

# 算法设计与分析

## 第3章 动态规划 (2)

谢晓芹

哈尔滨工程大学计算机科学与技术学院

# 动态规划算法设计步骤

- 找最优子结构性质
- 递归定义最优值
- 自底向上计算最优值
- 构造最优解

最优子结构：原问题，子问题

递归⇒值

填表

递归⇒解

# 最长公共子序列

## ■ 应用

- 基于几何相似特征的石窟造像装饰图案生成方法
  - 计算机辅助设计与图形学学报, 20230816, 网络首发
- 一种基于LCS的物体碎片自动拼接方法
  - 计算机学报, 2005, 28 ( 3 )
- 带约束最长公共子序列快速算法
  - 南京大学学报(自然科学), 2009, 45 ( 5 )
  - 2010年第3期, 416~425页。
- 多维时序数据中的相似子序列搜索研究。
  - 计算机研究与发展, 2010,47(3):416-425 (国防科学技术大学计算机学院)
- Winnowing算法和动态规划算法在作业剽窃检测中的应用和比较
  - 计算机工程与科学, 2009, 31 ( 6 )

# 最长公共子序列

- 比较两个不同生物体的 DNA
  - 一个 DNA串:多个分子bases以不同的组合方式构成一个 DNA 串
    - bases (Adenine, Guanine, Cytosine [氧氨嘧啶], Thymine)
  - a strand of DNA  $\in$  finite set {A, C, G, T}
- 例如,两个生物体的 DNA
  - S1= ACCGGTCGAGTGCGCGGAAGCCGGCCGAA
  - S2 = GTCGTTCGGAATGCCGTTGCTCTGTAAA
- 研究目标：确定两个DNA序列的相似程度？

# 最长公共子序列

## ■ 如何定义S1 和S2 的相似度?

- 如果一个串S2是另一个串S1的子串，则S1和S2相似

- S1= GTCGTCGAA      S2= GTCGTCG

- 如果将一个串变换为另一个串，变换数最少

- S1= GACTAACG

- S2= GTCGTACT

- 寻找第3个串S<sub>3</sub>，S<sub>3</sub>中的所有bases都包含在S<sub>1</sub>和S<sub>2</sub>中，这些bases在S<sub>1</sub>和S<sub>2</sub>中不一定连续排列，但必须是按顺序排列的。S<sub>3</sub> 越长，则S<sub>1</sub>和S<sub>2</sub>的相似度就越大。

- 以这种相似性意义作为最长公共子序列问题的形式化定义

编辑距离Levenshtein距离

S <sub>1</sub>	B	D	C	A	B	A
S <sub>2</sub>	A	B	C	B	D	A
S <sub>3</sub>		B	C	B		A

# 最长公共子序列 ★

- 最长公共子序列 ( Longest Common Subsequence , LCS) 问题
  - 给定两个序列  $X = \langle x_1, x_2, \dots, x_m \rangle$  and  $Y = \langle y_1, y_2, \dots, y_n \rangle$ , 如何寻找  $X$  和  $Y$  的长度最大的公共子序列.
- 输入:  $X = \{x_1, x_2, \dots, x_m\}$  ,  $Y = \{y_1, y_2, \dots, y_n\}$
- 输出:  $Z = X$ 和 $Y$ 一个最长公共子序列
- 例如
  - X: A B C B D A B
  - Y: B D C A B A
  - 最长公共子序列:
    - B C B A = LCS ( X, Y )
    - B D A B, B C A B

# 最长公共子序列

- **子序列**：给定序列  $X$ ，如果存在  $X$  的**索引的一个严格增序列**  $\langle i_1, i_2, \dots, i_k \rangle$ ，使得对所有的  $j = 1, 2, \dots, k$ ，其中  $i_j \in \{1, 2, \dots, m\}$ ，且都有  $x[i_j] = z_j$ ，则称  $Z = \langle z_1, z_2, \dots, z_k \rangle$  是  $X$  的子序列
- 例如
  - $X = A, B, C, B, D, A, B$
  - $X$ 的子序列  $Z = B, C, D, B$
  - $Z$ 中元素在 $X$ 中的**索引序列**为：  
 $\langle i_1, i_2, i_3, i_4 \rangle = 2, 3, 5, 7.$
  - $Z$ 中元素的索引序列为： $\langle 1, 2, 3, 4 \rangle$

## 最长公共子序列

- 例  $X = \{A, B, C, B, D, A, B\}$
- $Z = \{B, C, D, B\}$  是子序列，相应的下标  $\{2, 3, 5, 7\}$ 。
- $W = \{C, A, B, D\}$  不是子序列，相应的下标  $\{3, 1, 2, 5\}$



# 最长公共子序列

- 最长：指子序列的元素个数最多

- 例如,

- $X = A, B, C, B, D, A, B$      $X = A, B, C, B, D, A, B$

- $Y = B, D, C, A, B, A$                        $Y = B, D, C, A, B, A$

- $Z_1 = B, C, A$                        $Z_2 = B, C, B, A$

- $Z_1$  是公共子序列，但不是a **LCS** of X and Y .

