Design and Analysis of Algorithms

Part II: Dynamic Programming

Lecture 12: Longest Common Subsequence

童咏昕

北京航空航天大学
计算机学院

动态规划篇概述

- **在算法课程第二部分“动态规划”主题中，我们将主要聚焦于如下经典问题：**
  - 0-1 Knapsack (0-1背包问题)
  - Maximum Contiguous Subarray II (最大连续子数组 II)
  - Longest Common Subsequences (最长公共子序列)
  - Longest Common Substrings (最长公共子串)
  - Minimum Edit Distance (最小编辑距离)
  - Rod-Cutting (钢条切割)
  - Chain Matrix Multiplication (矩阵链乘法)

问题背景：子序列

- 子序列
  - 将给定序列中零个或多个元素（如字符）去掉后所得结果
- 示例
  - 给定序列X

X	A	B	C	B	D	A	B

  - X的子序列

X_1	A	B	C	B	D	A	B

问题背景：子序列

- 子序列
  - 将给定序列中零个或多个元素（如字符）去掉后所得结果
- 示例
  - 给定序列X

X	A	B	C	B	D	A	B

  - X的子序列

X_1	A	B	C	B	D	A	B

X_2	A	B	C	B

问题背景：子序列

- 子序列
  - 将给定序列中零个或多个元素（如字符）去掉后所得结果
- 示例
  - 给定序列X

X	A	B	C	B	D	A	B

  - X的子序列

X_1	A	B	C	B	D	A	B

X_2	A	B	C	B

X_3	A	C	B	B

问题背景：公共子序列

- 给定两个序列X和Y

X	A	B	C	B	D	A	B

Y	B	D	C	A	B	A

- 公共子序列示例

X_1	C	A

Y_1	C	A

X_2	A	B	A

Y_2	A	B	A

X_3	B	C	A	B

Y_3	B	C	A	B

问题：如何求两个给定序列的最长公共子序列?

