哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

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

（1） 掌握网络恶意代码特征的提取流程

（2） 学习最长公共子序列的提取算法

2、 实现字符串最长公共子序列的提取算法

（1） 利用动态规划的方法实现字符串最长公共子序列的提取

（2） 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。

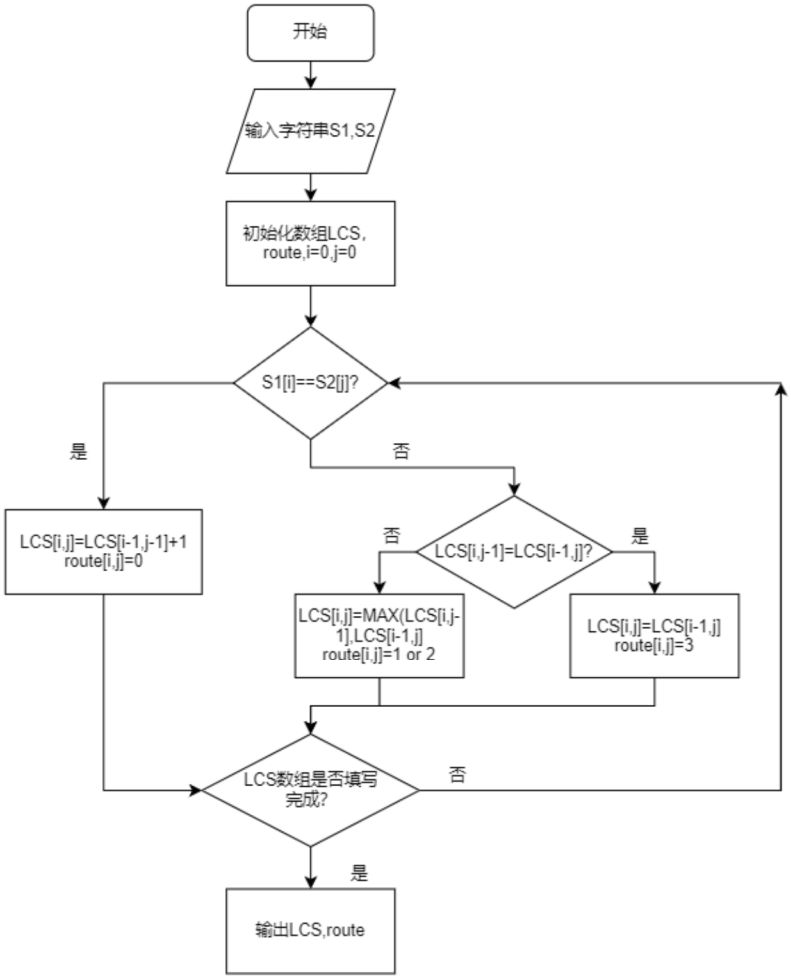
3、 实验结果和实验数据一起给出。

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)。

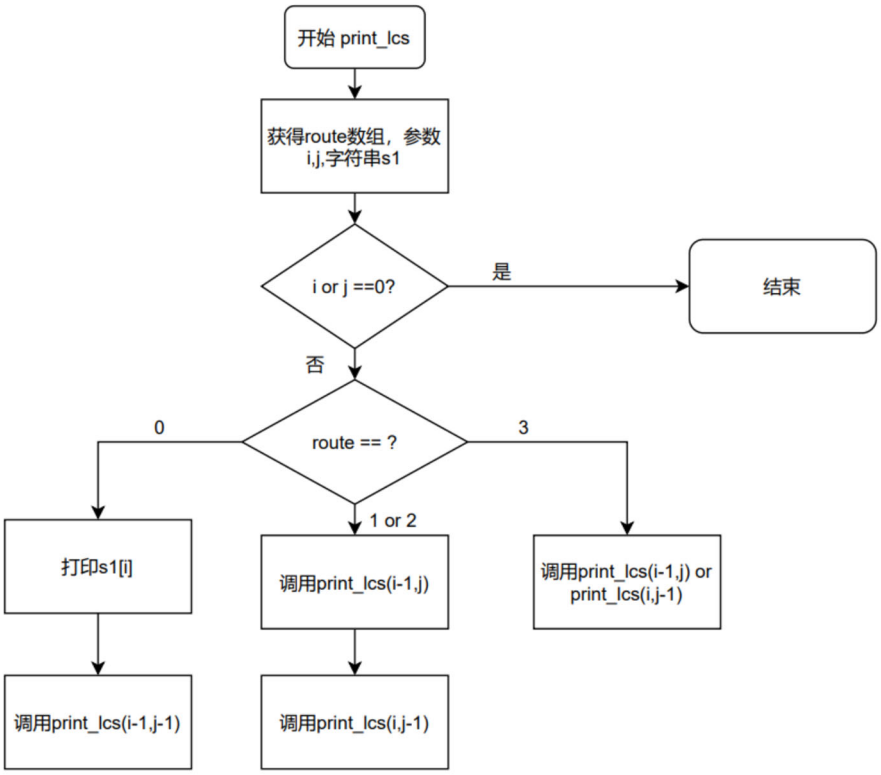
（1）构建：根据输入的两个字符串 S1 和 S2，构建 LCS 数组。并且为了方便输出，在这一步同时构建了一个路径数组 route 用于进行回溯

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：

```
D:\软件安全\lab3\venv\Scripts\python.exe D:/软件安全/lab3/lab3.py
s1:abcdefghij
s2:ghijabcd
[['a', 'b', 'c', 'd'], ['g', 'h', 'i', 'j']]

Process finished with exit code 0
```

```
D:\软件安全\lab3\venv\Scripts\python.exe D:/软件安全/lab3/lab3.py
s1:abcdefghi
s2:ghidefabc
[['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'h', 'i']]
```

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,re
sult)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,ne
w_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,resul
t)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,res
ult_str)1,result_set,result_str)

print('s1:'+s1)

print('s1:'+s1)

print('s2:'+s2)

print('s2:'+s2)

print(result_set)

print(result_set)
```