- $Z_2$  is a LCS of X and Y

- $\langle B, D, A, B \rangle, \langle B, C, A, B \rangle$  也是一个LCS

# 最长公共子序列

- $X = \langle x_1, x_2, \dots, x_m \rangle$  and  $Y = \langle y_1, y_2, \dots, y_n \rangle$
- 穷举法 ( A brute-force approach )
  - 列举出  $X$  的所有子序列，逐项核查这些子序列是否为  $Y$  的子序列
- 分析
  - 怎么检查？
    - 例如：检查BCBA 是否是BDCABA的子串？
  - 检查时间？
    - 检查每个子序列的时间： $O(n)$
    - 共有 $2^m$ 个  $X$  的子序列：一个子序列对应一个长为 $m$ 的位向量，对于  $X$  的一个索引的子集 $\{1, 2, \dots, m\}$
  - 最坏情况运行时间： $O(n2^m)$ : 指数运算时间，当序列较大时实际不可行

BDCABA

BCBA: 101011

# 最长公共子序列

## ■ 问题简化方法：

- 先找LCS的长度:  $|LCS()|$
- 再找LCS本身

最优值

最优解

## ■ 策略

- 子问题？考虑X和Y的前缀

## ■ 定义（第i 前缀）

- 设 $X=(x_1, x_2, \dots, x_m)$ 是一个序列，X的第i 前缀 $X_i$  是一个序列，定义为 $X_i=(x_1, \dots, x_i)$
- 例如： $X=(A, B, D, C, A)$ ,
  - $X_1=(A)$ ,  $X_2=(A, B)$ ,  $X_3=(A, B, D)$
  - $X_0$  是空序列

# 最长公共子序列 ★

## ■ 原问题

- 求  $X = \langle x_1, x_2, \dots, x_m \rangle$  和  $Y = \langle y_1, y_2, \dots, y_n \rangle$  的最长公共子序列

## ■ 子问题的自然分类：前缀

- $X_i = \langle x_1, x_2, \dots, x_i \rangle$ ,  $X$  的第  $i$  前缀,  $i = 0, 1, \dots, m$
- $Y_j = \langle y_1, y_2, \dots, y_j \rangle$ ,  $Y$  的第  $j$  前缀,  $j = 0, 1, \dots, n$

## ■ 问题转化为

- 求  $X$  的前缀  $X_i$  和  $Y$  的前缀  $Y_j$  的最长公共子序列

# 最长公共子序列-求解步骤

- 动态规划算法问题求解的步骤
  - 步骤1:最长公共子序列的最优子结构性质
  - 步骤2:子问题的递归结构
  - 步骤3:计算最优值
  - 步骤4:构造最优解

## 步骤1:最长公共子序列的结构

- $X = \langle x_1, x_2, \dots, x_m \rangle$  and  $Y = \langle y_1, y_2, \dots, y_n \rangle$
- LCS问题具有最优子结构属性

### ✓定理（最优子结构）

设序列 $X=\{x_1, x_2, \dots, x_m\}$ 和 $Y=\{y_1, y_2, \dots, y_n\}$ 的最长公共子序列为 $Z=\{z_1, z_2, \dots, z_k\}$ ，则

- (1) 若 $x_m=y_n$ ，则 $z_k=x_m=y_n$ ，且 $z_{k-1}$ 是 $X_{m-1}$ 和 $Y_{n-1}$ 的LCS，  
即 $\text{LCS}(X, Y) = \text{LCS}(X_{m-1}, Y_{n-1}) + \langle x_m=y_n \rangle$

例如：

$X=\text{ABECDF}$

$Y=\text{BAEDF}$

$Z=\text{BEDF}$

## 步骤1:最长公共子序列的结构

- 定理: (LCS的最优子结构) 设  $Z = \langle z_1, z_2, \dots, z_k \rangle$  是  $X = \langle x_1, x_2, \dots, x_m \rangle$  and  $Y = \langle y_1, y_2, \dots, y_n \rangle$  的LCS.
- 1. If  $x_m = y_n$ , then  $z_k = x_m = y_n$  and  $Z_{k-1}$  is an LCS of  $X_{m-1}$  and  $Y_{n-1}$ 

子问题1
- 证明 :

- 设  $z_k \neq x_m$ , 令  $Z' = \langle z_1, z_2, \dots, z_k, x_m \rangle$ , 则  $Z'$  是  $X$  和  $Y$  的**公共子序列**, 且  $\text{length}(Z') = k+1 \Rightarrow Z'$  是比  $Z$  更长的子序列  $\Rightarrow$  与题设  $Z$  是LCS矛盾
- 显然,  $Z_{k-1}$  是  $X_{m-1}$  和  $Y_{n-1}$  的 a CS,  $\text{length}(Z_{k-1}) = k-1$ . 假设  $Z_{k-1}$  不是最优, 设  $W$  是  $X_{m-1}$  和  $Y_{n-1}$  的 a CS, 且  $\text{length}(W) \geq k$ , 将  $x_m$  附加到  $W$  后面得到  $W'$ , 则  $W'$  是  $X_m$  和  $Y_n$  的 a CS, 且  $\text{length}(W') \geq k+1 \Rightarrow$  与题设  $Z$  是LCS矛盾

$$X = \langle x_1, \dots, x_i, \dots, x_m \rangle$$

$$Z = \langle z_1, \dots, z_k \rangle$$

$$Y = \langle y_1, \dots, y_j, \dots, y_n \rangle$$

# 步骤1:最长公共子序列的结构

## ✓定理（最优子结构）

设序列 $X=\{x_1, x_2, \dots, x_m\}$ 和 $Y=\{y_1, y_2, \dots, y_n\}$ 的最长公共子序列为 $Z=\{z_1, z_2, \dots, z_k\}$ ，  
则

