算法基础project1-实验报告

廖佳怡PB19151776

1. 实验设备和环境

编译环境：Windows10

编程语言：C++

机器内存：16.0GB

时钟主频：2.30GHz

2. 实验内容及要求

- 求矩阵链乘最优方案
  - 内容：

    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1_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。

  - 要求：

    - ex1/input/1_1_input.txt（已给出）：
      - 每个规模的数据占两行：
        - n
        - 矩阵大小向量$p = (p_0, p_1, \dots, p_n)$，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

- 求所有最长公共子序列
  - 内容：

    给定两个序列X、Y，求出这两个序列的所有最长公共子序列。
    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1_2_input.txt。
    输出所有最长公共子序列个数，并打印所有最长公共子序列，记录运行时间，画出曲线分析。

  - 要求：

- ex2/input/1_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
    - 最长公共子序列个数
    - 最长公共子序列：每个最长公共子序列占一行
  - time.txt：每个规模的运行时间占一行

3. 方法和步骤

exp1

- 方法

  一个非平凡的矩阵链乘问题实例的任何解都需要划分链，而任何最优解都是由子问题实例的最优解构成的，在确定分割点是有如下递归求解式：

  $$m[i,j] = \begin{cases} 0 若 i=j \\ min_{i\le k<j}\{m[i,k]+m[k+1,j]+p_{i-1}p_kp_j\} 若 i<j \end{cases}$$

  再由动态规划来求解。

- 步骤
  - 先通过 `Matrix_Chain_Order()` 函数以动态规划求得最优矩阵链乘方案

```
void Matrix_Chain_Order()
{
    for(int i=1;i<=n;i++)
        m[i][i]=0;
    for(int l=2;l<=n;l++)
    {
        for(int i=1;i<=n-l+1;i++)
        {
            int j=i+l-1;
            m[i][j]=9223372036854775807;
            for(int k=i;k<=j-1;k++)
            {
                long long q=m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j];
                if(q<m[i][j])
                {
                    m[i][j]=q;
                    s[i][j]=k;
                }
            }
        }
    }
}
```

  - 再通过 `Print_Optimal_parent(int i,int j)` 函数递归打印最优矩阵链乘方案

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

exp2

- 方法

  LCS问题具有这样的最优子结构。令$X =< x_1, x_2, \cdots, x_m >$和$Y =< y_1, y_2, \cdots, y_n >$为两个序列，$Z =< z_1, z_2, \cdots, z_k >$为X和Y的任意LCS。

1. 如果$x_m = y_n$，则$z_k = x_m = y_n$且Z_{k-1}是X_{m-1}和Y_{n-1}的一个LCS
2. 如果$x_m \neq y_n$，则$z_k \neq x_m$意味着Z是X_{m-1}和Y_n的一个LCS
3. 如果$x_m \neq y_n$，则$z_k \neq y_n$意味着Z是X_m和Y_{n-1}的一个LCS

   故可以得到递归公式如下：

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

   再由动态规划来求解。

- 步骤
  - 先通过`LCS_Length(int n)`函数以动态规划求得LCS的方案

```
void Print_LCS(int i,int j,int cur_len,int max_len)
{
    if(i==0||j==0)
    {
        res_len=max_len;
        for(int k=1;k<=res_len;k++)
            res[cnt][k]=res_str[k];
        cnt++;
        return;
    }
    if(b[i][j]==1)
    {
        res_str[cur_len]=s1[i];
        Print_LCS(i-1,j-1,cur_len-1,max_len);
        return;
    }
    else if(b[i][j]==2)
    {
        Print_LCS(i-1,j,cur_len,max_len);
        return;
    }
    else if(b[i][j]==3)
    {
        Print_LCS(i,j-1,cur_len,max_len);
        return;
    }
    else if(b[i][j]==5)
    {
        Print_LCS(i,j-1,cur_len,max_len);
        Print_LCS(i-1,j,cur_len,max_len);
        return;
    }
}
```

- 再通过`Print_LCS(int i,int j,int cur_len,int max_len)`函数递归打印LCS

```
int LCS_Length(int n)
{
    for(int i=0;i<=n;i++)
    {
        c[i][0]=0;
        c[0][i]=0;
    }
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
        {
            if(s1[i]==s2[j])
            {
                c[i][j]=c[i-1][j-1]+1;
                b[i][j]=1;
            }
            else if(c[i-1][j]>c[i][j-1])
            {
                c[i][j]=c[i-1][j];
                b[i][j]=2;
            }
            else if(c[i-1][j]<c[i][j-1])
            {
                c[i][j]=c[i][j-1];
                b[i][j]=3;
            }
            else if(c[i-1][j]==c[i][j-1])
            {
                c[i][j]=c[i][j-1];
                b[i][j]=5;
            }
        }
    }
    return c[n][n];
}
```

