以找到一個長度>k-1的LCS, 但是加上$x_m = y_n$這一個字元, 表示可以找到一個X_m和Y_n的LCS長度>k (矛盾)

2. If $x_m \neq y_n$, then $z_k \neq x_m$表示Z是X_{m-1}及Y_n的LCS之一



假設Z不是X_{m-1}和Y_n的LCS, 則有W為X_{m-1}和Y_n的LCS, 長度>k, 則W亦為X_m和Y_n的LCS, 長度>k (矛盾)

3. If $x_m \neq y_n$, then $z_k \neq y_n$表示Z是X_m及Y_{n-1}的LCS之一

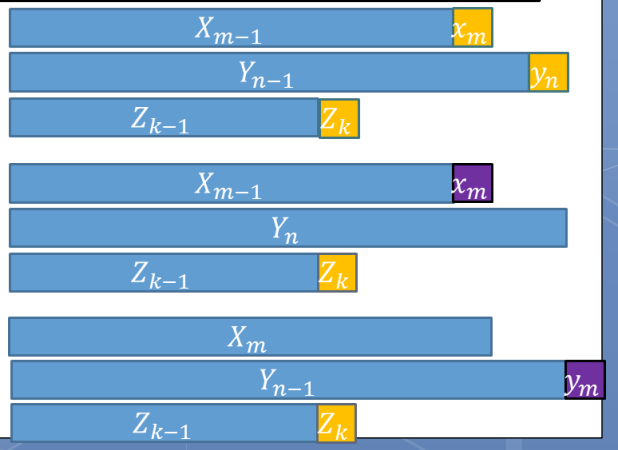
證明類似上面2.的證明.

Optimal Substructure

- 給定兩字串$X = \langle x_1, x_2, \dots, x_m \rangle, Y = \langle y_1, y_2, \dots, y_n \rangle$, 及$Z = \langle z_1, z_2, \dots, z_m \rangle$為X和Y的LCS(之一)

1. If $x_m = y_m$, then $z_k = x_m = y_n$ and Z_{k-1} 是X_{m-1}及Y_{n-1}的LCS之一
2. If $x_m \neq y_n$, then $z_k \neq x_m$表示Z是X_{m-1}及Y_n的LCS之一
3. If $x_m \neq y_n$, then $z_k \neq y_n$表示Z是X_m及Y_{n-1}的LCS之一

大問題的解裡面有小問題的解!

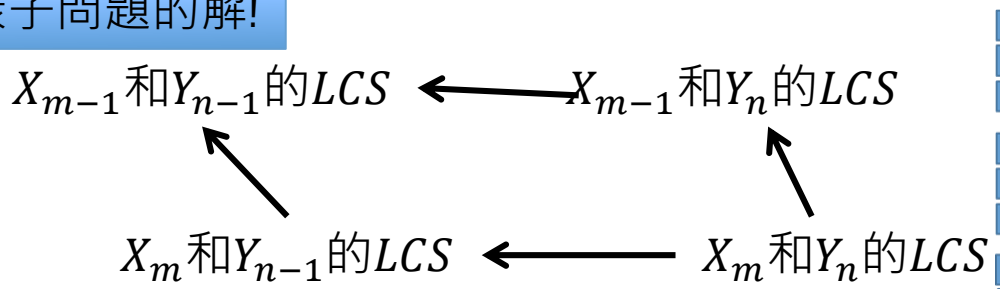


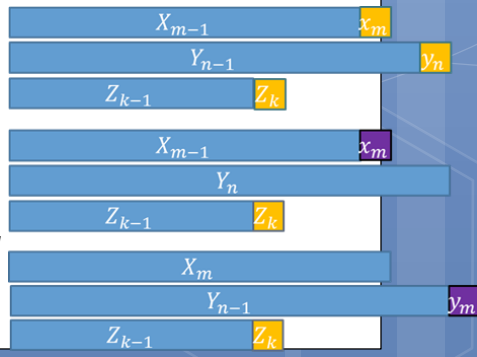
Overlapping subproblem

- 給定兩字串$X = \langle x_1, x_2, \dots, x_m \rangle, Y = \langle y_1, y_2, \dots, y_n \rangle$, 及$Z = \langle z_1, z_2, \dots, z_m \rangle$為X和Y的LCS(之一)

1. If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} 是X_{m-1}及Y_{n-1}的LCS之一
2. If $x_m \neq y_n$, then $z_k \neq x_m$表示Z是X_{m-1}及Y_n的LCS之一
3. If $x_m \neq y_n$, then $z_k \neq y_n$表示Z是X_m及Y_{n-1}的LCS之一

不同問題需要同樣子問題的解!