- (1) 若 $x_m=y_n$ ，则 $z_k=x_m=y_n$ ，且 $Z_{k-1}$ 是 $X_{m-1}$ 和 $Y_{n-1}$ 的LCS，  
即 $\text{LCS}(X, Y) = \text{LCS}(X_{m-1}, Y_{n-1}) + \langle x_m=y_n \rangle$ 。
- (2) 若 $x_m \neq y_n$ 且 $z_k \neq x_m$ ，则 $Z$ 是 $X_{m-1}$ 和 $Y$ 的LCS，  
即 $\text{LCS}(X, Y) = \text{LCS}(X_{m-1}, Y)$

例如： $X=\text{ABADE}\textcolor{red}{F}$ ， $Y=\text{BAED}\textcolor{red}{D}$   $x_6=F, y_4=D, Z=\text{BAE}\textcolor{red}{E}$

可以看到：  $x_6 \neq y_4$  且  $z_3 \neq x_6$

则：  $Z=\text{BAE}$  是 $X$ 去掉最后一位即 $\textcolor{red}{X}_5=\text{ABADE}$  和 $\textcolor{red}{Y}=\text{BAED}$ 的LCS



# 步骤1:最长公共子序列的结构

## ✓定理（最优子结构）

设序列 $X=\{x_1, x_2, \dots, x_m\}$ 和 $Y=\{y_1, y_2, \dots, y_n\}$ 的最长公共子序列为 $Z=\{z_1, z_2, \dots, z_k\}$ ，  
则

- (1) 若 $x_m=y_n$ ，则 $z_k=x_m=y_n$ ，且 $Z_{k-1}$ 是 $X_{m-1}$ 和 $Y_{n-1}$ 的LCS，  
即 $\text{LCS}(X, Y) = \text{LCS}(X_{m-1}, Y_{n-1}) + \langle x_m = y_n \rangle$ 。
- (2) 若 $x_m \neq y_n$ 且 $z_k \neq x_m$ ，则 $Z$ 是 $X_{m-1}$ 和 $Y$ 的LCS，  
即 $\text{LCS}(X, Y) = \text{LCS}(X_{m-1}, Y)$
- (3) 若 $x_m \neq y_n$ 且 $z_k \neq y_n$ ，则 $Z$ 是 $X$ 和 $Y_{n-1}$ 的LCS，  
即 $\text{LCS}(X, Y) = \text{LCS}(X, Y_{n-1})$


## 步骤1:最长公共子序列的结构

- 定理: (LCS的最优子结构) 设  $Z = \langle z_1, z_2, \dots, z_k \rangle$  是  $X = \langle x_1, x_2, \dots, x_m \rangle$  and  $Y = \langle y_1, y_2, \dots, y_n \rangle$  的LCS
- 2. If  $x_m \neq y_n$ , and  $z_k \neq x_m \Rightarrow Z$  is an LCS of  $X_{m-1}$  and  $Y$  子问题2
- 3. If  $x_m \neq y_n$ , and  $z_k \neq y_n \Rightarrow Z$  is an LCS of  $X$  and  $Y_{n-1}$  子问题3
- 证明：
  - 2. 若  $z_k \neq x_m$ , 则  $Z$  是  $X_{m-1}$  和  $Y$  的 a CS. 设存在一个  $X_{m-1}$  和  $Y$  的公共子序列  $W$ , 其  $\text{length}(W) > k$ , 那么,  $W$  是  $X$  和  $Y$  的 a CS  $\Rightarrow$  与题设  $Z$  是 an LCS 矛盾
  - 3. Symmetric to 2.

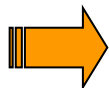
动态规划基本要素1：

最优子结构：问题的最优解包含着其子问题的最优解。

## 步骤1:最长公共子序列的结构

- LCS问题具有**最优子结构性质**: 
  - 两个序列的 an LCS 包含了这两个序列的**前缀**的 an LCS
- For example

$$\begin{aligned} X &= \langle x_1, x_2, \dots, x_m \rangle \\ Z &= \langle z_1, z_2, \dots, z_k \rangle \\ Y &= \langle y_1, y_2, \dots, y_n \rangle \end{aligned}$$



$$\begin{aligned} X_{m-1} &= \langle x_1, x_2, \dots, \underline{x_{m-1}} \rangle \\ Z_{k-1} &= \langle z_1, z_2, \dots, \underline{z_{k-1}} \rangle \\ Y_{n-1} &= \langle y_1, y_2, \dots, \underline{y_{n-1}} \rangle \end{aligned}$$

$$x_m = y_n$$

$$\begin{aligned} X_{m-1} &= \langle x_1, x_2, \dots, \underline{x_{m-1}} \rangle \\ Z &= \langle z_1, z_2, \dots, \underline{z_k} \rangle \\ Y &= \langle y_1, y_2, \dots, y_n \rangle \end{aligned}$$

$$\begin{aligned} x_m &\neq y_n \\ z_k &\neq x_m \end{aligned}$$

$$\begin{aligned} X &= \langle x_1, x_2, \dots, x_m \rangle \\ Z &= \langle z_1, z_2, \dots, \underline{z_k} \rangle \\ Y_{n-1} &= \langle y_1, y_2, \dots, \underline{y_{n-1}} \rangle \end{aligned}$$

