# 实验二 动态规划

*袁雨 PB20151804*

## 一、实验设备和环境

### 1. 实验设备

设备：HUAWEI  MateBook X Pro

处理器：Intel(R) Core(TM) i5-10210U CPU @1.60GHz 2.11 GHz

### 2. 实验环境

vscode, gcc

## 二、实验内容和要求

### 1. 实验内容

**实验2.1：求矩阵链乘最优方案**

- n个矩阵链乘，求最优链乘方案，使链乘过程中乘法运算次数最少。
- n的取值5, 10, 15, 20, 25，矩阵大小见2_1_input.txt。
- 求最优链乘方案及最少乘法运算次数，记录运行时间，画出曲线分析。
- 仿照P214 图15-5，打印n=5时的结果并截图。
- 提示：
  - 考虑4B int类型，上限2147483647；8B long long类型，上限9,223,372,036,854,775,807。
  - 计算过程，所给数据求出的乘法运算次数变量可能超出int类型，但在long long范围内。

**实验2.2：求最长公共子序列**

- 给定两个序列X、Y，求出这两个序列的最长公共子序列（某一个即可）。
- X, Y序列由A、B、C、D四种字符构成,序列长度分别取10、15、20、25、30，见2_2_input.txt。
- 打印最长公共子序列，记录运行时间，画出曲线分析。

### 2.实验要求

**编程要求**

- C/C++

**目录格式**

- 实验需建立根文件夹，文件夹名称为：编号-姓名-学号-project2，在根文件夹下需包括实验报告和ex1、ex2实验文件夹，每个实验文件夹包含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2.1 矩阵链乘 输入输出

- ex1/input/2_1_input.txt（已给出）：
  - 每个规模的数据占两行：
    - n
    - 矩阵大小向量p=(p_0, p_1,..., p_n)，矩阵A_i大小为p_i−1∗p_i
- ex1/output/
  - result.txt：每个规模的结果占两行
    - 最少乘法运算次数
    - 最优链乘方案（要求输出括号化方案，参考P215 print_opt_parens算法）
  - time.txt：每个规模的运行时间占一行
- 同行数据间用空格隔开

#### 实验2.2 最长公共子序列 输入输出

- ex2/input/2_2_input.txt（已给出）：
  - 每个规模的数据占三行：
    - n：X、Y序列长度
    - X：X序列
    - Y：Y序列
- ex2/output/
  - result_i.txt：X、Y序列长度为i的结果
    - 最长公共子序列长度
    - 最长公共子序列
  - time.txt：每个规模的运行时间占一行

**实验报告**

- 实验设备和环境、实验内容及要求、方法和步骤、结果与分析。
- 比较实际复杂度和理论复杂度是否相同，给出分析。

# 三、实验方法和步骤

#### 实验2.1 求矩阵链乘最优方案

令 $m[i,j]$ 表示计算矩阵 $A_{i.j}$ 所需标量乘法次数的最小值。

该问题为动态规划问题：

$$m[i,j]=\begin{cases}0 & \text{如果 } i=j\\ \min_{i\leqslant k<j}\{m[i,k]+m[k+1,j]+p_{i-1}p_kp_j\} & \text{如果 } i<j\end{cases}$$

- 参考教材矩阵链乘法部分编写函数：

- `void MATRIX_CHAIN_ORDER(long long *p, int n, FILE *fp);`

    函数中`m[i][j]`的值给出了子问题最优解的代价，`s[i][j]`保存最优括号化方案的分割点位置 k，即使得`m[i][j]=m[i][k]+ m[k+1][j]+p[i-1]p[k]P[j]`成立的值。

- `void FPRINT_OPTIMAL_PARENS(long long *s, int n, int i, int j, FILE *fp);`

    将教材中的输出到屏幕，即`printf()`，改成输出到文件fp，即`fprintf()`。

- 在`main`函数中：
    - 使用`fopen()`、`fscanf()`、`fprintf()`、`fclose()`等函数完成对文件的打开、读写、关闭等操作。
    - 使用`QueryPerformance()`等函数完成计时。对输入数据的存取、初始化等不计入运行时间。
    - 调用`void MATRIX_CHAIN_ORDER(long long *p, int n, FILE *fp);`
- 使用Excel进行数据处理，作出各算法在不同输入规模下的运行时间曲线图，并对结果进行分析讨论。

