实验一：动态规划

PB19030888张舒恒

实验设备和环境

PC一台，Win11企业版操作系统，gcc 9.1.0编译器，Clion 2021.2.2代码编辑器，Excel绘图工具

实验内容

1.矩阵链乘最优方案

2.所有最长公共子序列

实验要求

代码限制C/C++，建立根文件夹80-张舒恒-PB19030888-project1，在根文件夹下建立本实验报告，ex1和ex2实验文件夹，每个实验文件夹中建立3个子文件夹：input文件夹：存放输入数据，src文件夹：源程序，output文件夹：输出数据。

实验步骤及方法

矩阵链乘

1.文件输入输出

采用绝对路径的输入输出流

```
string from = "C:\\Users\\ASUS\\Desktop\\AL\\ex1\\input\\1_1_input.txt";
string dest = "C:\\Users\\ASUS\\Desktop\\AL\\ex1\\output\\result.txt";
string time = "C:\\Users\\ASUS\\Desktop\\AL\\ex1\\output\\time.txt";
ifstream file_in;
file_in.open(from);
file_out.open(dest);
time_out.open(time);
```

2.矩阵链乘算法

定时器启动，动态规划法求解矩阵链乘的最优解，定时器关闭

```
chrono::steady_clock::time_point t1 = chrono::steady_clock::now();
for(int i = 1; i<= N; i++)
    dp[i][i] = 0;
for (int len = 2; len <= N; len++) {
    for (int i = 1; i <= N - len + 1; i++) {
        int j = i + len - 1;
        dp[i][j] = 9223372036854775807;
        for (int k = i; k < j; k++) {
            long long tmp = dp[i][k] + dp[k + 1][j] + nums[i-1] * nums[k] *
nums[j];
            if(tmp < dp[i][j]){
```

```
                dp[i][j] = tmp;
                s[i][j] = k;
            }
        }
    }
}
chrono::steady_clock::time_point t2 = chrono::steady_clock::now();
```

3.打印矩阵链乘的最优解

采用递归的方式打印

```
void print(int s[][150], int i, int j){
    if(i == j)
        file_out << "A" << i;
    else{
        file_out << "(";
        print(s, i, s[i][j]);
        print(s, s[i][j] + 1, j);
        file_out << ")";
    }
}
```

4.打印运行时间和N=5时动态规划表，画出时间曲线并分析

```
C:\Users\ASUS\Desktop\AL\cmake-build-debug\AL.exe
m[1][1]=0 m[1][2]=15903764653528 m[1][3]=74062781976714 m[1][4]=128049683226820 m[1][5]=154865959097238
m[2][2]=0 m[2][3]=43981152513978 m[2][4]=105723424955724 m[2][5]=138766801119366
m[3][3]=0 m[3][4]=119490227350806 m[3][5]=183439291324068
m[4][4]=0 m[4][5]=120958281818244
m[5][5]=0
s[1][2]=1 s[1][3]=1 s[1][4]=1 s[1][5]=1
s[2][3]=2 s[2][4]=3 s[2][5]=4
s[3][4]=3 s[3][5]=4
s[4][5]=4
```

最长公共子序列

1.文件输入输出

采用绝对路径的输入输出流

```
string from = "C:\\Users\\ASUS\\Desktop\\AL\\ex2\\input\\1_2_input.txt";
string time = "C:\\Users\\ASUS\\Desktop\\AL\\ex2\\output\\time.txt";
ifstream file_in;
file_in.open(from);
ofstream time_out;
time_out.open(time);
```

2.最长公共子序列算法

定时器启动，动态规划法求解最长公共子序列，定时器关闭

```
chrono::steady_clock::time_point t1 = chrono::steady_clock::now();
int length = lcs_length(m, n);
chrono::steady_clock::time_point t2 = chrono::steady_clock::now();
```

```
int lcs_length(int m, int n){
    c = vector<vector<int> >(m+1,vector<int>(n+1));
    for(int i=0; i<m+1; ++i){
        for(int j=0; j<n+1; ++j){
            if (i == 0 || j == 0)
                c[i][j] = 0;
            else if(X[i-1] == Y[j-1])
                c[i][j] = c[i-1][j-1] + 1;
            else
                c[i][j] = max(c[i-1][j], c[i][j-1]);
        }
    }
    return c[m][n];
}
```

