# 矩阵链相乘相关实现报告

王资 18214668 计算机技术

## 1.基本思想

1.1 枚举矩阵链相乘的所有方案

利用分治的思想，把矩阵链递归地拆分成两部分，然后把这两部分组合起来。由于原思路为递归思路，在实现字符串组合的问题上会有困难，但这个递归问题很容易转换成递推问题，因为原问题是一个Catalon 数列，具有下述关系： $p(n)=\sum_{i=1}^{n} p(i)p(n-i)$ 也即可以创造一个数组，索引为 i 的元素存放 p(i)的所有可能组合的情况，再递推地完成 i+1 以及之后的情况，其中 p(1)为单个矩阵。

1.2 输出矩阵链相乘计算量最优情况下的连成方案

课件提供的算法实现中，有二维矩阵 s，该矩阵与备忘录矩阵类似，记录了从 i 到 j 的最优子方案，因此，只要递归地找回这些子方案，就能够得到并输出最终解的最优方案。

## 2.代码中的数据结构

2.1 枚举矩阵链相乘的所有方案

用 vector<vector<string>>实现了一个二维的矩阵，根据在基本思想中描述，每个索引 i 存放着 i 个矩阵相乘的所有方案。

2.2 输出矩阵链相乘计算量最优情况下的连成方案

p[] 为矩阵链中的行、列长度链，memo[][]为最优代价备忘录，path[][]为记录最优方案的备忘录。

## 3.程序流程

3.1 枚举矩阵链相乘的所有方案

1.输入相乘的矩阵链的长度 n

2.利用在基本思想中提到的递推关系，得到 p(i), i=1:n 的所有可能方案，最终返回 p(n)，即长度为 n 的矩阵链相乘的所有可能方案。

3.2 输出矩阵链相乘计算量最优情况下的连成方案

1.输入 n+1 个数，代表矩阵链的行、列长度值

2.从备忘录 memo[][]的对角元素初始化为 1 开始，逐渐向右动态地计算最优子方案，同时记录方案的路径，存入 path[][]中

3.根据 path[][]，递归地得到 path[1][n]的方案，即最优方案。

# 4.调试过程出现的问题和解决方法

4.1 枚举矩阵链相乘的所有方案

使用原来的递归思路很难将中间结果存入适合的位置中，查阅资料深入了解矩阵链相乘问题后，得知这个一个 Catalon 数的问题，也得知 Catalon 数的表达式，因此该问题可以简单地转换为递推问题，没有递归调用就可以很轻松地解决这个问题了。

4.2 输出矩阵链相乘计算量最优情况下的连成方案

由于课件提供的算法已经有把最优步骤存入 s[][]（我的实现中使用的变量名为 path）中的方法，因此只要递归地读取 s[1][n]即可解决该问题，因此总体上没有出现什么大 的问题。

