Project2动态规划实验报告

徐煜森 PB16110173

1. **实验要求**

实验1：实现矩阵链乘问题的求解算法。对n的取值分别为：5、10、20、30 ，随机生成 n+1 个整数值（ p0、p1、…、pn ）代表矩阵的规模，其中第i 个矩阵(1≤i ≤ n)的规模为pi-1 × pi ，用动态规划法求出矩阵链乘问题的最优乘法次序，统计算法运行所需时间，画出时间曲线，进行性能分析。

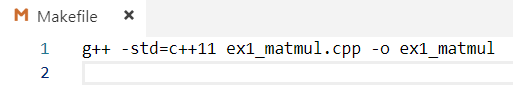
实验2：实现最长公共子序列问题的求解算法。序列X的长为m，序列Y的长为n，序列X和Y的元素从26个大写字母中随机生成，m和n的取值：

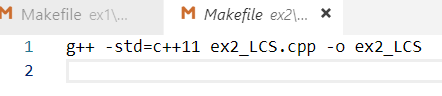
第1组：(15，10)，(15，20)，(15，30)，(15，40)，(15，50)，(15，60)

第2组：(15，25)，(30，25)，(45，25)，(60，25)，(75，25)，(90，25)

给出算法运行所需的时间，画出时间曲线 ，进行性能分析。

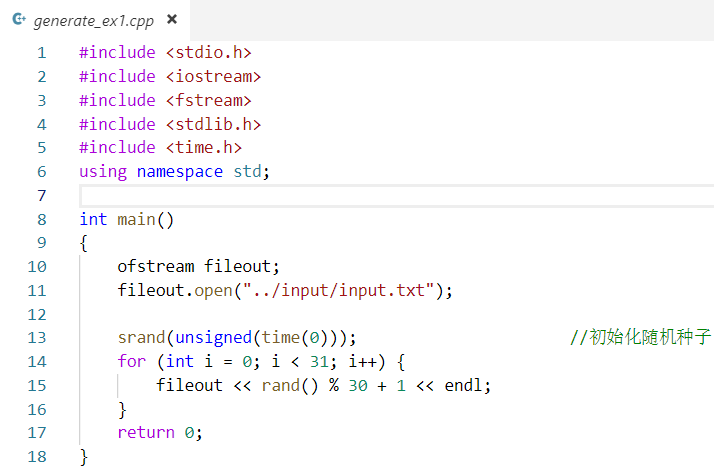
1. **实验环境**
2. Windows10 64位x86，机器内存8G，时钟主频2.59GHz
3. 软件环境：Visual Studio 2017
4. **实验过程**
5. **Makefile**





为方便助教在Linux环境下进行测试，为两个实验写了两个Makefile。不过我实验是在Windows 10下使用visual studio 2017做的。

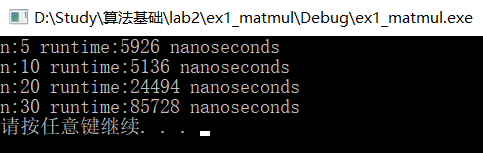
1. **Ex1**
   1. 生成数据



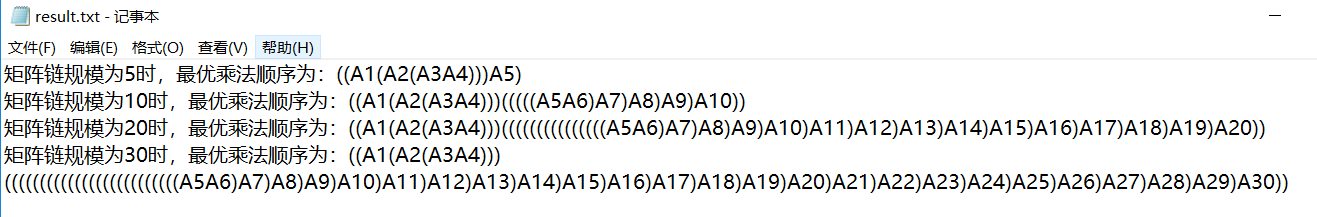
使用rand()%30 + 1生成[1, 30]区间的数据，每行一个数据，共31行。

