

**2021年秋季学期  
计算学部《软件安全》**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 余涛 |
| 学号 | 1180300829 |
| 专业 | 信息安全 |
| 班号 | 1803202 |
| 手机号码 | 15586430583 |

## 1、实验项目描述

面向网络恶意代码的特征提取

1. 理解基于最长公共子序列的协议特征提取方法
2. 掌握网络恶意代码特征的提取流程
3. 学习最长公共子序列的提取算法
4. 实现字符串最长公共子序列的提取算法
5. 利用动态规划的方法实现字符串最长公共子序列的提取
6. 依据输入的字符串构建L(m,n)数组，利用L(m,n)数组查找两个字符串之间的最长公共子序列

## 2、实验要求

1. 实验数据准备。利用ASCII字符集做为输入集，不考虑多字节编码的中文、英文字符集。
2. 程序的输入部分：2个字符串。输出部分：这2个字符串的最长公共子序列，如有多个一同给出。
3. 实验结果和实验数据一起给出。

## 3、实验结果

首先对该实验算法进行分析，分为两部分：构建和输出。其中构建指的是根据两个序列，利用动态规划的方法，构建L数组。而输出则是根据L数组进行输出。（注：在本实验中有要求如果有多个最长公共子序列则需要一同输出，因此需要格外注意这种特殊情况）。

（1）构建：根据输入的两个字符串S1和S2，构建LCS数组。并且为了方便输出，在这一步同时构建了一个路径数组route用于进行回溯

构建流程图如下：



如上图可以发现，除了构建LCS数组，也构建一个route二维数组，这个route数组能够指出回溯的路径，方便查找到最长公共子序列。

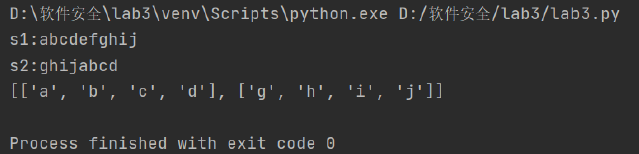
（2）输出：在输出算法我们可以根据route数组进行回溯。只需要从route数组的最大下标开始回溯。当route数组为0时，当前所代表的字符就是最长公共子序列中的一个字符。当route数组为1时，向左回溯；当route数组为为2时，向上回溯。当route数组为3时，就需要考虑分支情况。采用递归的方法回溯。

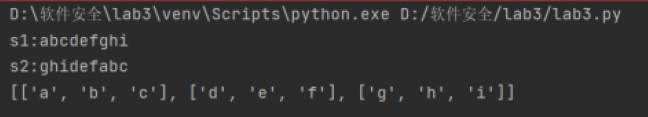
构建流程图如下：



如上图可以发现，只需要利用route数组和字符串s1进行回溯。

结果如下：





对以上两种结果可以发现，成功的输出了最长公共子序列（对于多个最长公共子序列也能实现）。

## 4、实验代码

**import** numpy **as** np

**def** get\_matrix(s1,s2):

len\_1=len(s1)

len\_2=len(s2)

**return** np.zeros((len\_1+1,len\_2+1),dtype=np.int32)

**def** lcs(s1,s2):

matrix= get\_matrix(s1,s2)

best\_route=get\_matrix(s1,s2)best\_route=get\_matrix(s1,s2)

**for** m **in** range(len(s1)):**for** m **in** range(len(s1)):

**for** n **in** range(len(s2)):**for** n **in** range(len(s2)):

**if** s1[m]==s2[n]:**if** s1[m]==s2[n]:

best\_route[m+1,n+1]=0best\_route[m+1,n+1]=0

matrix[m+1,n+1]=matrix[m,n]+1matrix[m+1,n+1]=matrix[m,n]+1

**else**:**else**:

**if** matrix[m,n+1]>matrix[m+1,n]:**if** matrix[m,n+1]>matrix[m+1,n]:

matrix[m+1,n+1]=matrix[m,n+1]matrix[m+1,n+1]=matrix[m,n+1]

best\_route[m+1,n+1]=1best\_route[m+1,n+1]=1

**elif** matrix[m+1,n]>matrix[m,n+1]:**elif** matrix[m+1,n]>matrix[m,n+1]:

matrix[m + 1, n + 1] = matrix[m+1, n ]matrix[m + 1, n + 1] = matrix[m+1, n ]