问题定义

- 形式化定义

最长公共子序列问题

Longest Common Subsequence Problem

输入

- 序列$X = < x_1, x_2, \dots, x_n >$和序列$Y = < y_1, y_2, \dots, y_m >$

输出

- 求解一个公共子序列$Z = < z_1, z_2, \dots, z_l >$，令

$$\max |Z|$$

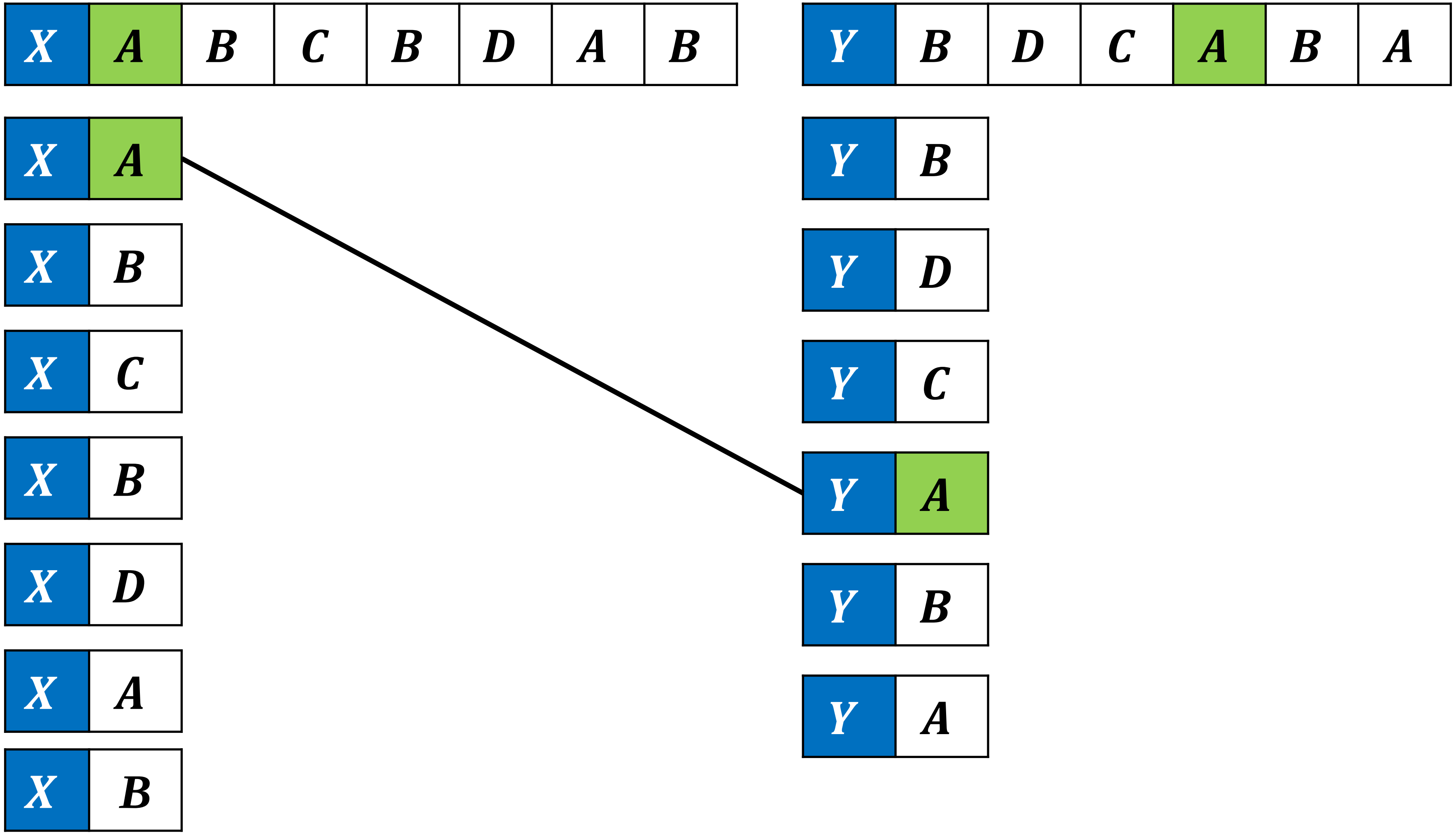
优化目标

$$s.t. < z_1, z_2, \dots, z_l > = < x_{i_1}, x_{i_2}, \dots, x_{i_l} > = < y_{j_1}, y_{j_2}, \dots, y_{j_l} >$$
$$(1 \le i_1 < i_2, \dots, i_l \le n; 1 \le j_1 < j_2, \dots, j_l \le m)$$