* 1. 矩阵链乘的动态规划算法实现与课堂上介绍的一致，将在代码截图与解析中说明。
  2. 实验大体测试时间的框架与project 1相同，更改了输入输出部分和算法。
  3. 实验结果截图

屏幕输出的运行时间（运行时间也按要求输出至文件）：

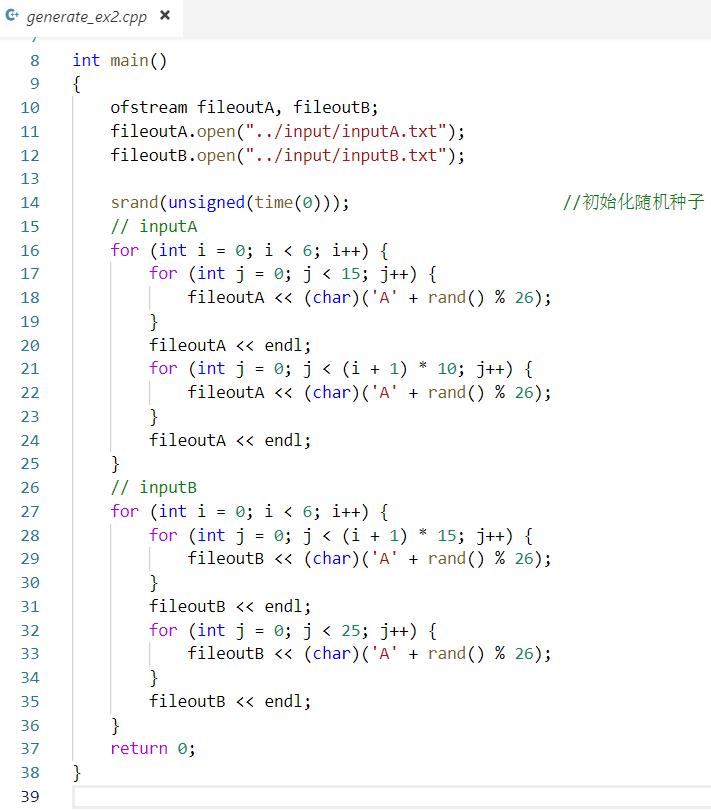


计算结果输出：



1. **Ex2**
   1. 生成数据

按要求生成A、B两组数据，每组六对，具体规模见实验要求。



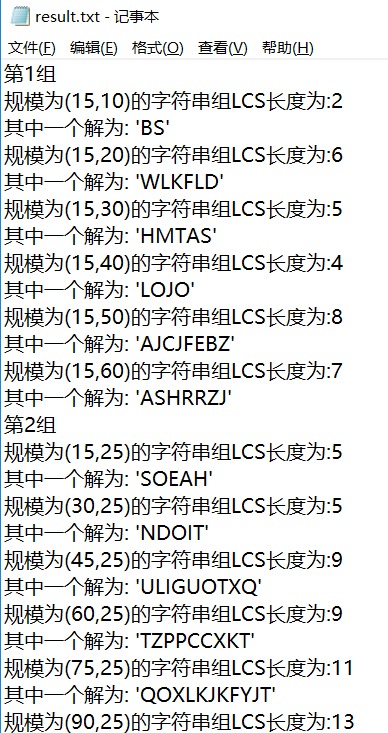
* 1. 最长公共子序列的动态规划算法实现与课堂上介绍的一致，将在代码截图与解析中说明
  2. 实验框架与实验1相同，更改了输入输出和算法。
  3. 实验结果截图