# 5.运行结果

5.1 枚举矩阵链相乘的所有方案

程序使用 A 来代表一个矩阵。

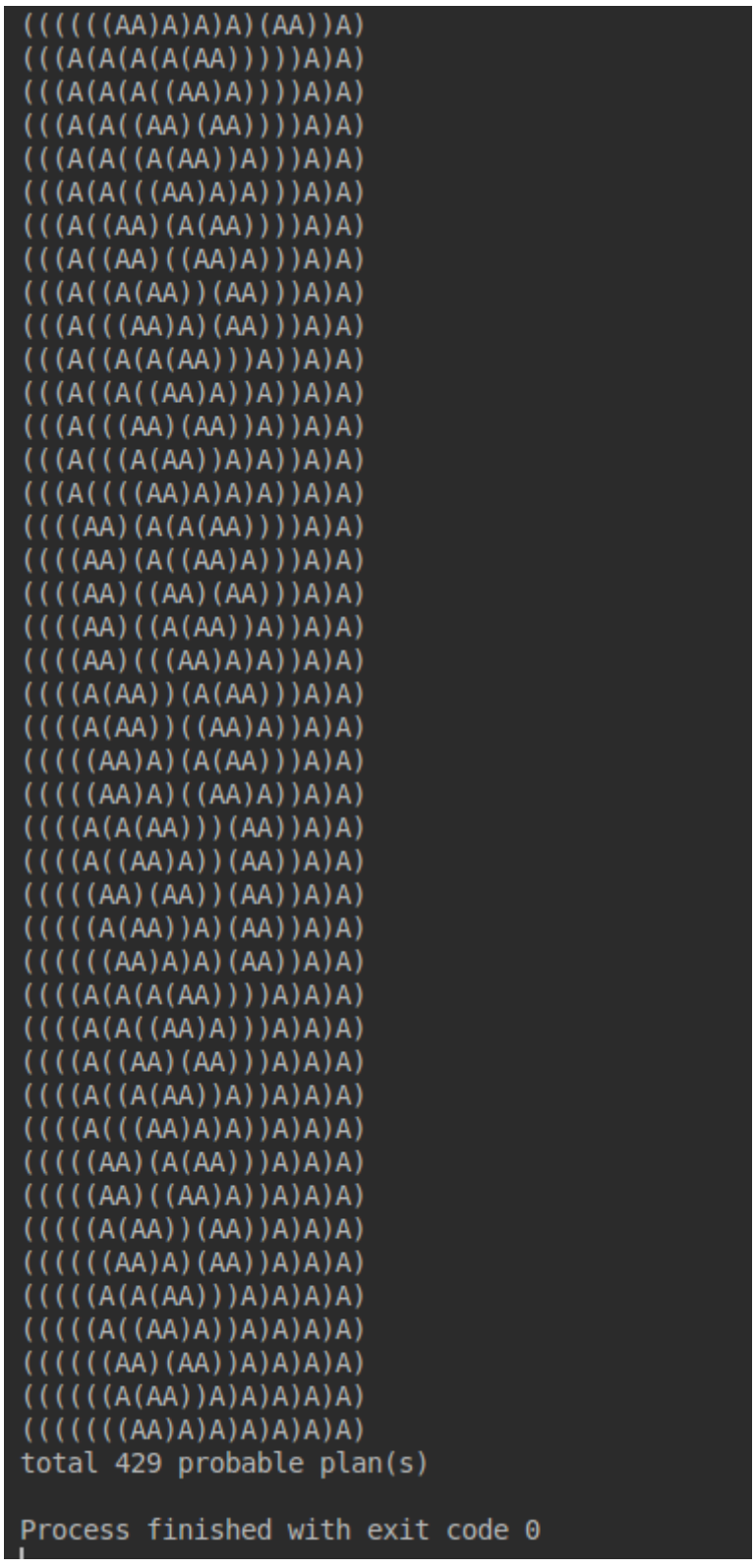
输入为 1

```
/home/wz/CLionProjects/mat_mul/cmake-build-debug/mat_mul
1
A
total 1 probable plan(s)

Process finished with exit code 0
```

输入为 3

```
/home/wz/CLionProjects/mat_mul/cmake-build-debug/mat_mul
3
(A(AA))
((AA)A)
total 2 probable plan(s)

Process finished with exit code 0
```

输入为 8




（没有截取全部输出）

5.2 输出矩阵链相乘计算量最优情况下的连成方案

输入为 50 10 40 30 5 -1（-1 代表输入终止），该输入即课堂演示中用枚举计算所有可能方案的计算值的矩阵行、列长。

```
/home/wz/CLionProjects/matrix_chain/cmake-build-debug/matrix_chain
 50 10 40 30 5 -1
(A(A(AA)))
10500

Process finished with exit code 0
```

该结果与枚举计算的结果相同。

输入为 30 35 15 5 10 20 25 -1（-1 代表输入终止），该输入出自课件中的一个演示。

```
/home/wz/CLionProjects/matrix_chain/cmake-build-debug/matrix_chain
30 35 15 5 10 20 25 -1
((A(AA))((AA)A))
15125

Process finished with exit code 0
```

该结果与课件的演示结果相同。

# 6.总结

本实验使用分治的方法实现枚举矩阵链相乘的所有方案，输出矩阵链相乘的最优方案在提供了动态规划的部分后，实际上也是用分治的方法来输出该方案，因此本次实验的主体部分还是分治法的实现。在实现枚举矩阵链相乘的所有方案过程中，我认识到如果能把递归问题转换为递推问题，那么解决问题的效率，以及程序运行的效率都会大大提升，因此日后在遇到类似的分治问题时，得到基本递归式后，要思考能否可以转化成递推问题，提高实现效率和运行效率。

# 7.存在问题和改进设想

在实现枚举矩阵链相乘的所有方案中，使用一个二维矩阵保存从 1 到 n 的情况下所有可能的方案，这样会导致程序在时间复杂度和空间复杂度都是指数型的，输入值大于 16 时，会耗尽机器内存并宕机，通过学习动态规划的思想，很容易认识到，这样的解决方案存在大量的重复计算和重复存储，因此应当可以使用一个恰当的数据结构，至少使得空间复杂度不是指数型的。