$$\begin{aligned} x_m &\neq y_n \\ z_k &\neq y_n \end{aligned}$$

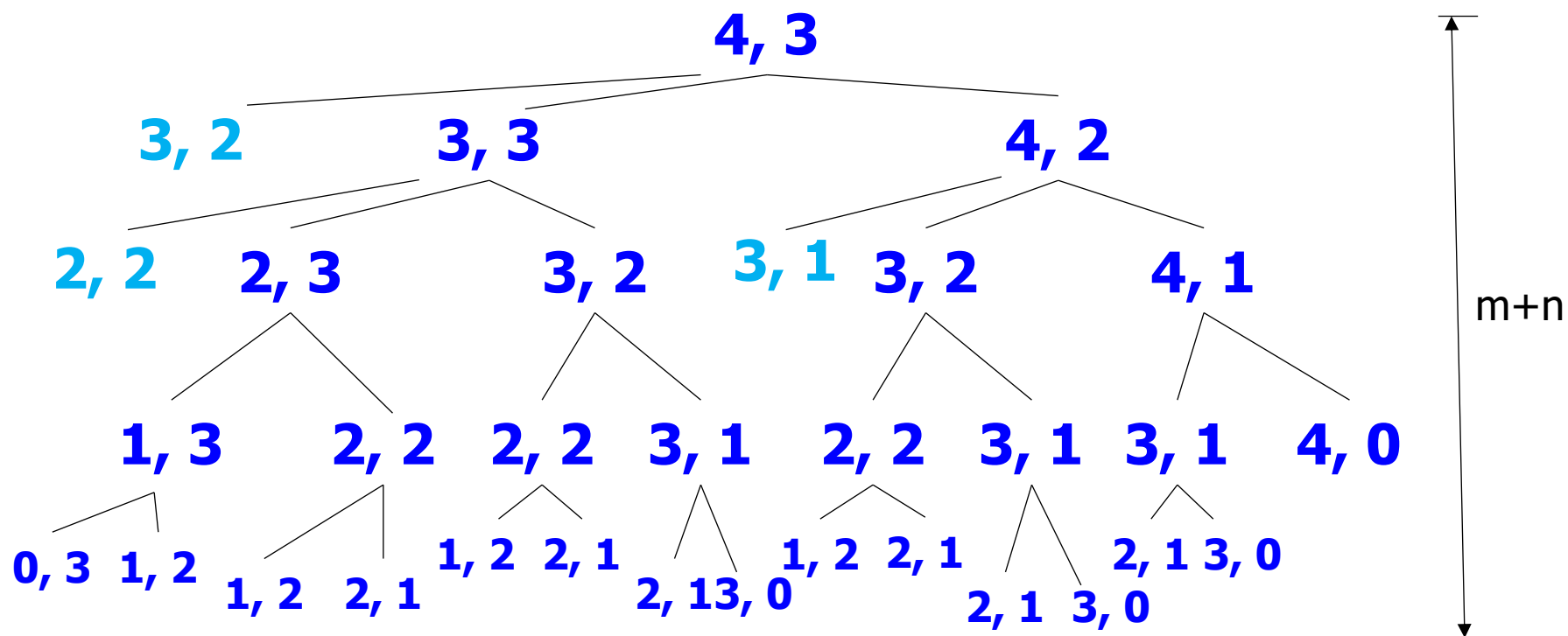
有没有子问题重叠性质?