屏幕输出的运行时间（运行时间也按要求输出至文件）：

其中i表示组，j表示对，如i = 1，j = 1表示第1组第一对数据。

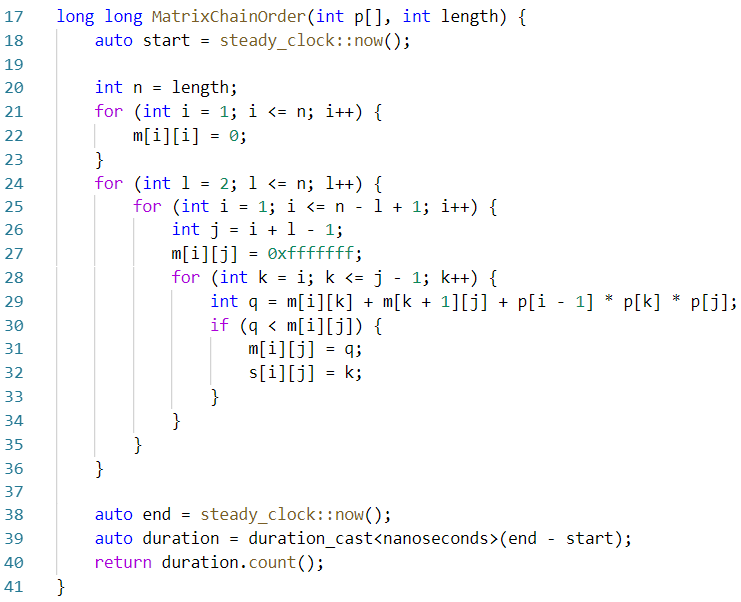


计算结果输出：



1. **关键代码截图**
2. **Ex1**

动态规划算法：

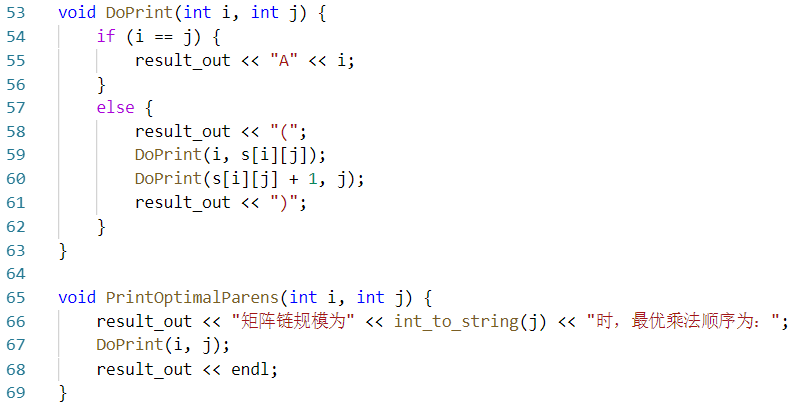


可以看到动态规划算法与课本上一致，其中值得注意的是：

1. 实验中p，m，s均为全局数组，在算法执行前需要进行必要的初始化。
2. 可以看出这里令n = length，而课本上伪代码中令n = length – 1。原因是我的实现中length为矩阵的个数，而课本上伪代码的length为数组的长度，即为矩阵个数 + 1。