#### 实验2.2 求最长公共子序列

定义 $c[i,j]$ 表示 $X_i$ 和 $Y_j$ 的 LCS 的长度。

该问题为动态规划问题：

$$c[i,j]=\begin{cases}0 & \text{若 } i=0 \text{ 或 } j=0\\ c[i-1,j-1]+1 & \text{若 } i,j>0 \text{ 且 } x_i=y_j\\ \max(c[i,j-1],c[i-1,j]) & \text{若 } i,j>0 \text{ 且 } x_i\neq y_j\end{cases}$$

- 参考教材最长公共子序列部分编写函数：
    - `void LCS_LENGTH(char *x,char *y,int m,int n,FILE *fp);`

        将`c[i][j]`的值保存在二维数组`c`中，并按行主次序计算表项。过程还维护一个二维数组`b`，帮助构造最优解。`b[i][j]`指向的表项对应计算`c[i][j]`时所选择的子问题最优解，以-1表示书中的`←`，0表示`↖`，1表示`↑`。

    - `void FPRINT_LCS(int *b, char *x, int m, int i, int j ,FILE *fp);`

        将教材中的输出到屏幕，即`printf()`，改成输出到文件fp，即`fprintf()`。

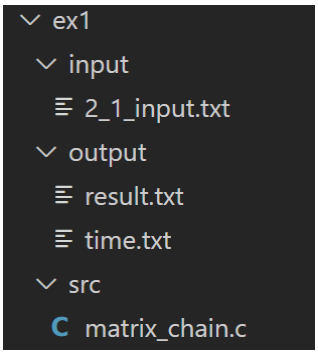
- 在`main`函数中：
    - 使用`fopen()`、`fscanf()`、`fprintf()`、`fclose()`等函数完成对文件的打开、读写、关闭等操作。
    - 使用`QueryPerformance()`等函数完成计时。对输入数据的存取、初始化等不计入运行时间。
    - 调用`void LCS_LENGTH(char *x,char *y,int m,int n,FILE *fp);`
- 使用Excel进行数据处理，作出各算法在不同输入规模下的运行时间曲线图，并对结果进行分析讨论。

# 四、实验结果与分析

## 实验2.1 求矩阵链乘最优方案

### 1.实验结果

- 目录结构：




- result、time

| n | result | time(s) |
|---|---|---|
| 5 | 154865959097238<br>(A1(((A2A3)A4)A5)) | 0.000008 |
| 10 | 42524697503391<br>((A1A2)(((((((A3A4)A5)A6)A7)A8)A9)A10)) | 0.000005 |
| 15 | 5400945319618<br>((((((((((((((A1A2)A3)A4)A5)A6)A7)A8)A9)A10)A11)A12)A13)A14)A15) | 0.000009 |
| 20 | 319329979644400<br>((A1(A2(A3(A4(A5(A6(A7(A8(A9(A10(A11(A12(A13(A14A15))))))))))))))<br>((((A16A17)A18)A19)A20)) | 0.000017 |
| 25 | 574911761218280<br>((A1(A2(A3(A4(A5(A6(A7(A8(A9(A10A11)))))))))))<br>((((((((((((A12A13)A14)A15)A16)A17)A18)A19)A20)A21)A22)A23)A24)A25)) | 0.000030 |

- n=5时的`m`和`s`：

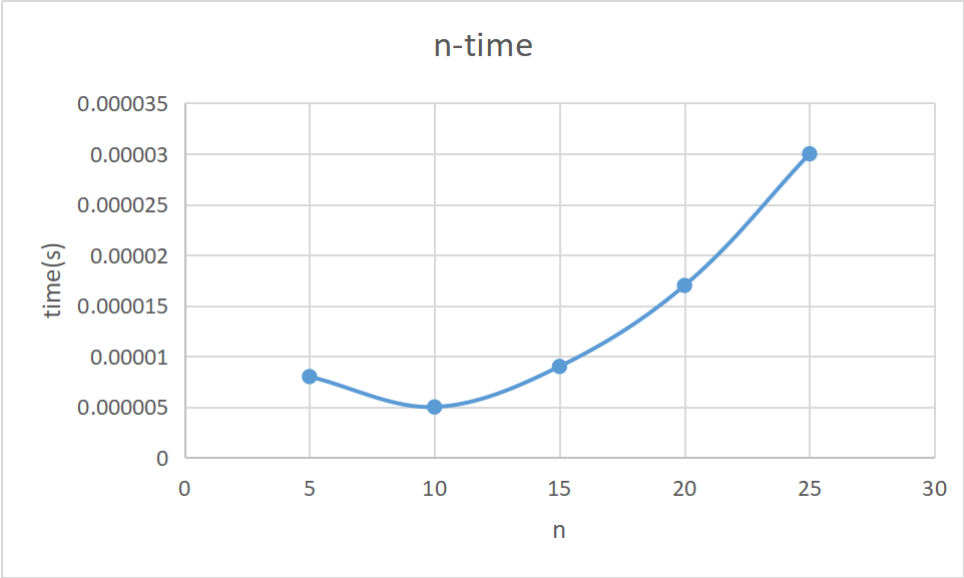
```
n=5:

m
i\j     5               4               3               2               1
5       0
4       120958281818244 0
3       183439291324068 119490227350806 0
2       138766801119366 105723424955724 43981152513978  0
1       154865959097238 128049683226820 74062781976714  15903764653528  0

s
i\j 5   4   3   2
4   4
3   4   3
2   4   3   2
1   1   1   1   1
```