## 步骤1:最长公共子序列的结构

- LCS问题具有重叠子问题性质 ★

■ 例如：X = ( A, B, C, D )    m=4

Y = ( A, C, D )            n=3



动态规划基本要素2：

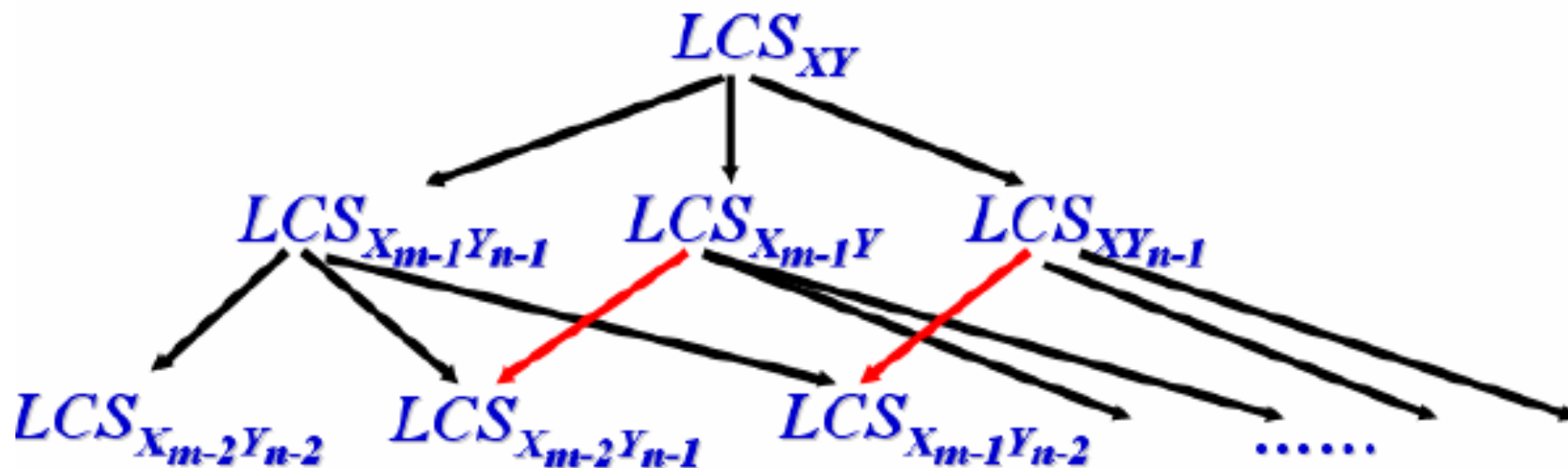
重叠子问题：递归算法自顶向下解问题时，有些子问题被反复计算多次。

# 步骤1:最长公共子序列的结构 ✨

## ✓最优解结构

- (1)  $LCS_{XY} = LCS_{X_{m-1}Y_{n-1}} + \langle x_m = y_n \rangle$  if  $x_m = y_n$
- (2)  $LCS_{XY} = LCS_{X_{m-1}Y}$  if  $x_m \neq y_n$  且  $z_k \neq x_m$
- (3)  $LCS_{XY} = LCS_{XY_{n-1}}$  if  $x_m \neq y_n$  且  $z_k \neq y_n$

## ✓重叠子问题



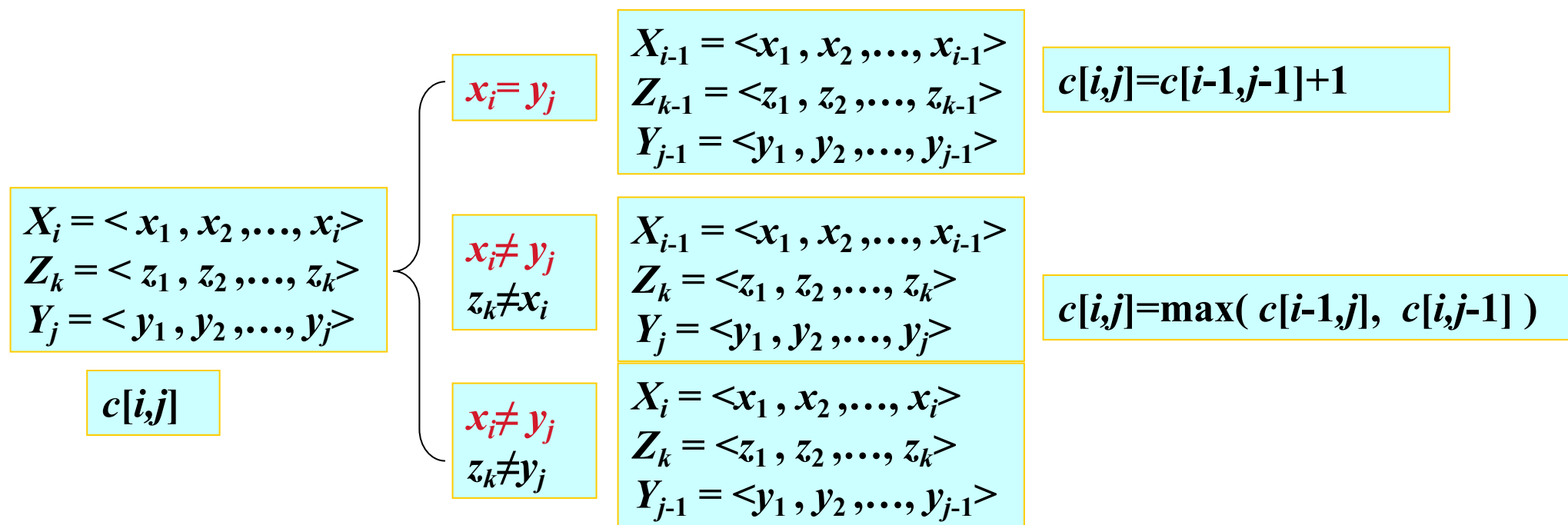
## 步骤1:最长公共子序列的结构

原问题：X和Y的最长公共子序列  $X_i, Y_j$

$x_i=y_j$	原问题	$X_i, Y_j$	$Z[1..k]$
	子问题	$X_{i-1}, Y_{j-1}$	$Z[1..k-1]$
$x_i \neq y_j$	原问题	$X_i, Y_j$	$Z[1..k]$
	子问题	$X_{i-1}, Y_j$ 或 $X_i, Y_{j-1}$	$Z[1..k]$

## 步骤2: 子问题的递归结构

- 当求 an LCS时，定理表明了有一个或两个子问题需要考虑
- 定义： $c[i, j]$ 记录序列 $X_i$ 和 $Y_j$ 的最长公共子序列的长度
  - $c[i, j] = |\text{LCS}(X_i, Y_j)|$
  - $c[i, j] = 0, i \cdot j = 0$



## 步骤2: 子问题的递归结构

### ■ LCS问题的最优子结构可导出递归公式

- $X_i = \{x_1, x_2, \dots, x_i\}$  ;  $Y_j = \{y_1, y_2, \dots, y_j\}$ 。
- 用  $c[i][j]$  记录序列  $X_i$  和  $Y_j$  的最长公共子序列的长度
- 当  $i=0$  或  $j=0$  时，空序列是  $X_i$  和  $Y_j$  的最长公共子序列,  $C[i][j]=0$ 。



$$c[i][j] = \begin{cases} 0 & i = 0, j = 0 \\ c[i-1][j-1] + 1 & i, j > 0; x_i = y_j \\ \max\{c[i][j-1], c[i-1][j]\} & i, j > 0; x_i \neq y_j \end{cases}$$