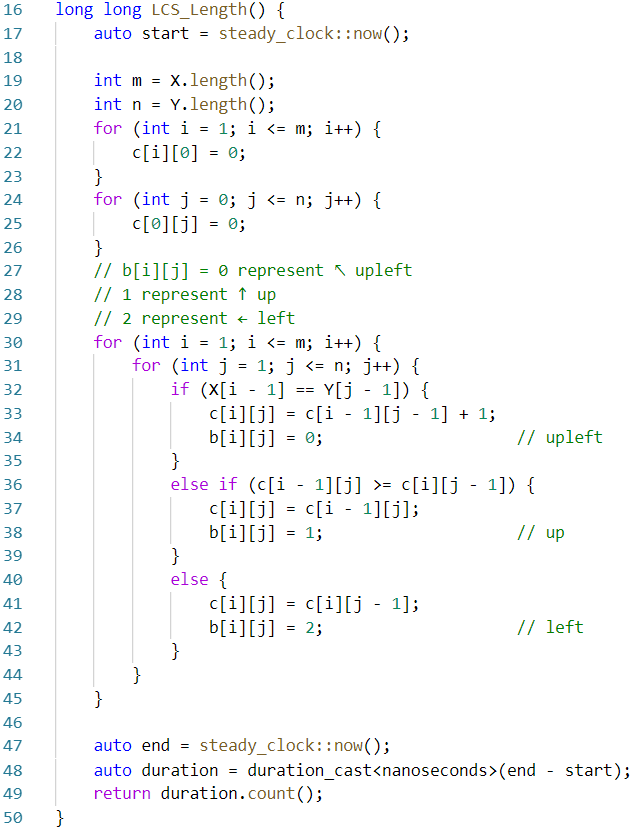
结果输出：

输出的方法与课本一致，均为递归调用输出。



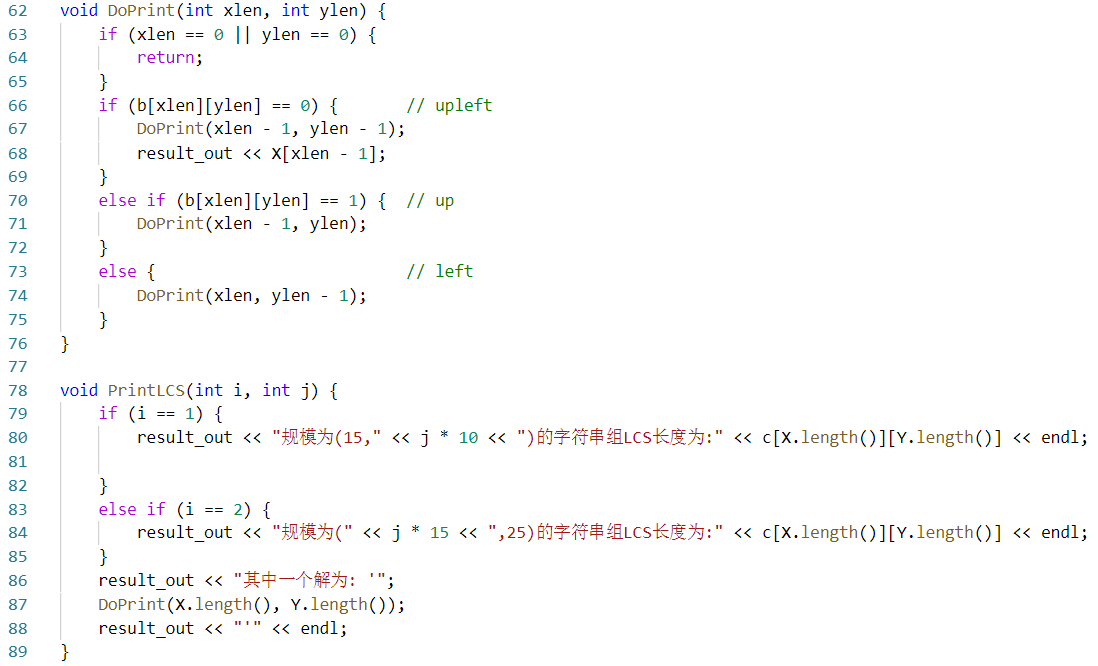
1. **Ex2**

动态规划算法：



动态规划算法与课本上一致，其中使用0,1,2分别代表左上↖，上↑，左←。

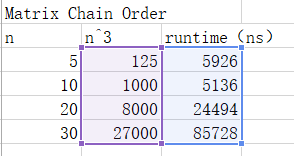
结果输出：



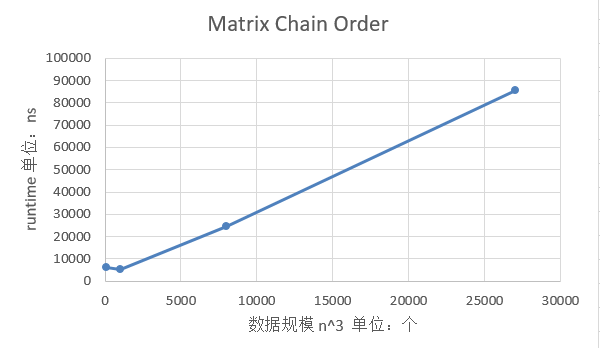
按照要求格式进行输出，输出LCS的方法与课本一致为递归输出。

1. **实验结果及分析**
2. **Ex1**

实验数据如下：



画出折线图如下：

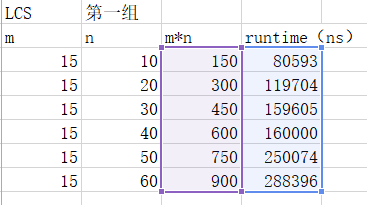


分析：

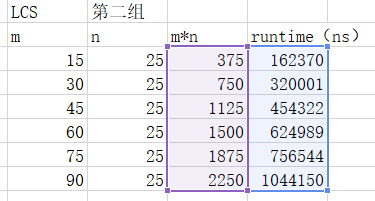
1. 发现运行时间与数据规模n^3基本成线性关系，符合期望Ο(n^3)。
2. 发现在n=5时运行时间反而比n=10时长，分析原因可能是cache存在的缘故。在n=5首次运算时各个数组被调入cache，n=10时发现大部分需要用到的数据都可以在cache中找到，减少了从内存调入cache的时间。
