Design and Analysis of Algorithms

Part II: Dynamic Programming

Lecture 13: Longest Common Substrings

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
  - 将给定序列中零个或多个元素（如字符）去掉后所得结果
- 子串
  - 给定序列中零个或多个**连续**的元素（如字符）组成的子序列
- 示例
  - 给定序列X

X	A	B	C	B	D	A	B

X的子序列

X_{seq}	A	C	B	B

X的子串

X_{str}	B	C	B

问题背景：公共子串

- 给定两个序列X和Y

X	A	B	C	A	D	B	B

Y	B	C	E	D	B	B

- 公共子串示例

X_1	B

Y_1	B

X_2	B	C

Y_2	B	C

X_3	D	B	B

Y_3	D	B	B

问题：如何求两个给定序列的最长公共子串?

问题定义

- 形式化定义

最长公共子串问题

Longest Common Substring 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

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

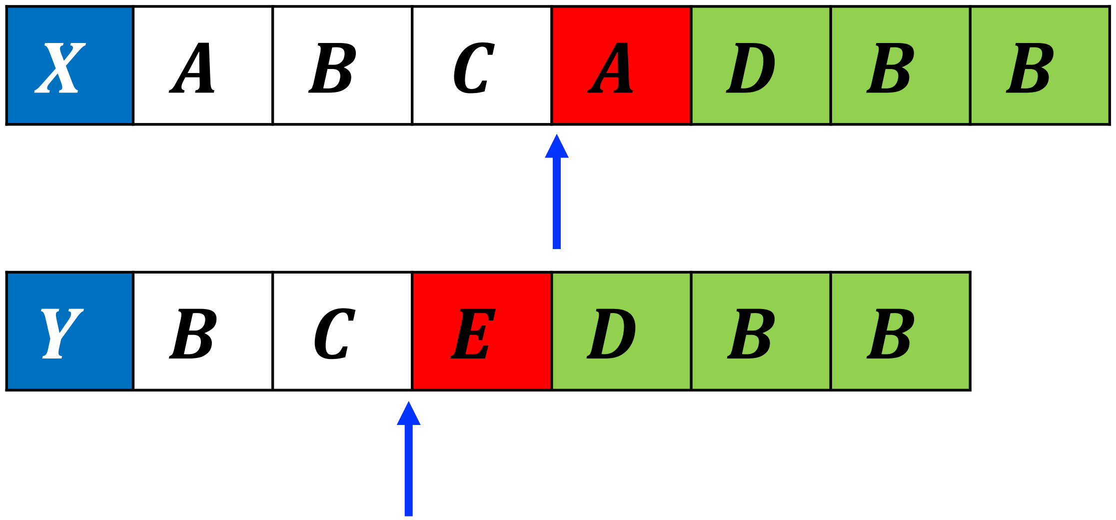
$$\max |Z|$$

优化目标

$$s.t. Z = < x_i, x_{i+1}, \dots, x_{i+l-1} > = < y_j, y_{j+1}, \dots, y_{j+l-1} >$$
$$(1 \le i \le n - l + 1; 1 \le j \le m - l + 1)$$