## 步骤3: 计算最优值

### ■ LCS的递归算法

- 输入： $X_i, Y_j$
- 输出：数组 $c$

$$c[i][j] = \begin{cases} 0 & i = 0, j = 0 \\ c[i-1][j-1] + 1 & i, j > 0; x_i = y_j \\ \max\{c[i][j-1], c[i-1][j]\} & i, j > 0; x_i \neq y_j \end{cases}$$

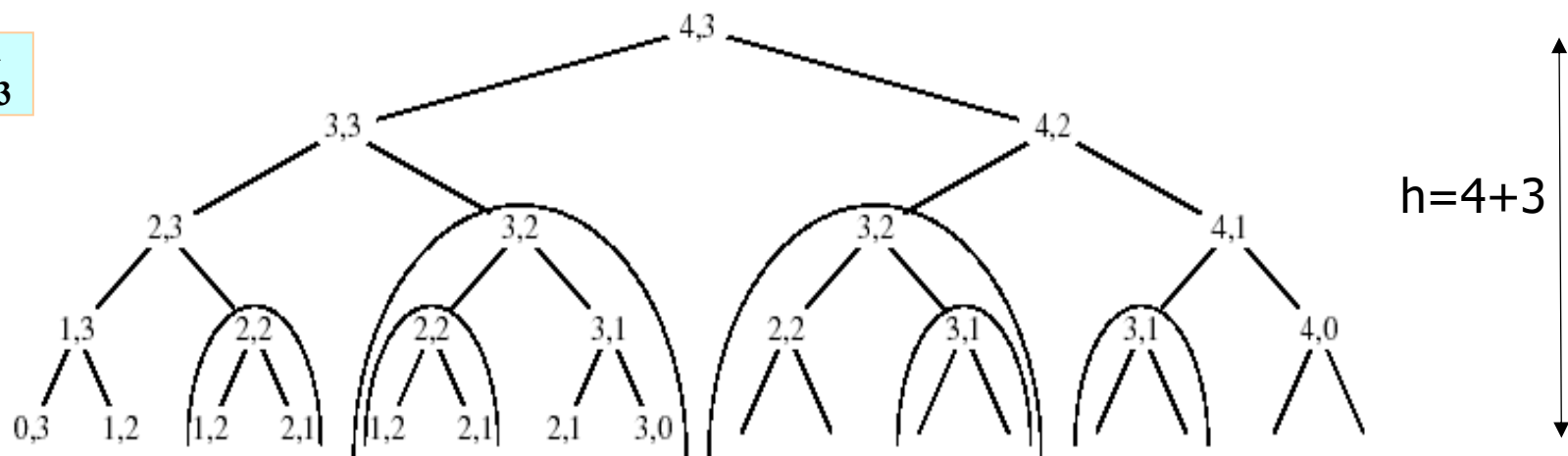
```
LCSLength(X, Y, i, j){  
    if (i=0 or j=0) c[i][j]=0;  
    if (X[i]=Y[j])  
        then c[i][j]←LCSLength(X,Y, i-1, j-1)+1;  
        else c[i][j]←max {LCSLength (X,Y,i-1,j),  
                          LCSLength (X,Y,i,j-1)};  
}
```

## 步骤3: 计算最优值 (LCS长度)

### ■ 分析

- 直接利用递归式容易写出指数级的递归算法。

例如:  $X_4, Y_3$



- 高度 =  $m+n \Rightarrow$  指数级别复杂度
- 子问题个数:  $\Theta(nm)$

动态规划基本要素2:

重叠子问题: 递归算法自顶向下解问题时, 有些子问题被反复计算多次。

## 步骤3: 计算最优值

### ■ 改进：LCS备忘录方法

- 求解完一个子问题后，把答案保存在表里，在下一次需要解此子问题时，直接到表中查表，而无需重新计算

```
LCSLength(X, Y, i, j){  
    if c[i][j]=NIL then{  
        if (i=0 or j=0) c[i][j]=0;  
        if (X[i]=Y[j])  
            then c[i][j]←LCSLength(X,Y, i-1, j-1)+1;  
            else c[i][j]←max {LCSLength (X,Y,i-1,j),  
                             LCSLength (X,Y,i,j-1)};  
    }  
    return c[i][j]  
}
```

复杂度分析:

时间复杂度 =  $\Theta(mn)$

空间复杂度 =  $\Theta(mn)$

## 步骤3: 计算最优值

### ■ LCS的动态规划算法:按自底向上方法进行求解

输入：序列X和Y；

$$X_m = \langle x_1, x_2, \dots, x_m \rangle$$
$$Y_n = \langle y_1, y_2, \dots, y_n \rangle$$