3. **Ex2**

实验数据如下：

第一组：

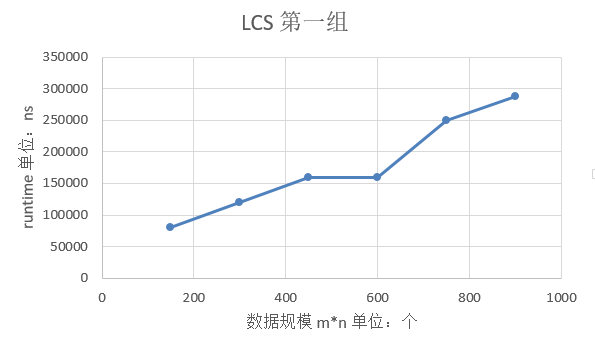


第二组：

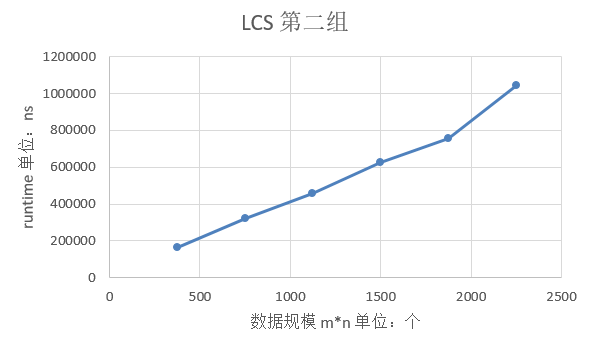


画出折线图：

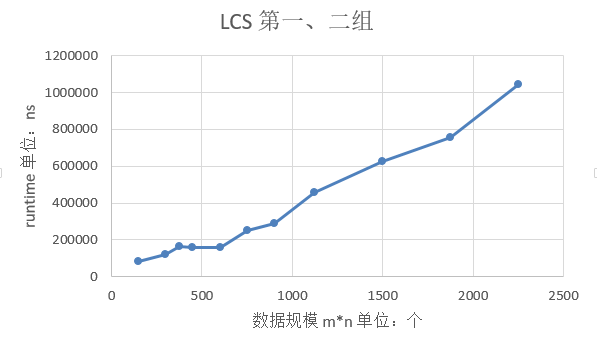
第一组：



第二组：



将两组数据画在同一张图上：



分析：

1. 综合看来，运行时间与数据规模m\*n基本成线性关系，符合期望Ο(m\*n)。
2. 从第一组和第二组图中可以看出，除第一组图中有波动外，运行时间几乎与m\*n成完美的线性关系。而将第一组和第二组的数据画在一张图上时，可以看出在m\*n较小时，运行时间有明显波动。分析原因可能是在m\*n较小时，对数组的初始化消耗时间不可忽略，导致运行时间的常数因子有所变化。
3. 当m\*n较大时，运行时间基本稳定与m\*n成线性关系。
4. **实验总结**

本次实验中有些重要的小问题，就如在Ex1的实验过程中提到的n=length与课本不一致的问题。看似是个小疏忽，实际上反映出自己对算法的掌握仍有不足，对算法没有理解到位，值得反思。