best\_route[m + 1, n + 1] = 2best\_route[m + 1, n + 1] = 2

**else**:**else**:

matrix[m+1,n+1]=matrix[m+1,n]matrix[m+1,n+1]=matrix[m+1,n]

best\_route[m + 1, n + 1] = 3best\_route[m + 1, n + 1] = 3

**return** best\_routereturn best\_route

**def** print\_lcs(s1,best\_route,i,j,result\_set,result):

**def** print\_lcs(s1,best\_route,i,j,result\_set,result):

**if** i==0if i==0 **or** j==0:**or** j==0:

**if** len(result)==0:**if** len(result)==0:

returnreturn

result.reverse()result.reverse()

**if** result **in** result\_set:**if** result **in** result\_set:

returnreturn

result\_set.append(result)result\_set.append(result)

returnreturn

**if** best\_route[i,j]==0:**if** best\_route[i,j]==0:

result.append(s1[i result.append(s1[i -- 1])1])

priprint\_lcs(s1,best\_route,int\_lcs(s1,best\_route,i--1,j1,j--1,result\_set,result)1,result\_set,result)

**elif** best\_route[i,j]==1:**elif** best\_route[i,j]==1:

print\_lcs(s1,best\_route,iprint\_lcs(s1,best\_route,i--1,j,result\_set,result)1,j,result\_set,result)

**elif** best\_route[i,j]==2:**elif** best\_route[i,j]==2:

print\_lcs(s1,best\_route,i,jprint\_lcs(s1,best\_route,i,j--1,result\_set,result)1,result\_set,result)

**elif** best\_route[i,j]==3:**elif** best\_route[i,j]==3:

new\_result=result[:]new\_result=result[:]

print\_lcs(s1,best\_route,iprint\_lcs(s1,best\_route,i--1,j,result\_set,new\_result)1,j,result\_set,new\_result)

new\_result=result[:]new\_result=result[:]

print\_lcs(s1,best\_route,i,jprint\_lcs(s1,best\_route,i,j--1,result\_set,new\_result)1,result\_set,new\_result)

**def** print\_lcs1(s1,best\_route,i,j,result):

**def** print\_lcs1(s1,best\_route,i,j,result):

**if** i==0 **or** j==0:**if** i==0 **or** j==0:

rreturneturn

**else**:**else**:

**if** best\_route[i,j]==0:**if** best\_route[i,j]==0:

result.append(s1[iresult.append(s1[i--1])1])

print\_lcs1(s1,best\_route,iprint\_lcs1(s1,best\_route,i--1,j1,j--1,result)1,result)

**elif** best\_route[i,j]==1:**elif** best\_route[i,j]==1:

print\_lcs1(s1,best\_route,iprint\_lcs1(s1,best\_route,i--1,j,result)1,j,result)

**elif** best\_route[i,j]==2:**elif** best\_route[i,j]==2:

print\_lcs1(s1,best\_route,i,jprint\_lcs1(s1,best\_route,i,j--1,result)1,result)

**elif** best\_route[i,j]==3:**elif** best\_route[i,j]==3:

result.append('(')result.append('(')

print\_lcs1(s1,best\_route,iprint\_lcs1(s1,best\_route,i--1,j,result)1,j,result)

result.append('+')result.append('+')

print\_lcs1(s1,best\_route,i,jprint\_lcs1(s1,best\_route,i,j--1,result)1,result)

result.append(')')result.append(')')

s1='abcdefghi'

s1='abcdefghi'

s2='ghidefabc'

s2='ghidefabc'

g=lcs(s1,s2)

g=lcs(s1,s2)

result\_set=[]

result\_set=[]

result\_str=[]

result\_str=[]

print\_lcs(s1,g,g.shape[0]

print\_lcs(s1,g,g.shape[0]--1,g.shape[1]1,g.shape[1]--1,result\_set,result\_str)1,result\_set,result\_str)

**print**('s1:'+s1)

**print**('s1:'+s1)

**print**('s2:'+s2)

**print**('s2:'+s2)

**print**(result\_set)

**print**(result\_set)