4. 结果与分析

exp1

1. 最优链乘方案及最少乘法运算次数

```
154865959097238
(A1(((A2A3)A4)A5))
42524697503391
((A1A2)(((((((A3A4)A5)A6)A7)A8)A9)A10))
5400945319618
((((((((((((((A1A2)A3)A4)A5)A6)A7)A8)A9)A10)A11)A12)A13)A14)A15)
319329979644400
((A1(A2(A3(A4(A5(A6(A7(A8(A9(A10(A11(A12(A13(A14A15))))))))))))))((((A16A17)A18)A19)A20))
574911761218280
((A1(A2(A3(A4(A5(A6(A7(A8(A9(A10A11))))))))))(((((((((((((A12A13)A14)A15)A16)A17)A18)A19)A20)A21)A22)A23)A24)A25))
```

2. 运行时间的曲线分析

- 每个规模的运行时间

```
0.6us
2.6us
6.5us
13us
23.2us
```

- 时间复杂度理论分析：

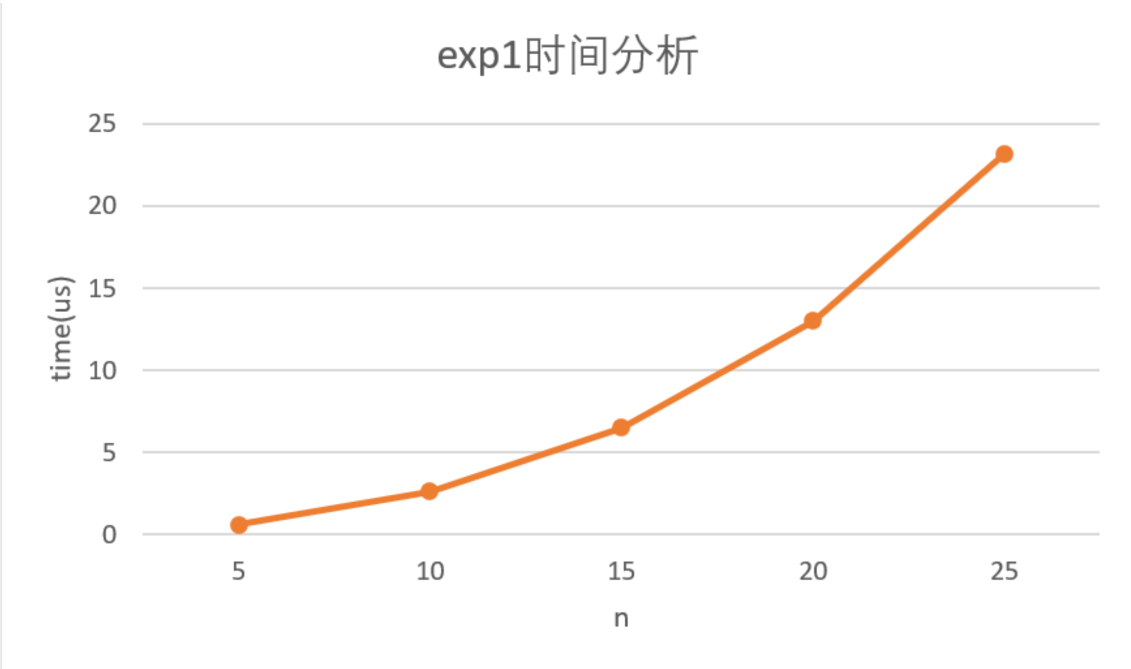
  由于MATRIX-CHAIN-ORDER包含三重for循环，每层的循环变量（l、i和k）最多取n-1个值，可知算法运行时间为$O(n^3)$。

  又最内层循环体内容的运行时间为$\Omega(1)$的，故总运行时间为
  $\sum_{l=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{i+l-2}\Omega(1)=\sum_{l=2}^{n}\sum_{i=1}^{n-l+1}\Omega(l-1)=\sum_{l=2}^{n}\Omega((n-l+1)(l-1))=\Omega(\frac{n^3-n}{6})$
  可知运行时间为$\Omega(n^3)$的。