约束条件

蛮力枚举



- 序列X和序列Y各选择一个位置$X[7]$和$Y[6]$
- 依次检查元素是否匹配
  - 元素相等继续匹配
  - 元素不等(或某序列已达端点)匹配终止

蛮力枚举

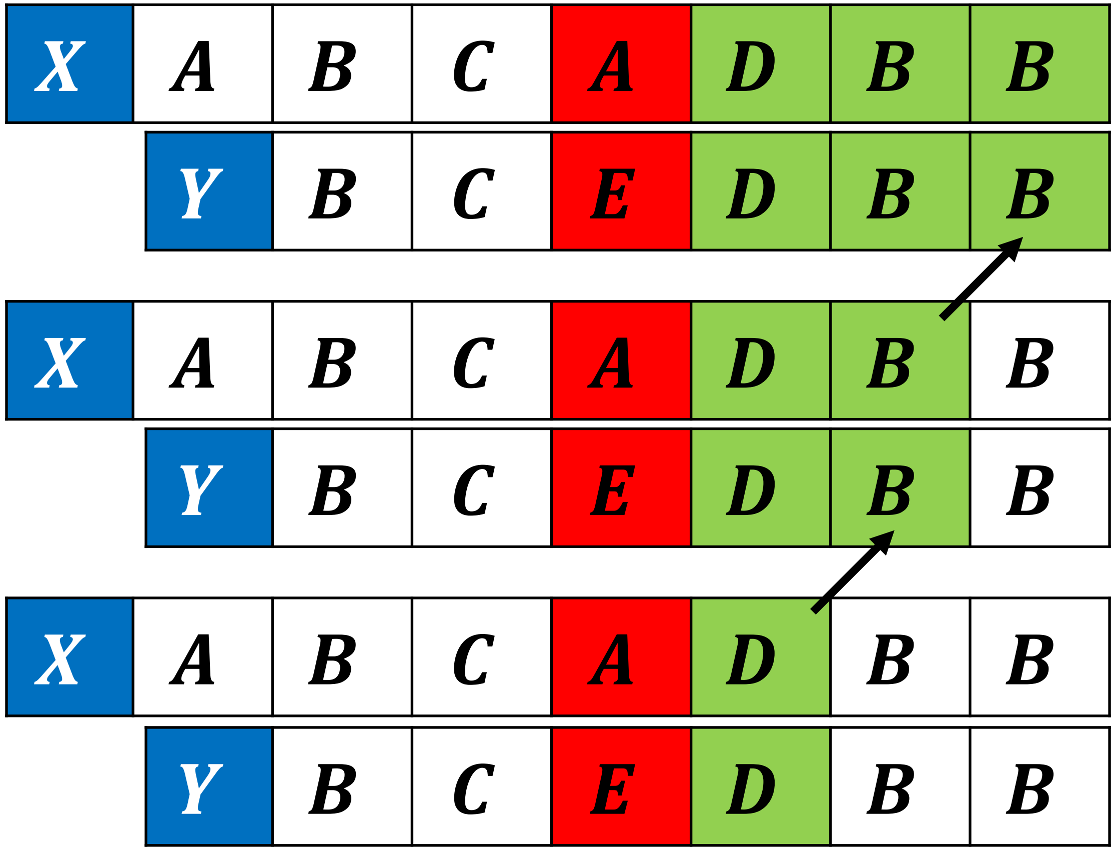
X	A	B	C	A	D	B	B

Y	B	C	E	D	B	B

最长公共子串长度为3

- 枚举所有的$X[i], Y[j]$
- 求以其为结尾的尽可能长的公共子串
- 记录最长公共子串长度

枚举观察



- 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$Y[1..j]$中，以x_i和y_j结尾的最长公共子串$Z[1..l]$的长度

X_i	x_1	x_2	…	x_{i-1}	x_i
Y_j	y_1	y_2	…	y_{j-1}	y_j

x_i

$\|$

Z_l	z_1	…	z_{l-1}	z_l

$\|$

y_j

- 明确原始问题
  - $p_{max} = \max\limits_{1\le i\le n, 1\le j\le m}\{C[i,j]\}$
    - $X[1..n]$和$Y[1..m]$中最长公共子串的长度

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

- 情况1：$x_7 \neq y_6$

X	A	B	C	A	D	B	B

Y	B	C	E	D	B	A

- 情况2：$x_7 = y_6$

X	A	B	C	A	D	B	B

Y	B	C	E	D	B

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

- 情况1：$x_7 \neq y_6$

$$C[7,6] = 0$$

X	A	B	C	A	D	B	B
	Y	B	C	E	D	B	A

不存在以其结尾的公共子串

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

- 情况1：$x_i \neq y_j$

$$C[i,j] = 0$$

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

无子问题

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

- 情况2：$x_7 = y_6$

$C[7,6]$

X	A	B	C	A	D	B	B
	Y	B	C	E	D	B	B

存在以其结尾的公共子串

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

X	A	B	C	A	D	B	B
	Y	B	C	E	D	B	B

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

- 情况2：$x_i = y_j$

$C[i,j]$

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

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

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

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

$$\begin{tikzcd} \boxed{C[i,j]} \\ \boxed{C[i-1,j-1]} \arrow[u, "+1"'] \end{tikzcd}$$

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，最长公共子串为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：依次求解问题

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

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

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，最长公共子串为0
- 原始问题
  - $p_{max} = \max_{1\le i\le n, 1\le j\le m} \{C[i,j]\}$

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

最优方案追踪

- 记录决策过程
  - 最长公共子串末尾位置为p_{max}
  - 最长公共子串长度为l_{max}
- 输出最优方案
  - 最长公共子串$< x_{p_{max}-l+1}, x_{p_{max}-l+2}, \dots, x_{p_{max}} >$