输出：数组c和b；

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

使用表b [1..m, 1..n] 构造最优解

```
LCSLength(X, Y)           // X and Y as inputs
1  m ← length[X];
2  n ← length[Y];
3  for i ← 1 to m          // Table c[0..m, 0..n] stores c[i,j],
4      do c[i, 0] ← 0; // computed in row-major order.
5  for j ← 0 to n
6      do c[0, j] ← 0;
7  for i ← 1 to m
8      do for j ← 1 to n
9          do if xi = yj
10             then c[i, j] ← c[i-1, j-1] + 1;
11                 b[i, j] ← “↖”;
12             else if c[i-1, j] ≥ c[i, j-1]
13                 then {c[i, j] ← c[i-1, j]; //use Xi-1, Yj
14                     b[i, j] ← “↑”;}
15             else {c[i, j] ← c[i, j-1];
16                 b[i, j] ← “←”;}
17  return c and b
```

### 步骤3: 计算最优值 (LCS长度)

$$c[i][j] = \begin{cases} 0 & i = 0, j = 0 \\ c[i-1][j-1] + 1 & i, j > 0; x_i = y_j \\ \max\{c[i][j-1], c[i-1][j]\} & i, j > 0; x_i \neq y_j \end{cases}$$

c[3,0]

c[3,1]

c[3,2]

c[3,3]

c[3,4]

c[2,0]

c[2,1]

c[2,2]

c[2,3]

c[2,4]

c[1,0]

c[1,1]

c[1,2]

c[1,3]

c[1,4]

c[0,0]

c[0,1]

c[0,2]

c[0,3]

c[0,4]

## 步骤3: 计算最优值 (LCS长度)

$$X_i = \langle x_1, x_2, \dots, x_i \rangle$$
$$Y_j = \langle y_1, y_2, \dots, y_j \rangle$$
$$c[i, j] =$$
$$\begin{cases} 0 & (\text{if } i = 0 \text{ or } j = 0), \\ c[i-1, j-1] + 1, & (\text{if } i, j > 0 \text{ and } x_i = y_j), \\ \max(c[i-1, j], c[i, j-1]), & (\text{if } i, j > 0 \text{ and } x_i \neq y_j). \end{cases}$$

		$j$	0	1	2	3	4	5	6
		$y_j$		B	D	C	A	B	A
$i$	$x_i$								
0			0	0	0	0	0	0	0
1	A		0	0	0	0	1	1	1
2	B		0	1	1	1	1	2	2
3	C		0	1	1	2	2	2	2
4	B		0	1	1	2	2	3	3
5	D		0	1	2	2	2	3	3
6	A		0	1	2	2	3	3	4
7	B		0	1	2	2	3	4	4

$$\text{LCSLength}(X, Y)$$
//  $X$  and  $Y$  as inputs1  $m \leftarrow \text{length}[X]$ 2  $n \leftarrow \text{length}[Y]$ 3 **for**  $i \leftarrow 1$  **to**  $m$  // Table  $c[0..m, 0..n]$  stores  $c[i, j]$ ,4 **do**  $c[i, 0] \leftarrow 0$  // computed in row-major order.5 **for**  $j \leftarrow 0$  **to**  $n$ 6 **do**  $c[0, j] \leftarrow 0$ 7 **for**  $i \leftarrow 1$  **to**  $m$ 8 **do for**  $j \leftarrow 1$  **to**  $n$ 9 **do if**  $x_i = y_j$ 10 **then**  $c[i, j] \leftarrow c[i-1, j-1] + 1$ 11  $b[i, j] \leftarrow \nwarrow$ 12 **else if**  $c[i-1, j] \geq c[i, j-1]$ 13 **then**  $c[i, j] \leftarrow c[i-1, j]$  //use  $X_{i-1}, Y_j$ 14  $b[i, j] \leftarrow \uparrow$ 15 **else**  $c[i, j] \leftarrow c[i, j-1]$  //use  $X_i, Y_{j-1}$ 16  $b[i, j] \leftarrow \leftarrow$ 17 **return**  $c$  and  $b$ 

打印输出: BCBA

## 步骤3: 计算最优值 (LCS长度)

```
void LCSLength(int m, int n, char *x, char *y,
int **c, int **b){
    int i, j;
    for (i = 1; i <= m; i++) c[i][0] = 0;
    for (i = 1; i <= n; i++) c[0][i] = 0;
    for (i = 1; i <= m; i++)
        for (j = 1; j <= n; j++) {
            if (x[i]==y[j]) {
                c[i][j]=c[i-1][j-1]+1;
                b[i][j]= “↖”;}
            else if (c[i-1][j]>=c[i][j-1]) {
                c[i][j]=c[i-1][j]; b[i][j]=“↑”;}
            else {c[i][j]=c[i][j-1]; b[i][j]= “←”;}
        }
}
```

### 算法复杂度分析:

算法的计算时间上界为  $O(mn)$ 。

算法所占用的空间显然为  $O(mn)$ 。

### 步骤3: 计算最优值 (LCS长度)

- 由于每个数组单元的计算耗费 $O(1)$ 时间，LCSLength算法耗时 $O(mn)$
- 使用b表( $b[i, j]$ )来重构 an LCS 的元素，路径用阴影或连线标注