约束条件

蛮力枚举

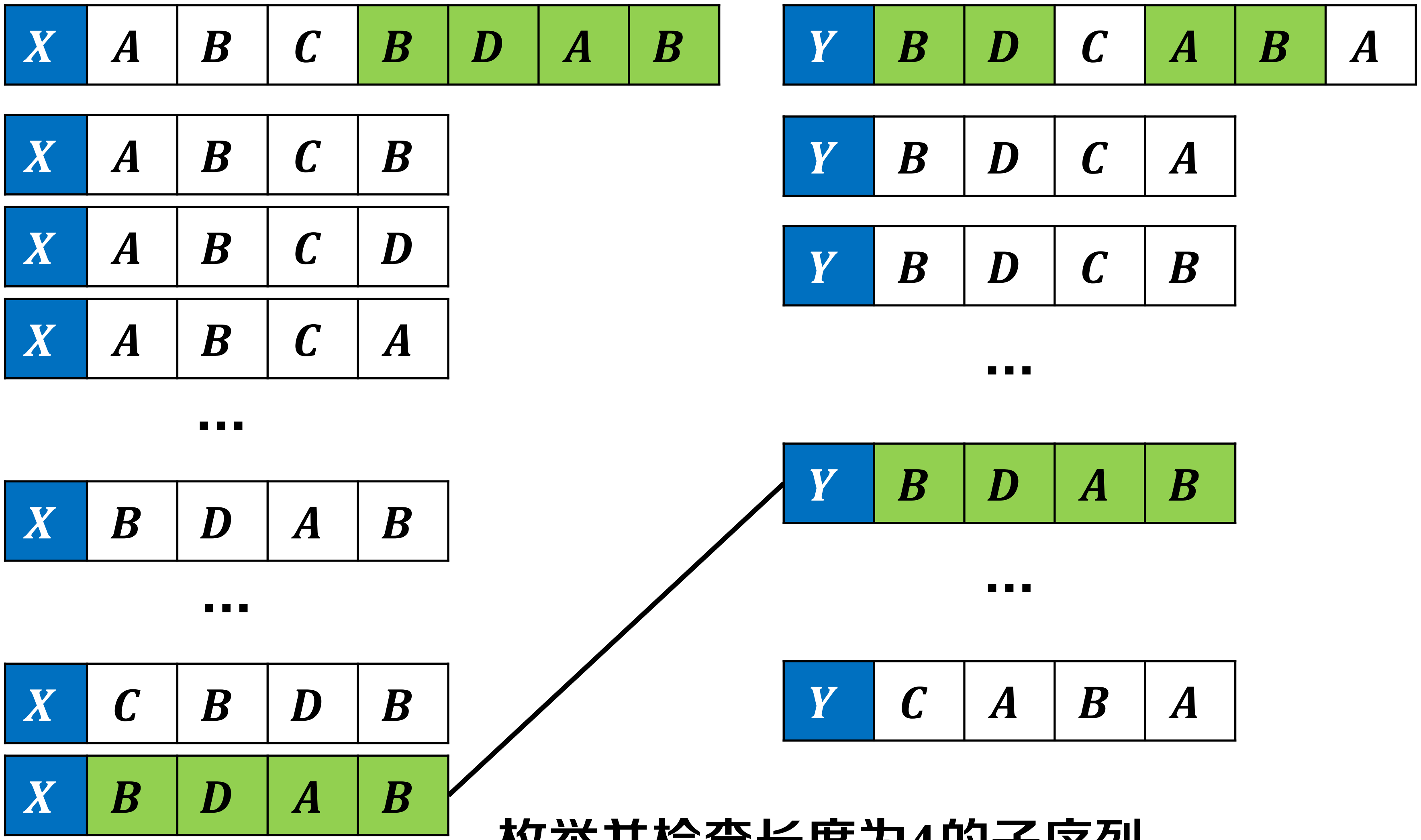
- 枚举所有子序列



枚举并检查长度为1的子序列

蛮力枚举

- 枚举所有子序列



枚举并检查长度为4的子序列

蛮力枚举

- 枚举所有子序列

X	Y	
X *A* *B* *C* *B* *D* *A* *B*	*Y* *B* *D* *C* *A* *B* *A*	
X *B*	*Y* *B*	长度为1
X *A* *B*	*Y* *A* *B*	长度为2
X *A* *B* *A*	*Y* *A* *B* *A*	长度为3
X *B* *D* *A* *B*	*Y* *B* *D* *A* *B*	长度为4
X	*Y*	长度为5
X	*Y*	长度为6