### 2.实验分析

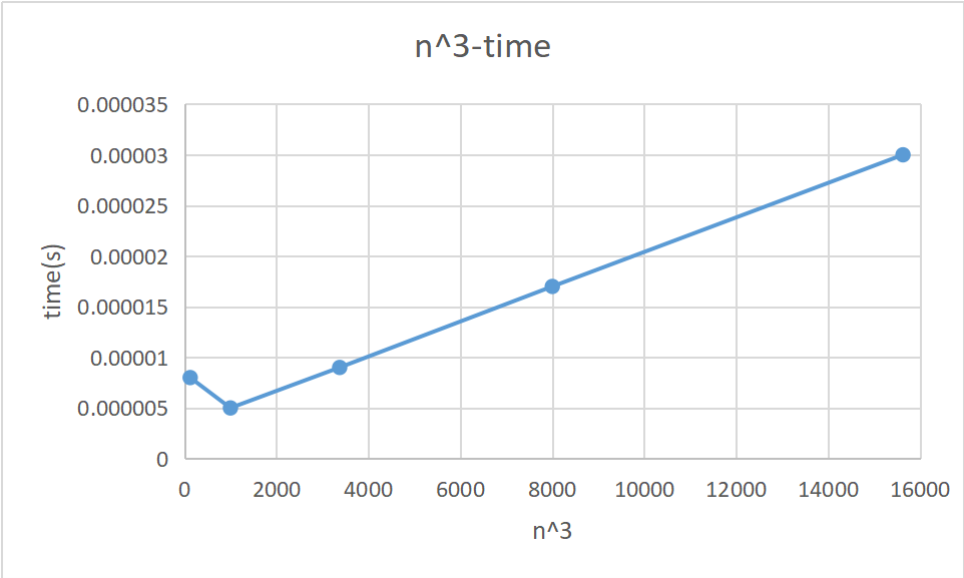
以数据规模n为横轴，程序运行时间time为纵轴，作图如下：




可见其近似为多项式时间。

矩阵链乘算法的理论时间复杂度为$\Omega\left(n^3\right)$。

以$n^3$为横轴，程序运行时间time为纵轴，作图如下：



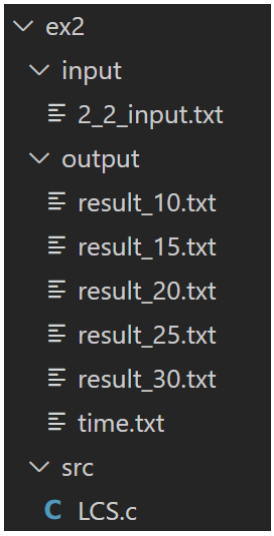

可见除了第一个数据规模，其他的数据规模$n^3$与运行时间time成线性关系，符合理论复杂度。

而第一个数据规模的异常原因应该是在之后的运行中，编译器对变量、数组的初始化等进行了优化（在实验一中验证过）。

## 实验2.2 求最长公共子序列

### 1.实验结果

- 目录结构

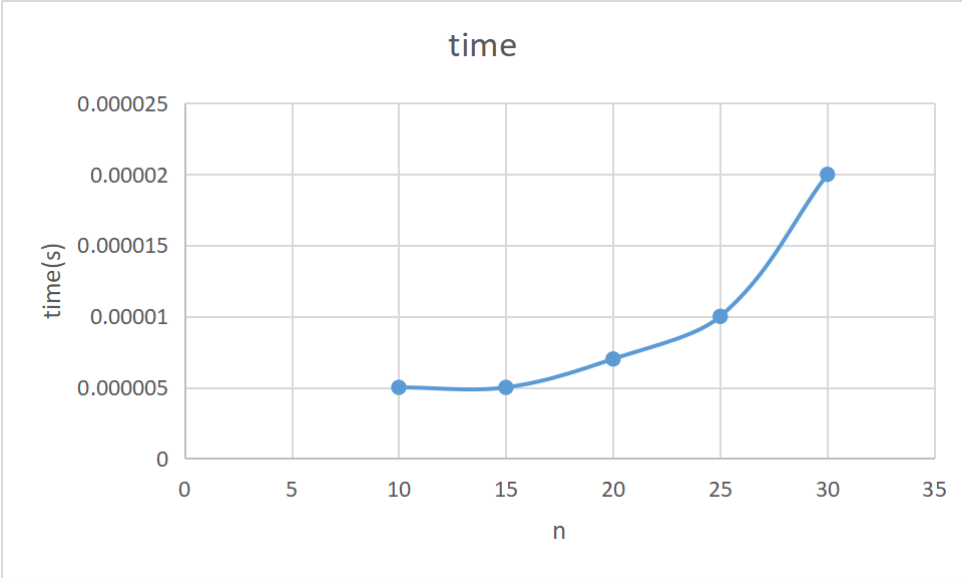



- result、time

| n | result | time(s) |
|---|---|---|
