# 华东师范大学数据科学与工程学院上机实践报告

**课程名称：**算法设计与分析　　**年级：**19 级　　**上机实践成绩：**

**指导教师：**金澈清　　**姓名：**龚敬洋

**上机实践名称：**最长公共子序列　　**学号：**10195501436　　**上机实践日期：**2020/11/20

**上机实践编号：**No.7　　**组号：**1-436

## 一、目的

1. 熟悉算法设计的基本思想
2. 掌握计算最长公共子序列的方法

## 二、内容与设计思想

1. 编写随机整数生成算法，生成 0 到 9 范围内的 N 个随机整数并输出；
2. 编写计算最长公共子序列方法的代码；
3. 随机生成两组范围为 0 到 9 的 5、50、500、5000 个随机整数，并求两组整数的最长公共子序列
4. 随机生成一组范围为 0 到 9 的 5000 个随机整数和另一组范围为 0 到 9 的 5、50、500、5000 个随机整数，并求两组整数的最长公共子序列

## 三、使用环境

推荐使用 C/C++集成编译环境。

## 四、实验过程

*1. 写出计算最长公共子序列方法的代码*

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
int d[5005][5005];
int main(){
    ifstream fin1("data1.txt");
    ifstream fin2("data2.txt");
    clock_t start, stop;
    int a[5005], b[5005], r[5005], n1 = 0, n2 = 0, p, q, cnt;
    cout<<"原始序列 1: ";
    while (!fin1.eof()){
        fin1>>a[n1 + 1];
        n1++;
    }
    for(int i = 1; i <= n1 - 1; i++) cout<<a[i]<<" ";
    cout<<endl;
    cout<<"原始序列 2: ";
    while(!fin2.eof()){
        fin2>>b[n2 + 1];
        n2++;
```

```
    }
    for(int i = 1; i <= n2 - 1; i++) cout<<b[i]<<" ";
    cout<<endl;
    n1--;
    n2--;
    start = clock();
    for(int i = 0; i <= n1; i++)
        for(int j = 0; j <= n2; j++){
            if(i == 0 || j == 0) d[i][j] = 0;
            else{
                if(a[i] == b[j]) d[i][j] = d[i - 1][j - 1] + 1;
                else d[i][j] = max(d[i][j - 1], d[i - 1][j]);
            }
        }
    p = n1;
    q = n2;
    cnt = d[n1][n2] - 1;
    while (p > 0 && q > 0){
        if(d[p][q] == (d[p - 1][q - 1] + 1) && a[p] == b[q]){
            r[cnt] = a[p];
            cnt--;
            p = p - 1;
            q = q - 1;
            continue;
        }
        if(d[p][q-1] == d[p - 1][q]){
            q = q - 1;
        }
        else if(d[p][q] == d[p][q - 1]){
            q = q - 1;
        }
        else{
            p = p - 1;
        }
    }
    stop = clock();
    cout<<"最长公共子序列: ";
    for(int i = 0; i < d[n1][n2]; i++) cout<<r[i]<<" ";
    cout<<endl;
    cout<<"Total time: "<<(double)(stop - start) / CLOCKS_PER_SEC<<endl;
    fin1.close();
    fin2.close();
    return 0;
}
```

2. *将各个实验的代码运行结果进行截图*

序列长度均为 5 时的运行结果：

```
原始序列1: 0 0 8 2 5
原始序列2: 9 4 3 3 2
最长公共子序列: 2
Total time: 4.2e-05
```

序列长度均为 50 时的运行结果：

```
原始序列1: 0 5 8 6 1 6 0 4 4 6 6 0 5 1 8 5 9 6 2 5 3 7 8 8 8 7 6 8 5 5 8 1 4 8 6 4 6 8 2 4 0 0 5 6 8 2 7 3 5 3
原始序列2: 5 8 6 1 6 4 0 7 8 6 4 1 0 8 1 6 3 6 9 5 9 9 4 8 7 2 0 6 7 0 1 3 9 7 0 2 4 8 5 3 7 6 7 8 3 7 2 8 7 4
最长公共子序列: 5 8 6 1 6 4 0 7 8 6 4 8 6 6 8 2 0 0 5 6 8 2 7
Total time: 0.000157
```

序列长度均为 500 时的运行时间：