蛮力枚举

- 枚举所有子序列

X	*A*	*B*	*C*	*B*	*D*	*A*	*B*

Y	*B*	*D*	*C*	*A*	*B*	*A*

X	*B*	*Y*	*B*	长度为1

X	*A*	*B*	*Y*	*A*	*B*	长度为2

X	*A*	*B*	*A*	*Y*	*A*	*B*	*A*	长度为3

X	*B*	*D*	*A*	*B*	*Y*	*B*	*D*	*A*	*B*	长度为4

最长公共子序列

枚举观察

X	B	D	A	B
X	D	A	B	
X	A	B		
X	B			

Y	B	D	A	B	长度为4
Y	D	A	B		长度为3
Y	A	B			长度为2
Y	B				长度为1

- 可能存在**最优子结构**和**重叠子问题**

问题：如何利用动态规划求解？

问题结构分析

- **给出问题表示**
  - $C[i,j]$：$X[1..i]$和$Y[1..j]$的最长公共子序列长度

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

- **明确原始问题**
  - $C[n,m]$： $X[1..n]$和$Y[1..m]$的最长公共子序列长度

问题结构分析

↓

递推关系建立

↓

自底向上计算

↓

最优方案追踪

递推关系建立：分析最优（子）结构

- 考察末尾字符
  - 情况1：$x_7 \neq y_6$

X	*A*	*B*	*C*	*B*	*D*	*A*	*B*

Y	*B*	*D*	*C*	*A*	*B*	*A*

  - 情况2：$x_7 = y_6$

X	*A*	*B*	*C*	*B*	*D*	*A*	*B*

Y	*B*	*D*	*C*	*A*	*B*

递推关系建立：分析最优（子）结构

- 考察末尾字符
  - 情况1：$x_7 \neq y_6$

$C[7,6]$

X	A	B	C	B	D	A	B

Y	B	D	C	A	B	A

max

$C[7,6-1]+0$

X	A	B	C	B	D	A	B

Y	B	D	C	A	B	A

$C[7-1,6]+0$

X	A	B	C	B	D	A	B

Y	B	D	C	A	B	A

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

递推关系建立：分析最优（子）结构

- $x_i \neq y_j$

$C[i,j]$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

max