		<i>j</i>						
		0	1	2	3	4	5	6
<i>i</i>	$y_j$	<i>B</i>	<i>D</i>	<i>C</i>	<i>A</i>	<i>B</i>	<i>A</i>	
0	$x_i$	0	0	0	0	0	0	0
1	<i>A</i>	0	↑	↑	↑	↖	←	↖
2	<i>B</i>	0	↖	←	←	↑	↖	←
3	<i>C</i>	0	↑	↑	↖	←	↑	↑
4	<i>B</i>	0	↖	↑	↑	↑	↖	←
5	<i>D</i>	0	↑	↖	↑	↑	↑	↑
6	<i>A</i>	0	↑	↑	↑	↖	↑	↖
7	<i>B</i>	0	↖	↑	↑	↑	↖	↑

		a	m	p	u	t	a	t	i	o	n
s	0	0	0	0	0	0	0	0	0	0	0
p	0	0	0	1	1	1	1	1	1	1	1
a	0	1	1	1	1	1	2	2	2	2	2
n	0	1	1	1	1	1	2	2	2	2	3
k	0	1	1	1	1	1	2	2	2	2	3
i	0	1	1	1	1	1	2	2	3	3	3
n	0	1	1	1	1	1	2	2	3	3	4
g	0	1	1	1	1	1	2	2	3	3	4
				p			a			i	n



## 步骤4：构造最优解

- 初始调用是  $\text{LCS}(b, X, m, n)$ 
  - 只要在  $b[i, j]$  中遇到 “↖”，表示  $x_i = y_j$  是 LCS 的一个元素
  - 每次递归时， $i$  和  $j$  至少有一个减值，算法的运行时间为  $O(m+n)$

		$j$						
		0	1	2	3	4	5	6
$i$	$y_j$		B	D	C	A	B	A
0	$x_i$	0	0	0	0	0	0	0
1	A	0	↑	↑	↑	↖	←	↖
2	B	0	↖	←	←	↑	↖	←
3	C	0	↑	↑	↖	←	↑	↑
4	B	0	↖	↑	↑	↑	↖	←
5	D	0	↑	↖	↑	↑	↑	↑
6	A	0	↑	↑	↑	↖	↑	↖
7	B	0	↖	↑	↑	↑	↖	↑

```
LCS( $b, X, i, j$ )
1  if  $i=0$  or  $j=0$ 
2    then return
3  if  $b[i, j] = \text{“}\nwarrow\text{”}$ 
4    then LCS( $b, X, i-1, j-1$ )
5        print  $x_i$ 
6  else if  $b[i, j] = \text{“}\uparrow\text{”}$ 
7        then LCS( $b, X, i-1, j$ )
8        else LCS( $b, X, i, j-1$ )
```

打印输出为：BCBA

# 算法的改进

- 给定一个算法，可以考虑改进其时间和空间开销
- 一些改变能简化代码，改进其常数系数，但不能改进渐近性能
  - 例如，在重构an LCS 可以仅使用表  $c$ ，而去掉 $b$ 表格。每一个  $c[i, j]$ 只依赖:  $c[i-1, j-1]$ ,  $c[i-1, j]$ , and  $c[i, j-1]$ 。给定 $c[i, j]$ 的值, 可以用 $O(1)$ 时间来确定是哪三个值用来计算 的 $c[i, j]$ , 而无需去检查表 $b$
  - 不使用表  $b$  可以节省 $\Theta(mn)$ 的空间开销，但算法的空间开销不会渐近减少，因为表  $c$  需要 $\Theta(mn)$ 的存储空间
- 一些算法能产生实质性的、在时间和空间上的渐近性能的提高
  - 可以减少LCS-LENGTH的空间的渐近开销。因为每次计算 $c[i, j]$ 时仅需要表  $c$  的两行，正在计算的行和上一行。甚至可以仅使用比表  $c$  的一行稍多一点的空间来计算  $c$
  - 如果我们仅要求 an LCS 的长度时，上述改进的算法有效。若需要重构 an LCS 的每个元素，上述改进算法不能保留足够的信息

# 最长公共子序列-小结

- 理解动态规划算法的概念
- 掌握动态规划算法的基本要素
  - 最优子结构性质
  - 重叠子问题性质
- 掌握设计动态规划算法的步骤
  - (1) 找出最优解的性质，并刻画其结构特征。
  - (2) 递归地定义最优值。
  - (3) 以自底向上的方式计算出最优值。
  - (4) 根据计算最优值时得到的信息，构造最优解。
- 重点难点
  - 基本要素和步骤

## 思考题

- 假设准备开始一次长途旅行。以0公里作为起点，一路上共有 $n$ 座旅店，距离起点的公里数分别为 $a_1 < a_2 < \dots < a_n$ 。旅途中，您只能在这些旅店停留，在哪里停留完全由您决定。最后一座旅店（ $a_n$ ）是您的终点。
- 理想情况下，您每天可以行进200公里，不过考虑到旅店间的实际距离，有时候可能达不到这么远。如果您某天走了 $x$ 公里，那么您将受到 $(200-x)^2$ 的惩罚。您需要计划好行程，以使得总的惩罚也即每天所受惩罚的总和最小。请给出一个高效的算法，用于确定一路上最优的停留位置序列。

## 思考题

- 给你一张里面 $n \times n$ 个格子组成的二维表格，每个格子里有一个正整数。让你从左上角的格子出发，只能**向下**或**向右**，走到右下角的格子，路径上经过的数字之和作为收益。如何最大化你的收益？请1) 给出问题的最优解的递归表达式。 2) 设计一个动态规划算法求解该问题，并分析算法的时间和空间复杂性。
  - 输入：二维数组reward[n][n]

100	200	200	200
300	100	100	200
200	500	600	400
100	400	500	800