```
Total time: 0.003804

Process finished with exit code 0
```

序列长度均为 5000 时的运行时间：

```
Total time: 0.325517

Process finished with exit code 0
```

序列长度为 5000、5 时的运行结果：

```
原始序列1: 6 3 3 9 8 7 9 8 8 2 7 3 0 3 3 3 0 3 3 4 1 4 2 9 7 9 9 4 4 4 0 2 6 1 1 7 1 4 3 0 2 7 0 2 4 7 1 2 1 6 2 0 1 3 6 3 5 0 4 1 7 2 9 4 9 8 1 8
原始序列2: 9 8 8 6 6
最长公共子序列: 9 8 8 6 6
Total time: 0.007618
```

序列长度为 5000、50 时的运行结果：

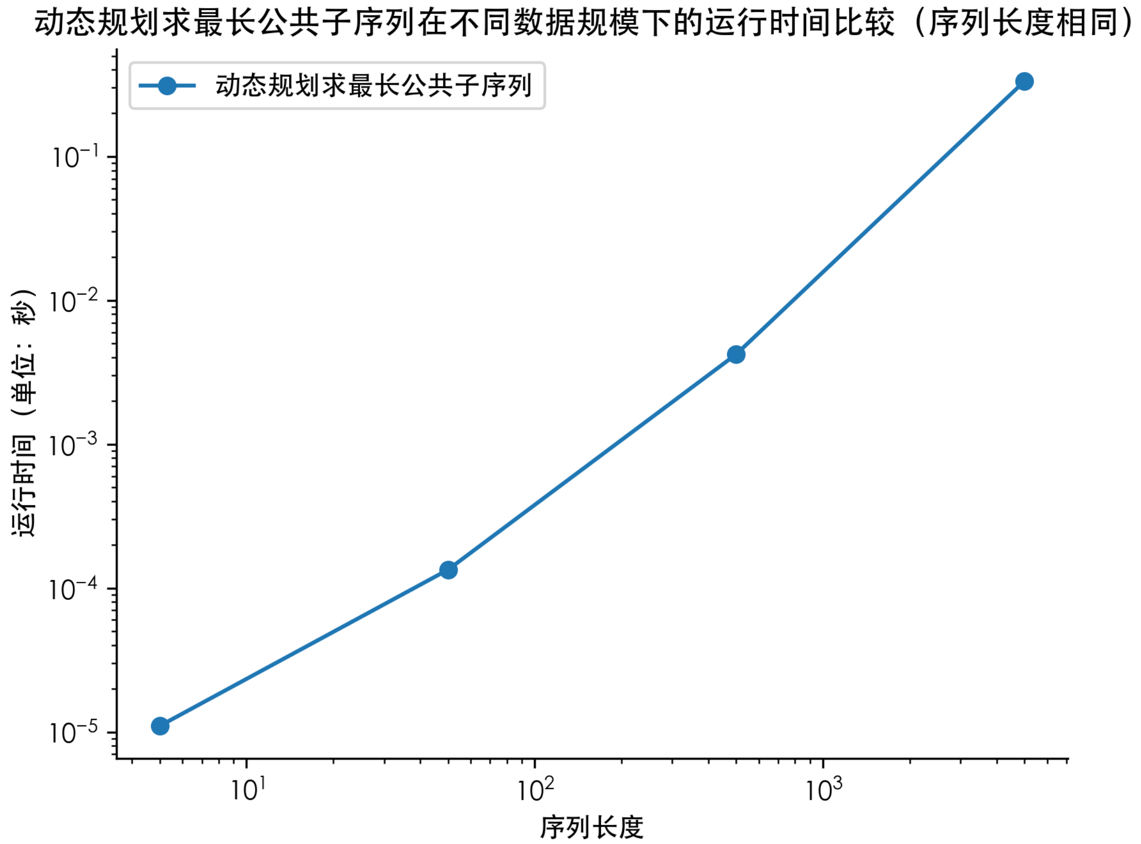
```
原始序列1: 7 1 2 8 2 9 7 0 5 6 7 2 8 2 0 6 9 0 6 2 6 9 2 7 2 2 8 3 8 3 5 0 4 1 8 7 0 1 6 0 2 9 3 0 4 4 7 6 8 6 8 4 1 7 7 0 5 9 8 2 7
原始序列2: 0 4 9 1 9 2 9 6 7 2 8 8 5 4 4 3 3 9 0 4 1 8 8 2 6 7 1 5 1 7 1 0 4 5 6 3 1 9 9 3 6 0 9 8 4 1 9 5 9 7
最长公共子序列: 0 4 9 1 9 2 9 6 7 2 8 8 5 4 4 3 3 9 0 4 1 8 8 2 6 7 1 5 1 7 1 0 4 5 6 3 1 9 9 3 6 0 9 8 4 1 9 5 9 7
Total time: 0.012788
```

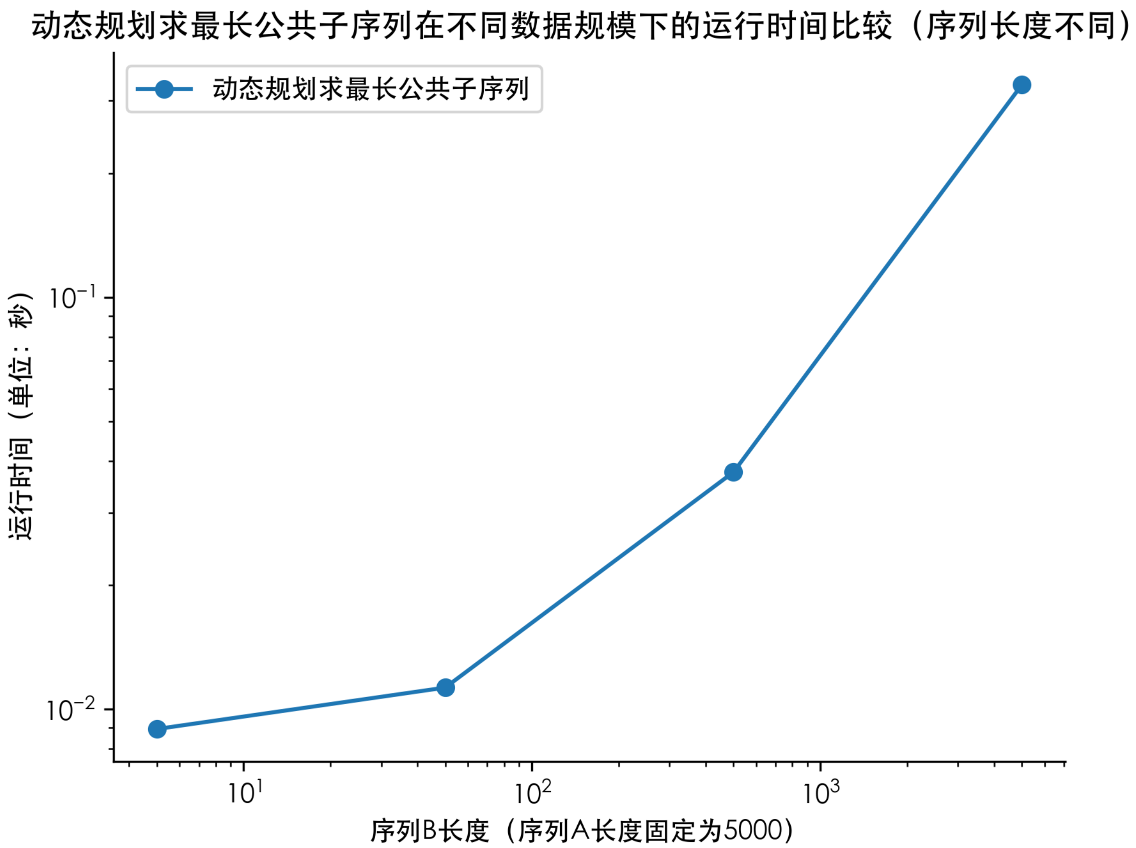
序列长度为 5000、500 时的运行时间：