| 10 | 5<br>CAABA | 0.000005 |
| 15 | 8<br>BADBCCCD | 0.000005 |
| 20 | 12<br>BACAAADCABAA | 0.000007 |
| 25 | 14<br>DCBABDDBDCCBDD | 0.000010 |
| 30 | 16<br>ADDBBCDBBCDDDCBD | 0.000020 |

**2.结果分析**

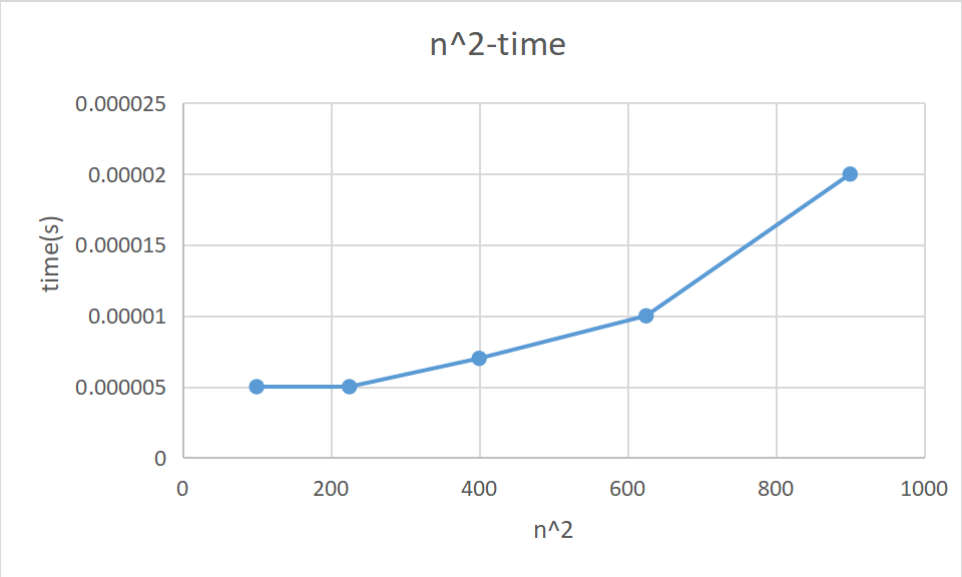
以数据规模n为横轴，程序运行时间time为纵轴，作图如下：




可见其近似为多项式时间。

LCS 算法运行时间为 $O(mn)$，因为本实验中$m = n$，故理论时间复杂度为$O(n^2)$。

以$n^2$为横轴，程序运行时间time为纵轴，作图如下：




可见数据规模$n^2$与运行时间time的关系与线性有一定差距。与矩阵链乘同理，编译器对初始化等进行了优化，此外还可能受其他的实验环境、理论时间复杂度的低阶项等影响。