$C[i,j-1]+0$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

$C[i-1,j]+0$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

- $\boxed{C[i,j]} = \max\{\boxed{C[i-1,j], C[i,j-1]}\}$

最优子结构

问题结构分析

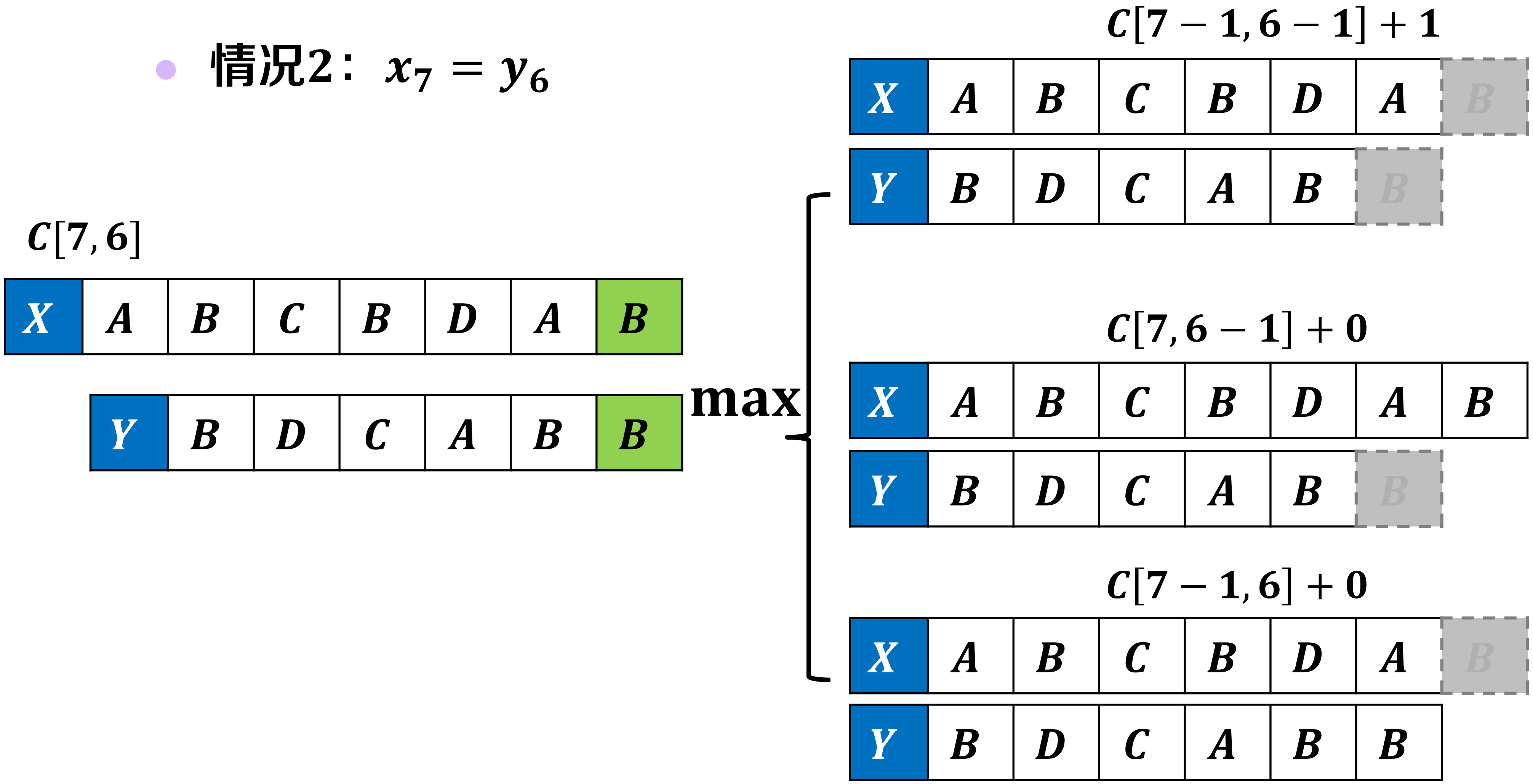
递推关系建立

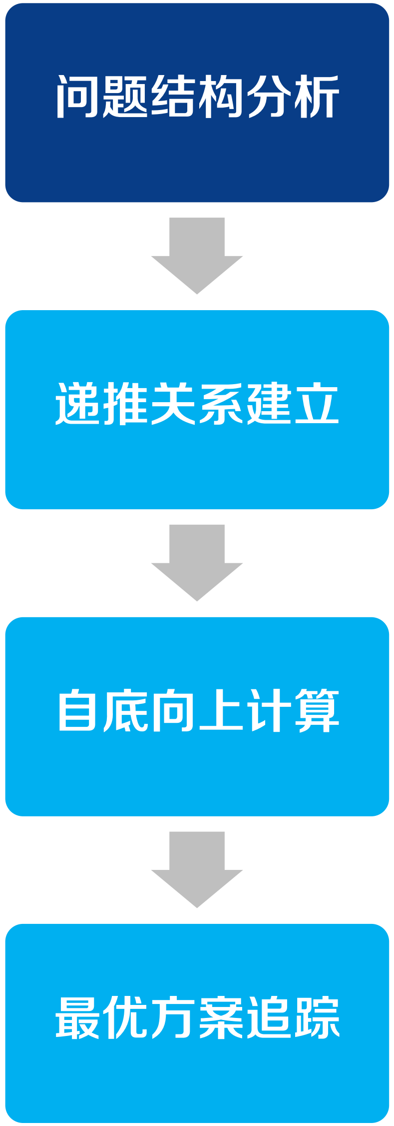
自底向上计算

最优方案追踪

递推关系建立：分析最优（子）结构

- 考察末尾字符
  - 情况2：$x_7 = y_6$





递推关系建立：分析最优（子）结构

- $x_i = y_j$

$C[i,j]$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

max

$C[i-1,j-1]+1$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

$C[i-1,j]+0$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

$C[i,j-1]+0$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

递推关系建立：分析最优（子）结构

- $x_i = y_j$

$C[i,j]$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

max

$C[i-1,j-1]+1$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

$C[i-1,j]+0$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

$C[i,j-1]+0$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

问题：3个问题是否都需要求解？

- 问题结构分析
- **递推关系建立**
- 自底向上计算
- 最优方案追踪

递推关系建立：分析最优（子）结构

- $x_i = y_j$
  - $C[i-1,j]$比$C[i-1,j-1]$至多大1
  - $C[i,j-1]$比$C[i-1,j-1]$至多大1
  - $C[i-1,j-1]+1$，另外两个+0