```
Total time: 0.039047

Process finished with exit code 0
```

*3.　分别画出各个实验结果的折线图*


动态规划求最长公共子序列在不同数据规模下的运行时间比较（序列长度相同）
动态规划求最长公共子序列
运行时间（单位：秒）
$10^{-1}$
$10^{-2}$
$10^{-3}$
$10^{-4}$
$10^{-5}$
$10^{1}$
$10^{2}$
$10^{3}$
序列长度


动态规划求最长公共子序列在不同数据规模下的运行时间比较（序列长度不同）
动态规划求最长公共子序列
运行时间（单位：秒）
$10^{-1}$
$10^{-2}$
$10^{1}$
$10^{2}$
$10^{3}$
序列B长度（序列A长度固定为5000）

## 五、总结

对上机实践结果进行分析，问题回答，上机的心得体会及改进意见。

使用动态规划算法求两个长度分别为$m$和$n$的序列的公共子序列的时间复杂度为$\Theta(mn)$，回溯打印公共子序列串的时间复杂度为$\Theta(m+n)$，故总的时间复杂度为$\Theta(mn)$。在实验中，当$m=n$时，算法的时间复杂度始终为$\Theta(n^2)$，故在对数坐标下运行时间呈线性变化。而当$m$从远小于$n$逐渐增大到$n$时，算法的时间复杂度从$\Theta(n)$逐渐变化为$\Theta(n^2)$，故在对数坐标下运行时间曲线的斜率逐渐增大。综上所述，实验结果与理论基本吻合。