title: [15W--ch02-strings]高级算法-LCS算法

author:

```
- name: 王赫^[2024244022,计算机科学与技术]
```

递归方程

幻灯片左上部分展示了 LCS 问题的递归方程。给定两个序列 X 和 Y，我们定义 $c[i,j]$ 为 $X[1..i]$ 和 $Y[1..j]$ 的 LCS 的长度。递归方程如下：

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1,j-1]+1 & \text{if } x[i] = y[j] \\ \max(c[i-1,j], c[i,j-1]) & \text{otherwise} \end{cases}$$

动态规划算法

```
LCS(X, Y, m, n, b)
1. for i = 1 to m do
2.     c[i, 0] = 0
3. for j = 0 to n do
4.     c[0, j] = 0
5. for i = 1 to m do
6.     for j = 1 to n do
7.         if x[i] == y[j] then
8.             c[i, j] = c[i-1, j-1] + 1
9.             b[i, j] = 1  // diagonal
10.        else if c[i-1, j] >= c[i, j-1] then
11.            c[i, j] = c[i-1, j]
12.            b[i, j] = 2  // up
13.        else
14.            c[i, j] = c[i, j-1]
15.            b[i, j] = 3  // left
```

LCS 打印算法

```
PrintLCS(b, X, i, j)
1. i = m
2. j = n
3. if i == 0 or j == 0 then exit
4. if b[i, j] == 1 then
5.     PrintLCS(b, X, i-1, j-1)
6.     print X[i]
7. else if b[i, j] == 2 then
8.     PrintLCS(b, X, i-1, j)
9. else
10.    PrintLCS(b, X, i, j-1)
```

功能：x=abc；y=acd -> abc_；a_cd

重构对齐后的子串

```
AlignLCS(b, X, Y, i, j)
1. aligned_X = "", i = m
2. aligned_Y = "", j = n
3. while i > 0 or j > 0 do
4.     if i > 0 and j > 0 and b[i, j] == 1 then
5.         aligned_X = X[i-1] + aligned_X
6.         aligned_Y = Y[j-1] + aligned_Y
7.         i = i - 1
8.         j = j - 1
9.     else if i > 0 and (j == 0 or b[i, j] == 2) then
10.        aligned_X = X[i-1] + aligned_X
11.        aligned_Y = "_" + aligned_Y
12.        i = i - 1
13.    else
14.        aligned_X = "_" + aligned_X
15.        aligned_Y = Y[j-1] + aligned_Y
16.        j = j - 1
17. return (aligned_X, aligned_Y)
```

动态规划构建 LCS 表格并同时构建对齐后的子串

```
LCS_And_Align(X, Y, m, n)
1. c = array(m+1, n+1)
2. aligned_X = ""
3. aligned_Y = ""
4. for i = 0 to m do
5.     c[i, 0] = 0
6. for j = 0 to n do
7.     c[0, j] = 0
8. for i = 1 to m do
9.     for j = 1 to n do
10.        if X[i-1] == Y[j-1] then
11.            c[i, j] = c[i-1, j-1] + 1
12.            aligned_X = aligned_X + X[i-1]
13.            aligned_Y = aligned_Y + Y[j-1]
14.        else if c[i-1, j] >= c[i, j-1] then
15.            c[i, j] = c[i-1, j]
16.            aligned_X = aligned_X + X[i-1]
17.            aligned_Y = aligned_Y + "_"
18.        else
19.            c[i, j] = c[i, j-1]
20.            aligned_X = aligned_X + "_"
21.            aligned_Y = aligned_Y + Y[j-1]
22. // Fill the remaining parts of the aligned strings if any
23. while i > 0 do
24.     aligned_X = X[i-1] + aligned_X
25.     aligned_Y = "_" + aligned_Y
26.     i = i - 1
27. while j > 0 do
28.     aligned_X = "_" + aligned_X
29.     aligned_Y = Y[j-1] + aligned_Y
30.     j = j - 1
31. return (aligned_X, aligned_Y)
```