X	x_1	$\dots$	$x_{p_{max}-l+1}$	$\dots$	$x_{p_{max}}$	$\dots$	x_n

长度为l_{max}

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	A	D	B	B
Y_j	B	C	E	D	B	B	

位置$p_{max} = 0$
长度$l_{max} = 0$

$C[\,]$

i \ 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

算法实例

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

$x_i \neq y_j$

位置$p_{max} = 0$
长度$l_{max} = 0$

$C[\,]$

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

算法实例

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

$x_i = y_j$

位置$p_{max} = 2$

长度$l_{max} = 1$

$C[\,]$

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

算法实例

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

位置$p_{max} = 7$
长度$l_{max} = 3$

$C[\,]$

$i \backslash j$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	1	0	0	0	1	1
3	0	0	2	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0
6	0	1	0	[illegible]	[illegible]	[illegible]	1
7	0	1	0	0	0	1	$\boxed{3}$

最长公共子串长度

算法实例

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

位置$p_{max} = 7$
长度$l_{max} = 3$

$C[\,]$

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

伪代码

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

输入: 两个字符串X, Y
输出: X和Y的最长公共子串
//初始化
$n \leftarrow \text{length}(X)$
$m \leftarrow \text{length}(Y)$
新建二维数组$C[0..n, 0..m]$
$l_{max} \leftarrow 0$
$p_{max} \leftarrow 0$
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

伪代码

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

```
//动态规划
for i ← 1 to n do
    for j ← 1 to m do
        if X_i ≠ Y_j then
            C[i, j] ← 0
        end
        else
            C[i, j] ← C[i − 1, j − 1] + 1
            if C[i, j] > l_max then
                l_max ← C[i, j]
                p_max ← i
            end
        end
    end
end
return l_max, p_max
```

伪代码

- **Print-LCS(X, l_{max}, p_{max})**

```
输入: 字符串 X, l_max, p_max
输出: X 和 Y的最长公共子串
if l_max = 0 then
|   return NULL
end
for i ← (p_max − l_max + 1) to p_max do
|   print X_i
end
```

时间复杂度分析

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

```
//动态规划
for i ← 1 to n do
    for j ← 1 to m do
        if X_i ≠ Y_j then
            C[i, j] ← 0
        end
        else
            C[i, j] ← C[i − 1, j − 1] + 1
            if C[i, j] > l_max then
                l_max ← C[i, j]
                p_max ← i
            end
        end
    end
end
return l_max, p_max
```