$C[i,j]$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

max

$C[i-1,j-1]+1$ 已充分

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

$C[i-1,j]+0$ 非必要

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

$C[i,j-1]+0$ 非必要

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

$$C[i-1,j-1]+1 \geq \max\{C[i,j-1], C[i-1,j]\}$$

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

递推关系建立：分析最优（子）结构

- $x_i = y_j$

$C[i,j]$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

$C[i-1,j-1]+1$

X	x_1	x_2	…	x_{i-1}	x_i
Y	y_1	y_2	…	y_{j-1}	y_j

- $\boxed{C[i,j]} = \boxed{C[i-1,j-1]} + 1$

最优子结构

问题结构分析

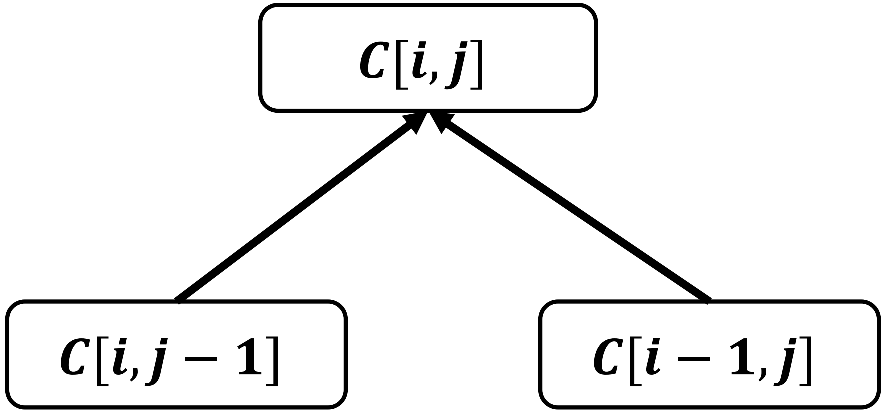
递推关系建立

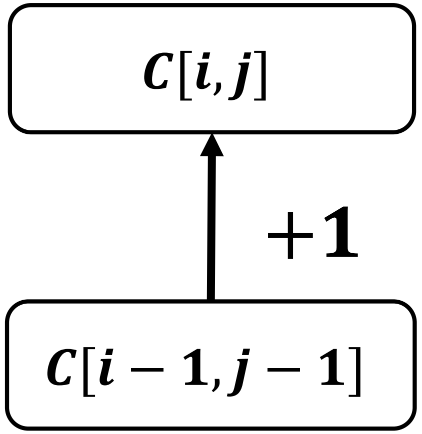
自底向上计算

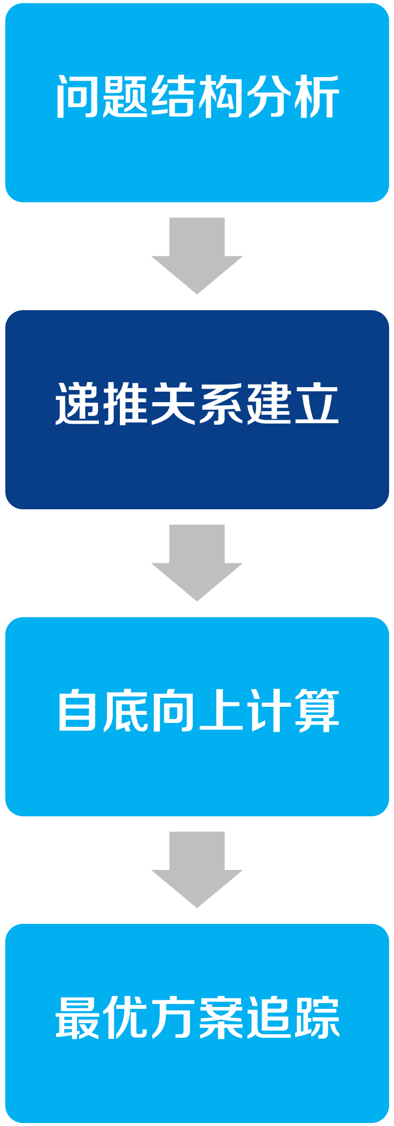
最优方案追踪

递推关系建立：构造递推公式

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







自底向上计算：确定计算顺序

- 初始化
  - $C[i,0] = C[0,j] = 0$
    - 某序列长度为0时，最长公共子序列长度为0

$C[i,j]$	$j=0$	$j=1$	$j=2$	…	$j=m$
$i=0$	0	0	0	0	0
$i=1$	0				
$i=2$	0				
…	0				
$i=n$	0				

初始化

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

自底向上计算：确定计算顺序

- 初始化
  - $C[i,0] = C[0,j] = 0$
    - 某序列长度为0时，最长公共子序列长度为0
- 递推公式
  - $$C[i,j] = \begin{cases} \max\{C[i-1,j], C[i,j-1]\}, & x_i \neq y_j \\ C[i-1,j-1]+1 & , x_i = y_j \end{cases}$$

$C[i,j]$	$j=0$	$j=1$	$j=2$	…	$j=m$
$i=0$	0	0	0	0	0
$i=1$	0				
$i=2$	0				
…	0			$C[i,j]$	
$i=n$	0				

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

自底向上计算：依次求解问题

- 初始化
  - $C[i,0] = C[0,j] = 0$
    - 某序列长度为0时，最长公共子序列长度为0
- 递推公式
  - $C[i,j] = \begin{cases} \max\{C[i-1,j], C[i,j-1]\}, & x_i \neq y_j \\ C[i-1,j-1]+1 & , x_i = y_j \end{cases}$

$C[i,j]$	$j=0$	$j=1$	$j=2$	…	$j=m$
$i=0$	0	0	0	0	0
$i=1$	0				
$i=2$	0				
…	0				
$i=n$	0				★

自底向上计算

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

最优方案追踪：记录决策过程

- 构造追踪数组$rec[1..n]$，记录子问题来源

  - $$rec[i,j]=\begin{cases} LU, & if\ C[i,j]=C[i-1,j-1]+1 \\ U, & if\ C[i,j]=C[i-1,j] \\ L, & if\ C[i,j]=C[i,j-1] \end{cases}$$

$C[i,j]$	$j=0$	$j=1$	$j=2$	$\ldots$	$j=m$
$i=0$					
$i=1$					
$i=2$					
$\ldots$				$C[i,j]$	
$i=n$					

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

最优方案追踪：输出最优方案

- 输出最长公共子序列

  - $$rec[i,j]=\begin{cases} LU, & if\ C[i,j]=C[i-1,j-1]+1 \\ U, & if\ C[i,j]=C[i-1,j] \\ L, & if\ C[i,j]=C[i,j-1] \end{cases}$$

$C[i,j]$	$j=0$	$j=1$	$j=2$	...	$j=m$
$i=0$					
$i=1$					
$i=2$				$rec[\,]=LU$	
...					
$i=n$			$rec[\,]=U$		

$rec[\,]=L$

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

算法实例

	1	2	3	4	5	6	7
X_i	A	B	C	B	D	A	B
Y_j	B	D	C	A	B	A	

$C[\,]$

$i \backslash j$	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							
7							

$rec[\,]$

$i \backslash j$	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						
7						

算法实例

	1	2	3	4	5	6	7
X_i	A	B	C	B	D	A	B
Y_j	B	D	C	A	B	A	

$C[\,]$

$i \backslash j$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0						
2	0						
3	0						
4	0						
5	0						
6	0						
7	0						

初始化

$rec[\,]$

$i \backslash j$	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						
7						

算法实例

	1	2	3	4	5	6	7
X_i	A	B	C	B	D	A	B
Y_j	B	D	C	A	B	A	

$X_i \neq Y_j$

$C[\,]$

$i \backslash j$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0						
2	0						
3	0						
4	0						
5	0						
6	0						
7	0						