4.打印所有最长公共子序列及其个数（不允许重复）

使用`set<string> lcs;`保存所有最长公共子序列，这样是按照不允许重复的方式进行计数的

```
string str;
lcs_print(m, n, str);
ofstream file_out;
file_out.open("C:\\Users\\ASUS\\Desktop\\AL\\ex2\\output\\result_" +
to_string(N) + ".txt");
file_out << "The number of LCSs is " << lcs.size() << endl;
set<string>::iterator it = lcs.begin();
for( ; it!=lcs.end(); it++)
    file_out << *it << endl;
lcs.clear();
```

```
void lcs_print(int i, int j, string lcs_str){
    while (i>0 && j>0){
        if (X[i-1] == Y[j-1]){
            lcs_str.push_back(X[i-1]);
            --i;
            --j;
        }
        else{
            if (c[i-1][j] > c[i][j-1])
                --i;
            else if (c[i-1][j] < c[i][j-1])
                --j;
            else{
                lcs_print(i-1, j, lcs_str);
                lcs_print(i, j-1, lcs_str);
                return;
            }
        }
    }
    reverse(lcs_str.begin(),lcs_str.end());
    lcs.insert(lcs_str);
```

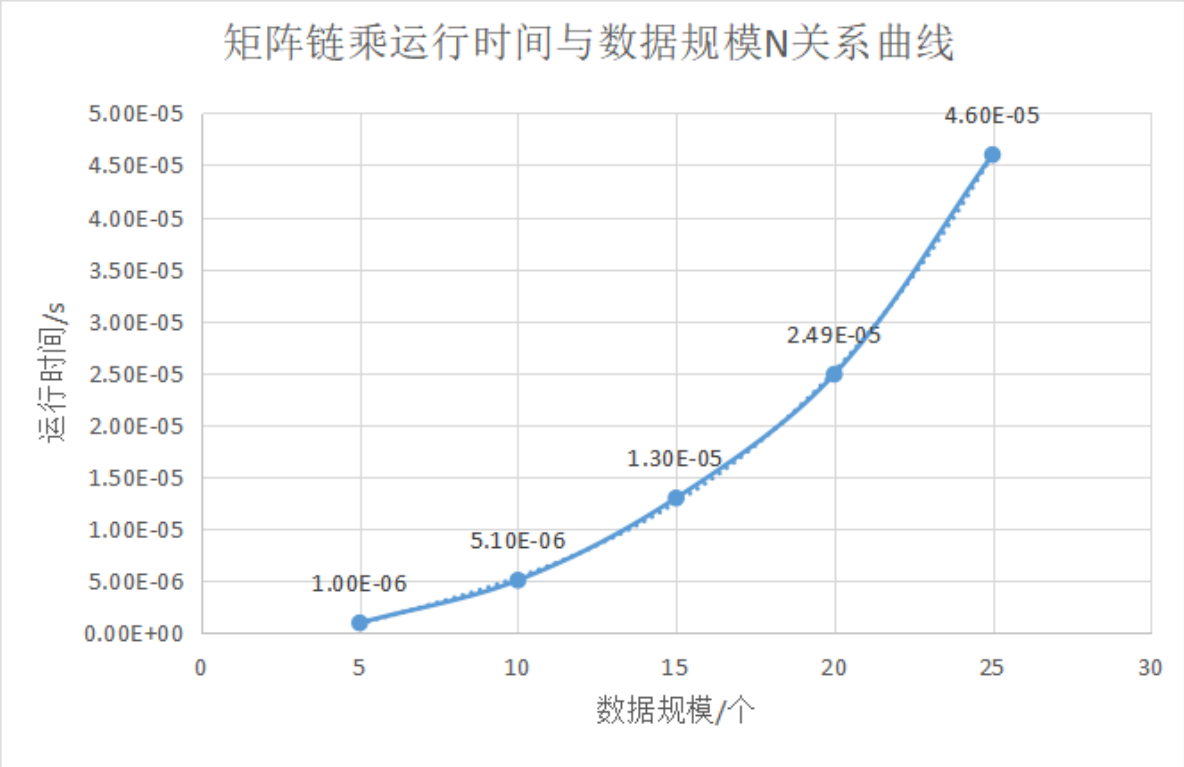
```
}
```

5.打印运行时间画出时间曲线并分析

实验结果分析

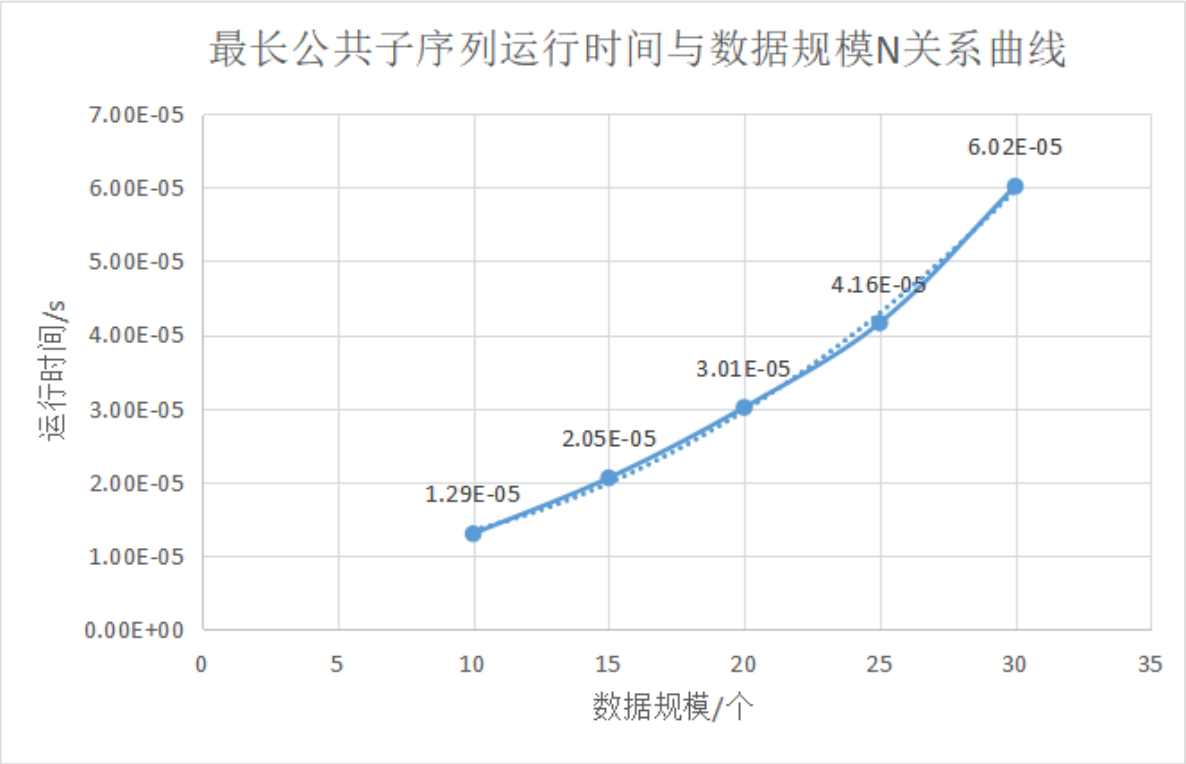
矩阵链乘

画出时间曲线，用三次函数进行拟合，结果基本符合三次函数增长模型，所以实际时间复杂度和理论时间复杂度近似相同，均为$O(n^3)$



最长公共子序列

画出时间曲线，用二次函数进行拟合，结果基本符合二次函数增长模型，N=25时有少许偏差可能是该样例相比其他样例在判断和计算上略微简单，但误差在可接受范围内，所以实际时间复杂度和理论时间复杂度近似相同，均为$O(n^2)$



实验总结

通过本次实验我基本掌握了动态规划算法解决实际问题的一般思路