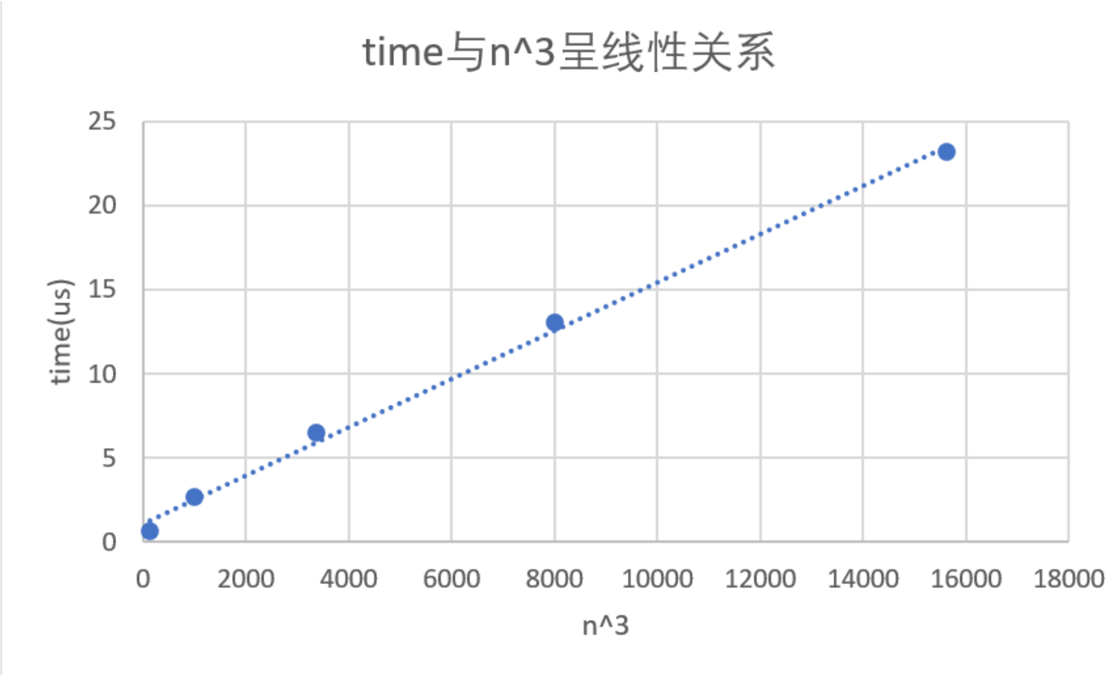
  再者，初始化为单层循环，时间为$\Theta(n)$的。

  故得到`Matrix_Chain_Order()`算法总运行时间为$\Theta(n^3)$的。

- 运行时间的折线图如下：



- 再画出time与n^3的散点图，由趋势线可知time与n^3呈线性关系，即符合$\Theta(n^3)$的时间复杂度理论分析。



3. n=5时的结果截图

```
m:
               5                4                3                2
1 154865959097238 128049683226820 74062781976714 15903764653528 0
2 138766801119366 105723424955724 43981152513978 0
3 183439291324068 119490227350806 0
4 120958281818244 0
5 0
s:
  5 4 3 2
1 1 1 1 1
2 4 3 2
3 4 3
4 4
```

exp2

- 所有最长公共子序列个数和所有最长公共子序列

  以n=10为例，结果如下：

```
20
DABAB
DABAB
CABAB
DABAB
DABAB
CABAB
DABAB
DABAB
CABAB
DABAB
DABAB
CABAB
DABAB
DABAB
CABAB
DABAB
DABAB
CABAB
DAABA
CAABA
```

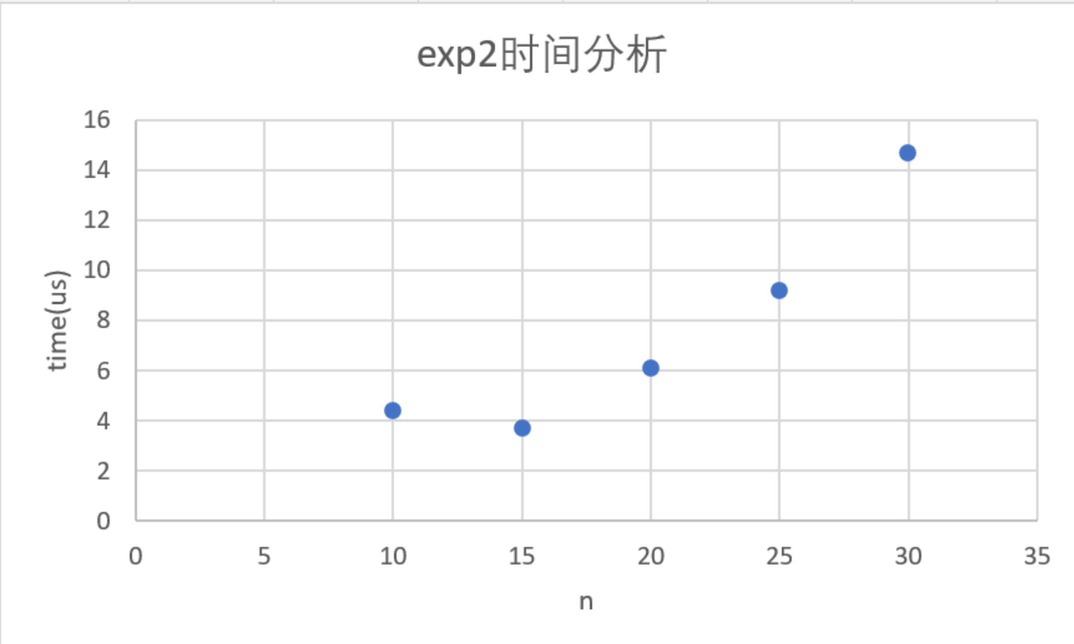
- 时间分析
  - 每个规模的运行时间如下：

```
4.4us
3.7us
6.1us
9.2us
14.7us
```

  - 时间复杂度分析：

    初始化为一个for循环，时间为$\Theta(n+1)$的。主体有两重for循环，最内层循环内容为$\Theta(1)$的，故时间为$\sum_{i=1}^{n}\sum_{j=1}^{n}\Theta(1)=\Theta(n^2)$。故`LCS_Length`的总时间为$\Theta(n^2)$的。

  - 运行时间的曲线分析：



忽略第一个偏离点，时间几乎与n^2呈线性关系，故与理论时间复杂度$\Theta(n^2)$相符合：