$\mathrm{C}[1,1] = \max\{C[1,0], C[0,1]\}$

$rec[\,]$

$i \backslash j$	1	2	3	4	5	6
1	U					
2						
3						
4						
5						
6						
7						

算法实例

	1	2	3	4	5	6	7
X_i	A	B	C	B	D	A	B
Y_j	B	D	C	A	B	A	

$X_i = Y_j$

$C[\,]$

$i \backslash j$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1		
2	0						
3	0						
4	0						
5	0						
6	0						
7	0						

$C[1,4] = C[0,3] + 1$

$rec[\,]$

$i \backslash j$	1	2	3	4	5	6
1	U	U	U	LU		
2						
3						
4						
5						
6						
7						

算法实例

	1	2	3	4	5	6	7
X_i	A	B	C	B	D	A	B
Y_j	B	D	C	A	B	A	

$C[\,]$

i \ j	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	1	1	1	1	2	2
3	0	1	1	2	2	2	2
4	0	1	1	2	2	3	3
5	0	1	2	2	2	3	3
6	0	1	2	2	3	3	4
7	0	1	2	2	3	4	4

$rec[\,]$

i \ j	1	2	3	4	5	6
1	U	U	U	LU	L	LU
2	LU	L	L	U	LU	L
3	U	U	LU	L	U	U
4	LU	U	U	U	LU	L
5	U	LU	U	U	U	U
6	U	U	U	LU	U	LU
[illegible]	[illegible]	[illegible]	U	U	LU	U

最长公共子序列的长度

算法实例

	1	2	3	4	5	6	7
X_i	*A*	*B*	*C*	*B*	*D*	*A*	*B*
Y_j	*B*	*D*	*C*	*A*	*B*	*A*	

$C[\,]$

i \ j	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	1	1	1	1	2	2
3	0	1	1	2	2	2	2
4	0	1	1	2	2	3	3
5	0	1	2	2	2	3	3
6	0	1	2	2	3	3	4
7	0	1	2	2	3	4	4

$rec[\,]$

i \ j	1	2	3	4	5	6
1	U	U	U	LU	L	LU
2	LU	L	L	U	LU	L
3	U	U	LU	L	U	U
4	LU	U	U	U	LU	L
5	U	LU	U	U	U	U
6	U	U	U	LU	U	LU
7	LU	U	U	U	LU	U

算法实例

	1	2	3	4	5	6	7
X_i	*A*	*B*	*C*	*B*	*D*	*A*	*B*
Y_j	*B*	*D*	*C*	*A*	*B*	*A*	

			A

$C[\,]$

i \ j	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	1	1	1	1	2	2
3	0	1	1	2	2	2	2
4	0	1	1	2	2	3	3
5	0	1	2	2	2	3	3
6	0	1	2	2	3	3	4
7	0	1	2	2	3	4	4

$rec[\,]$

i \ j	1	2	3	4	5	6
1	U	U	U	LU	L	LU
2	LU	L	L	U	LU	L
3	U	U	LU	L	U	U
4	LU	U	U	U	LU	L
5	U	LU	U	U	U	U
6	U	U	U	LU	U	LU
7	LU	U	U	U	LU	U

算法实例

	1	2	3	4	5	6	7
X_i	*A*	*B*	*C*	*B*	*D*	*A*	*B*
Y_j	*B*	*D*	*C*	*A*	*B*	*A*	

			A

$C[\,]$

i \ j	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	1	1	1	1	2	2
3	0	1	1	2	2	2	2
4	0	1	1	2	2	3	3
5	0	1	2	2	2	3	3
6	0	1	2	2	3	3	4
7	0	1	2	2	3	4	4

$rec[\,]$

i \ j	1	2	3	4	5	6
1	U	U	U	LU	L	LU
2	LU	L	L	U	LU	L
3	U	U	LU	L	U	U
4	LU	U	U	U	LU	L
5	U	LU	U	U	U	U
6	U	U	U	LU	U	LU
7	LU	U	U	U	LU	U

算法实例

	1	2	3	4	5	6	7
X_i	*A*	*B*	*C*	*B*	*D*	*A*	*B*
Y_j	*B*	*D*	*C*	*A*	*B*	*A*	

		B	A

$C[\,]$

$i \backslash j$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	1	1	1	1	2	2
3	0	1	1	2	2	2	2
4	0	1	1	2	2	3	3
5	0	1	2	2	2	3	3
6	0	1	2	2	3	3	4
7	0	1	2	2	3	4	4

$rec[\,]$

$i \backslash j$	1	2	3	4	5	6
1	U	U	U	LU	L	LU
2	LU	L	L	U	LU	L
3	U	U	LU	L	U	U
4	LU	U	U	U	LU	L
5	U	LU	U	U	U	U
6	U	U	U	LU	U	LU
7	LU	U	U	U	LU	U

