**从VB到Python之最长连续公共子串**

余姚二中 梁见斌

VB代码：

17.【加试题】（2018 学年第一学期浙江七彩阳光联盟第二次联考）用 VB 编程求两个字符串的最长连续公共子串，程序功能是：在文本框 Text1 和 Text2分别输入任意两串字符串 s1 和 s2，单击命令按钮 Command1，程序在标签 Label3 和 Label4 中分别输出这两个子串的最长连续公共子串和子串的长度。

其算法思想是：分别从字符串 s1 和 s2 的左边第一个字符开始检查，若发现这两个字符中有一个字符相同，则以这个字符为基准向右边扩大搜索范围，先检查其后面的第一个字符是否相同，若还是相同则继续搜索，直到找到不同的字符为止。然后按照该方法依次继续往后搜索，直到查找结束。

程序界面如图所示，实现上述功能的 VB 程序如下，请问答下列问题：

（1）根据代码，若输入的 s1 为“Teacher”,s2 为“teaciher”，则最长连续公共子串为\_\_\_\_\_\_\_\_。

（2）请在划线处填入合适代码。

Dim s1 As String, s2 As String, maxstr As String

Dim len1 As Integer, len2 As Integer, maxlen As Integer

Function Min(a As Integer, b As Integer) As Integer

If a >= b Then Min = b Else Min = a

End Function

Function Search(m As Integer, n As Integer) As Integer

Dim c As Integer

c = 1

Do While c <= Min(len1 - m, len2 - n)

If Mid(s1, m + c, 1) = Mid(s2, n + c, 1) Then

①

Else

Exit Do 'Exit Do 的作用是退出 Do 循环

End If

Loop

Search = c - 1

End Function

Private Sub Command1\_Click()

Dim i As Integer, j As Integer, k As Integer

s1 = Text1.Text: s2 = Text2.Text

len1 = Len(s1): len2 = Len(s2)

maxlen = 0: maxstr = ""

For i = 1 To len1

For j = 1 To len2

If Mid(s1, i, 1) = Mid(s2, j, 1) Then

k = ② 'k 用于记录公共连续子串的长度

If (k > maxlen) Then

maxlen = k

maxstr = ③

End If

End If

Next j

Next i

Label3.Caption = "最长连续公共子串为：" & maxstr

Label4.Caption = "该子串长度是：" & Str(maxlen)

End Sub

答案：（1）eac （1分）

（2）①c = c + 1 （1分）

②Search(i, j) + 1或 Search(i-1, j-1)（2分）

③Mid(s1, i, maxlen) 或Mid(s1, i, k)

或Mid(s2, j, k) 或mid(s2,j,maxlen) （2分）

对应Python代码：

#算法1：最直接的暴力穷举，需要三重循环，效率较低

def LSS\_length\_1(n, m):

#最大长度和子串的末端索引（按照左闭右开原则，实际上该索引对应的元素取不到）

max\_len, end\_pos = 0, 0

for i in range(n):

for j in range(m):

if x[i] == y[j]:

c, min\_len = 1, min(n - i, m - j)

while c < min\_len and x[i+c] == y[j+c]: #统计连续子串长度

c += 1

if max\_len < c: #更新最大长度和子串的末端索引

max\_len, end\_pos = c, j + c

return (max\_len, end\_pos)

with open('zcggzc.txt', 'r') as fin:

for line in fin.readlines(): #依次读取每行

line = line.strip() #去掉每行头尾空白

x, y = tuple(line.split())

print(x, y)

max\_len, end\_pos = LSS\_length\_1(len(x), len(y))

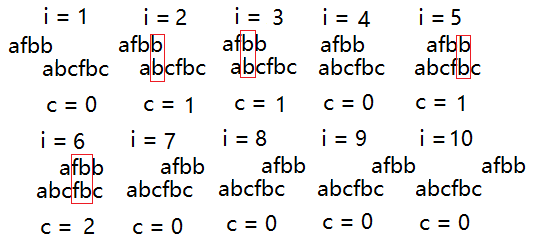
print("最大公共子串为：{} 其长度是：{}"