Dynamic Programming出招

2. 列出遞迴式子 (表示花費)

條件不同, 使用的subproblem不同

$$c[i,j] = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ c[i-1,j-1]+1 & \text{if } i,j>0 \text{ and } x_i = y_j \\ \max(c[i,j-1], c[i-1,j]) & \text{if } i,j>0 \text{ and } x_i \neq y_j \end{cases}$$

X_i and Y_j LCS的長度

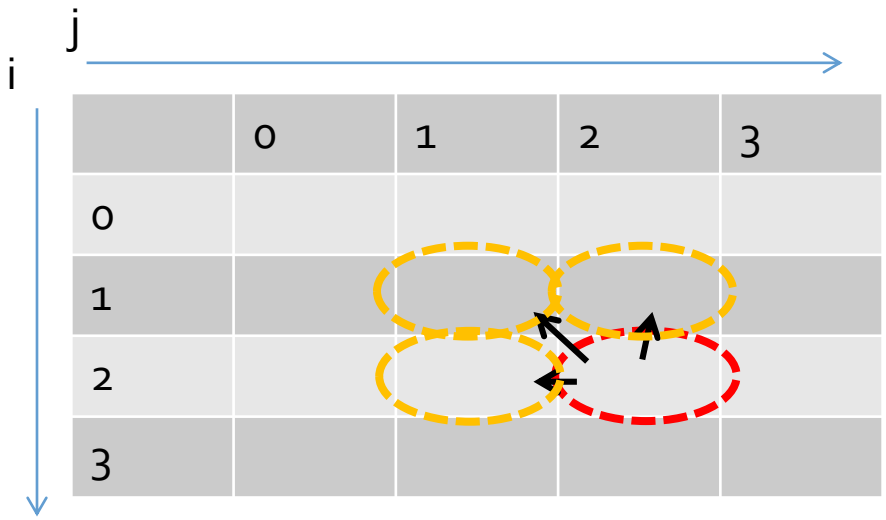
兩種選擇

$$c[i,j] = \begin{cases} 0 \\ c[i-1,j-1]+1 \\ \max(c[i,j-1], c[i-1,j]) \end{cases}$$

Dynamic Programming出招

3. 計算花費

- 使用dynamic programming填表
- 共有多少個entry? $\Theta(mn)$



每一格只用到左、
左上、上三格的資訊

使用bottom-up方法
兩層迴圈依序填入即可

例題

		B	D	C	A	B	A
		0	1	2	3	4	5	6
	0							
A	1							
B	2							
C	3							
B	4							
D	5							
A	6							
B	7							

```
LCS_Length(X,Y)
m=X.length
n=Y.length
let b[1..m,1..n] and c[0..m,0..n] be new tables
for i=1 to m
      c[i,0]=0
for j=0 to n
      c[0,j]=0
for i=1 to m
      for j=1 to n
            if x_i == y_j
                  c[i,j]=c[i-1,j-1]+1
                  b[i,j]=左上
            elseif c[i-1,j]≥c[i,j-1]
                  c[i,j]=c[i-1,j]
                  b[i,j]=上
            else
                  c[i,j]=c[i,j-1]
                  b[i,j]=左
return c and b
```

c紀錄LCS長度, b紀錄選擇結果

邊界起始值

填表: 兩層迴圈

$\Theta(mn)$

Dynamic Programming出招

4. 印出LCS結果

```
Print_LCS(b,X,i,j)
if i==0 or j==0
      return
if b[i,j]==左上
      Print_LCS(b,X,i-1,j-1)
      print x_i
elseif b[i,j]==上
      Print_LCS(b,X,i-1,j)
else
      Print_LCS(b,X,i,j-1)
```

$O(m+n)$