算法实例

	1	2	3	4	5	6	7
X_i	*A*	*B*	*C*	*B*	*D*	*A*	*B*
Y_j	*B*	*D*	*C*	*A*	*B*	*A*	

		B	A

$C[\,]$

i \ j	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	1	1	1	1	2	2
3	0	1	1	2	2	2	2
4	0	1	1	2	2	3	3
5	0	1	2	2	2	3	3
6	0	1	2	2	3	3	4
7	0	1	2	2	3	4	4

$rec[\,]$

i \ j	1	2	3	4	5	6
1	U	U	U	LU	L	LU
2	LU	L	L	U	LU	L
3	U	U	LU	L	U	U
4	LU	U	U	U	LU	L
5	U	LU	U	U	U	U
6	U	U	U	LU	U	LU
7	LU	U	U	U	LU	U

算法实例

	1	2	3	4	5	6	7
X_i	*A*	*B*	*C*	*B*	*D*	*A*	*B*
Y_j	*B*	*D*	*C*	*A*	*B*	*A*	

	C	B	A

$C[\,]$

$i \backslash j$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	1	1	1	1	2	2
3	0	1	1	2	2	2	2
4	0	1	1	2	2	3	3
5	0	1	2	2	2	3	3
6	0	1	2	2	3	3	4
7	0	1	2	2	3	4	4

$rec[\,]$

$i \backslash j$	1	2	3	4	5	6
1	U	U	U	LU	L	LU
2	LU	L	L	U	LU	L
3	U	U	LU	L	U	U
4	LU	U	U	U	LU	L
5	U	LU	U	U	U	U
6	U	U	U	LU	U	LU
7	LU	U	U	U	LU	U

算法实例

	1	2	3	4	5	6	7
X_i	*A*	*B*	*C*	*B*	*D*	*A*	*B*
Y_j	*B*	*D*	*C*	*A*	*B*	*A*	

	C	B	A

$C[\,]$

$i \backslash j$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	1	1	1	1	2	2
3	0	1	1	2	2	2	2
4	0	1	1	2	2	3	3
5	0	1	2	2	2	3	3
6	0	1	2	2	3	3	4
7	0	1	2	2	3	4	4

$rec[\,]$

$i \backslash j$	1	2	3	4	5	6
1	U	U	U	LU	L	LU
2	LU	L	L	U	LU	L
3	U	U	LU	L	U	U
4	LU	U	U	U	LU	L
5	U	LU	U	U	U	U
6	U	U	U	LU	U	LU
7	LU	U	U	U	LU	U

算法实例

	1	2	3	4	5	6	7
X_i	*A*	*B*	*C*	*B*	*D*	*A*	*B*
Y_j	*B*	*D*	*C*	*A*	*B*	*A*	

B	C	B	A

最长公共子序列

$C[\,]$

i \ j	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	1	1	1	1	2	2
3	0	1	1	2	2	2	2
4	0	1	1	2	2	3	3
5	0	1	2	2	2	3	3
6	0	1	2	2	3	3	4
7	0	1	2	2	3	4	4

$rec[\,]$

i \ j	1	2	3	4	5	6
1	U	U	U	LU	L	LU
2	LU	L	L	U	LU	L
3	U	U	LU	L	U	U
4	LU	U	U	U	LU	L
5	U	LU	U	U	U	U
6	U	U	U	LU	U	LU
7	LU	U	U	U	LU	U

伪代码

- **Longest-Common-Subsequence(X, Y)**

输入: 两个序列X,Y
输出: X和Y的最长公共子序列

$n \leftarrow \text{length}(X)$
$m \leftarrow \text{length}(Y)$
//初始化
新建二维数组$C[0..n, 0..m]$和$rec[0..n, 0..m]$
for $i \leftarrow 0$ *to* n **do**
  $C[i, 0] \leftarrow 0$
end
for $j \leftarrow 0$ *to* m **do**
  $C[0, j] \leftarrow 0$
end

初始化

时间复杂度分析

- **Longest-Common-Subsequence(X, Y)**

//动态规划

for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to m **do**
    if $X_i = Y_j$ **then**
      $C[i,j] \leftarrow C[i-1,j-1]+1$
      $rec[i,j] \leftarrow$ "LU"
    end
    else if $C[i-1,j] \geq C[i,j-1]$ **then**
      $C[i,j] \leftarrow C[i-1,j]$
      $rec[i,j] \leftarrow$ "U"
    end
    else
      $C[i,j] \leftarrow C[i,j-1]$
      $rec[i,j] \leftarrow$ "L"
    end
  end
end
return C, rec

时间复杂度：$O(n \cdot m)$

谢谢

BDA

Beihang University