.format(y[end\_pos-max\_len:end\_pos],max\_len))

算法1是最直接的暴力穷举，需要三重循环，效率较低；我们可以对其进行改进，使用一个优化了的穷举法，只需二重循环，效率有所提高。其原理如下：

将字符串x和y分别写在两把直尺上面，然后将x固定，x的左端和y的右端对齐，然后向右移动y，比较重叠部分字符串中公共子串的长度，直到y移动到x的右端。在这个过程中求得的最大公共子串就是x和y的最大公共子串。

     下图是求解过程的图示，红框表示重叠部分的公共子串。



#算法2：x不动，y从左向右逐个移动比较，移动n+m步，只需二重循环，效率有所提高

def LSS\_length\_2(n, m):

#最大长度和子串的末端索引（按照左闭右开原则，实际上该索引对应的元素取不到）

max\_len, end\_pos = 0, 0

for i in range(1, n+m):

start\_x, start\_y = max(i-m, 0), max(m-i, 0) #x和y的起始比较位置

cover\_len = i - start\_x - max(i-n, 0) #字符串重叠部分长度

c = 0

for j in range(cover\_len): #遍历重叠部分

if x[start\_x+j] == y[start\_y+j]: #统计连续子串长度

c += 1

else:

if max\_len < c: #更新最大长度和子串的末端索引

max\_len, end\_pos = c, start\_y+j

c = 0

return (max\_len, end\_pos)

前面两种算法都属于穷举法，他们的特点是算法思路简洁，便于编程，但是时间复杂度较高，属于低效算法。我们可以用动态规划思想解决“最长连续公共子串”问题，它的特点是引进一个二维列表b，用b[i][j]记录x[i]与y[j]的连续公共子串的长度，最长公共子串的长度为max(b[i][j]), i∈[1,n],j∈[1,m]。代码如下：

#算法3：动态规划，使用一个二维列表来存储中间值

def LSS\_length\_3(n, m):

#最大长度和子串的末端索引（按照左闭右开原则，实际上该索引对应的元素取不到）

max\_len, end\_pos = 0, 0

b = [[0 for i in range(len(y)+1)] for j in range(len(x)+1)]

for i in range(1, n+1):

for j in range(1, m+1):

if x[i-1] == y[j-1]:

b[i][j] = b[i-1][j-1] + 1

if max\_len < b[i][j]: #更新最大长度和子串的末端索引

max\_len, end\_pos = b[i][j], j

else:

b[i][j] = 0

return (max\_len, end\_pos)

考虑到算法3使用二维列表存储中间值，需要较大的空间，并且存在浪费现象，我们还可以对其进行降维优化，使用2个一维列表来存储中间值。代码如下：

#算法4：对LSS\_length\_3()进行降维优化，使用2个一维列表来存储中间值

def LSS\_length\_4(n, m):

#最大长度和子串的末端索引（按照左闭右开原则，实际上该索引对应的元素取不到）

max\_len, end\_pos = 0, 0

pre = [0] \* (m + 1) #pre[j]相当于b[i-1][j]

cur = [0] \* (m + 1) #cur[j]相当于b[i][j]

for i in range(1, n+1):

for j in range(1, m+1):

if x[i-1] == y[j-1]:

cur[j] = pre[j-1] + 1

if max\_len < cur[j]: #更新最大长度和子串的末端索引

max\_len, end\_pos = cur[j], j

else:

cur[j] = 0

for j in range(1, m+1):

pre[j] = cur[j]

return (max\_len, end\_pos)

至此，我们为大家展示了解决“最长连续公共子串”问题的几种常用算法，简明高效的算法与简洁明了的Python语言相结合，是学习算法，解决实际问题的有力武器。