# 实验三 恶意代码特征提取

班级：1703202

学号：1170300421

专业：信息安全

日期：2020.11.1

# 实验三 恶意代码特征提取

## 一、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

2、实现字符串最长公共子序列的提取算法

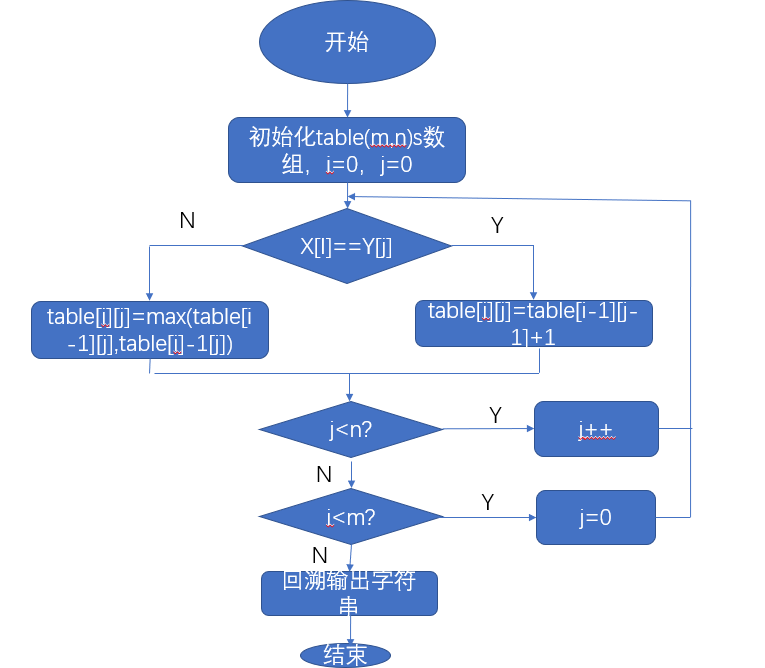
1. 利用动态规划的方法实现字符串最长公共子序列的提取
2. 依据输入的字符串构建L(m,n)数组，利用L(m,n)数组查找两个字符串之间的最长公共子序列

## 二、实验要求

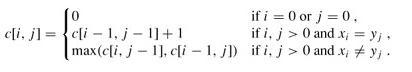
1. 实验数据准备。利用ASCII字符集做为输入集，不考虑多字节编码的中文、英文字符集。
2. 程序的输入部分：2个字符串。输出部分：这2个字符串的最长公共子序列，如有多个一同给出。
3. 实验结果和实验数据一起给出。

## 三、实验结果

1、设计的算法流程图



算法使用动态规划的方法，利用公式



构造动态规划表table。

我们需要在动态规划表上进行回溯 —— 从table[m][n]，开始进行判断：

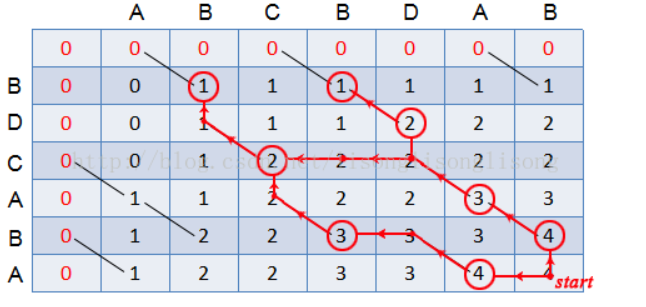
如果格子table[i][j]对应的X[i-1] == Y[j-1]，则把这个字符放入 LCS 中，并跳入table[i-1][j-1]中继续进行判断；

如果格子table[i][j]对应的 X[i-1] ≠ Y[j-1]，则比较table[i-1][j]和table[i][j-1]的值，跳入值较大的格子继续进行判断；

直到 i 或 j 小于等于零为止，倒序输出 LCS 。

如果出现table[i-1][j]等于table[i][j-1]的情况，说明最长公共子序列有多个，故两边都要进行回溯。

例如针对字符串ABCBDAB，BDCABA的动态规划表进行回溯如下回溯：



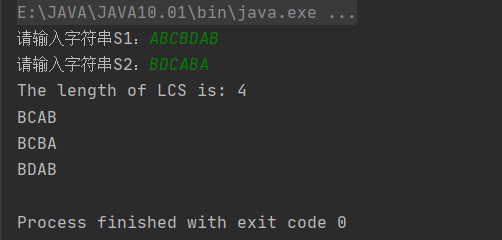
2. 关键的数据结构，及简单说明

字符串String ：S1，S2

动态规划数组：table[][]

TreeSet类：set，用于回溯时记录所有最长公共子序列

3.实验结果截图：



4.程序代码

package Lcs;

import java.util.Scanner;

import java.util.TreeSet;

public class lcscommen {

private String X;

private String Y;

private int[][] table; // 动态规划表

private TreeSet<String> set = new TreeSet<String>();

public lcscommen(String X, String Y) {

this.X = X;

this.Y = Y;

}

//求两个数中的较大者

private int max(int a, int b) {

return (a>b) ? a:b;

}

//构造表，并返回X和Y的LCS的长度

private int lcs(int m, int n) {

table = new int[m+1][n+1]; // 表的大小为(m+1)\*(n+1)

for(int i=0; i<m+1; ++i) {

for(int j=0; j<n+1; ++j) {

// 第一行和第一列置0

if (i == 0 || j == 0)

table[i][j] = 0;

else if(X.charAt(i-1) == Y.charAt(j-1))

table[i][j] = table[i-1][j-1] + 1;

else

table[i][j] = max(table[i-1][j], table[i][j-1]);

}

}

return table[m][n];

}

//回溯

private void traceBack(int i, int j, String lcs\_str) {

while (i>0 && j>0) {

if (X.charAt(i-1) == Y.charAt(j-1)) {

lcs\_str += X.charAt(i-1);

--i;

--j;

}

else {

if (table[i-1][j] > table[i][j-1])

--i;

else if (table[i-1][j] < table[i][j-1])

--j;

else { // 相等的情况

traceBack(i-1, j, lcs\_str);

traceBack(i, j-1, lcs\_str);

return;

}

}

}

set.add(reverse(lcs\_str));

}

//字符串逆序

private String reverse(String str) {

StringBuffer strBuf = new StringBuffer(str).reverse();

return strBuf.toString();

}

//打印输出

public void printLCS() {

int m = X.length();

int n = Y.length();

int length = lcs(m,n);

String str = "";

traceBack(m,n,str);

System.out.println("The length of LCS is: " + length);

for(String s : set) {

System.out.println(s);

}

}

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

System.out.print("请输入字符串S1：");

String test1=input.nextLine();

System.out.print("请输入字符串S2：");

String test2=input.nextLine();

lcscommen lcs = new lcscommen(test1,test2);

lcs.printLCS();

}

}