时间复杂度：$O(n \cdot m)$

小结

最长公共子序列

X	D	B	D	A	B
Y	C	A	B	B	

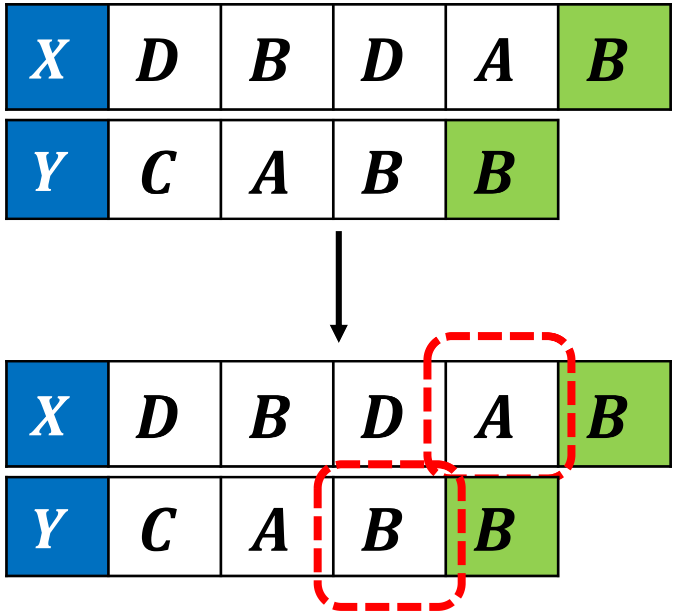
最长公共子串

X	D	B	D	A	B
Y	C	A	B	B	

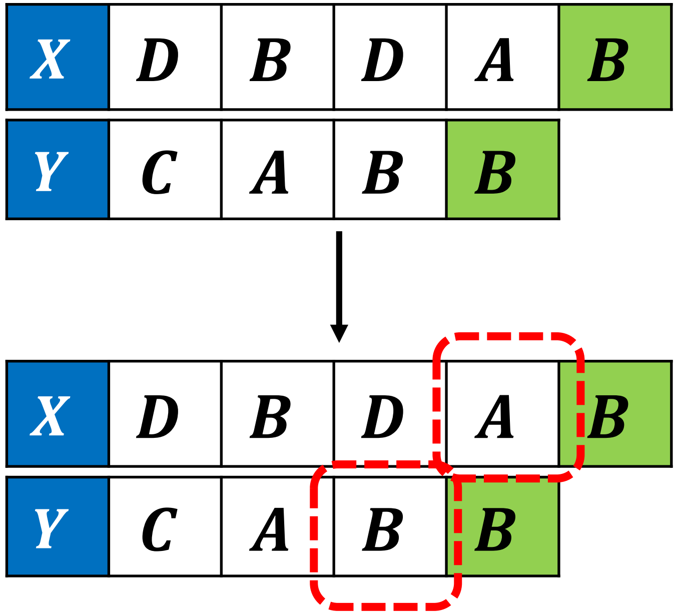
情况2：$x_5 = y_4$

小结

最长公共子序列



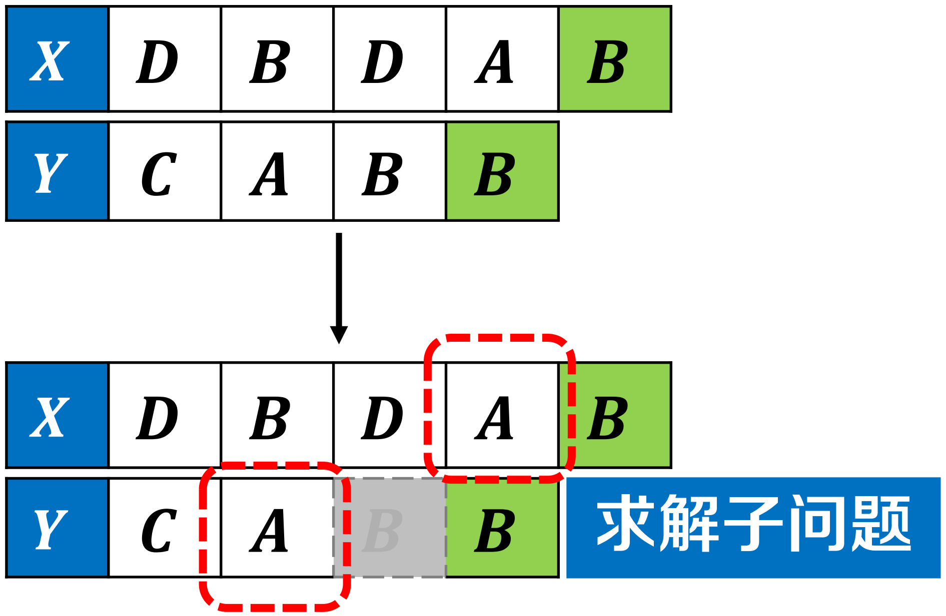
最长公共子串



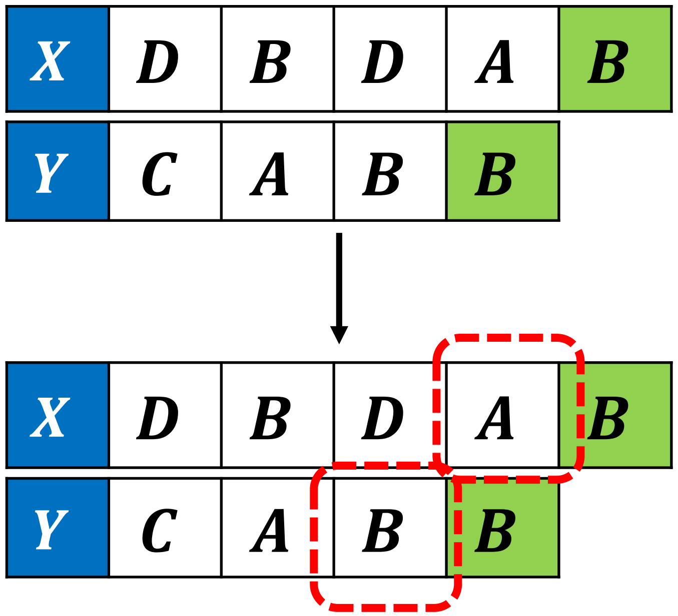
情况1：$x_4 \neq y_3$

小结

最长公共子序列



最长公共子串



小结

最长公共子序列

X	D	B	D	A	B
Y	C	A	B	B	

X	D	B	D	A	B
Y	C	A	B	B	

求解子问题

X	D	B	D	A	B
Y	C	A	B	B	

最长公共子串

X	D	B	D	A	B
Y	C	A	B	B	

X	D	B	D	A	B
Y	C	A	B	B	

无子问题

谢谢

BDA
Beihang University