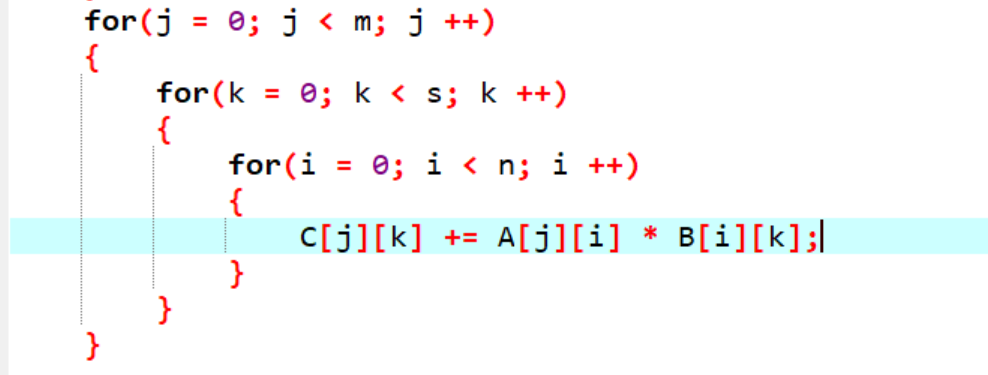
1. **实验目的**
2. 用Strassen计算矩阵乘法。
3. 体会Strassen算法的优越性。
4. 体会算法复杂程度这一概念。
5. **题目**

在屏幕上输入两个矩阵（先输入A的行数列数，再输入A矩阵，B矩阵同理），计算这两个矩阵的乘积并输出。

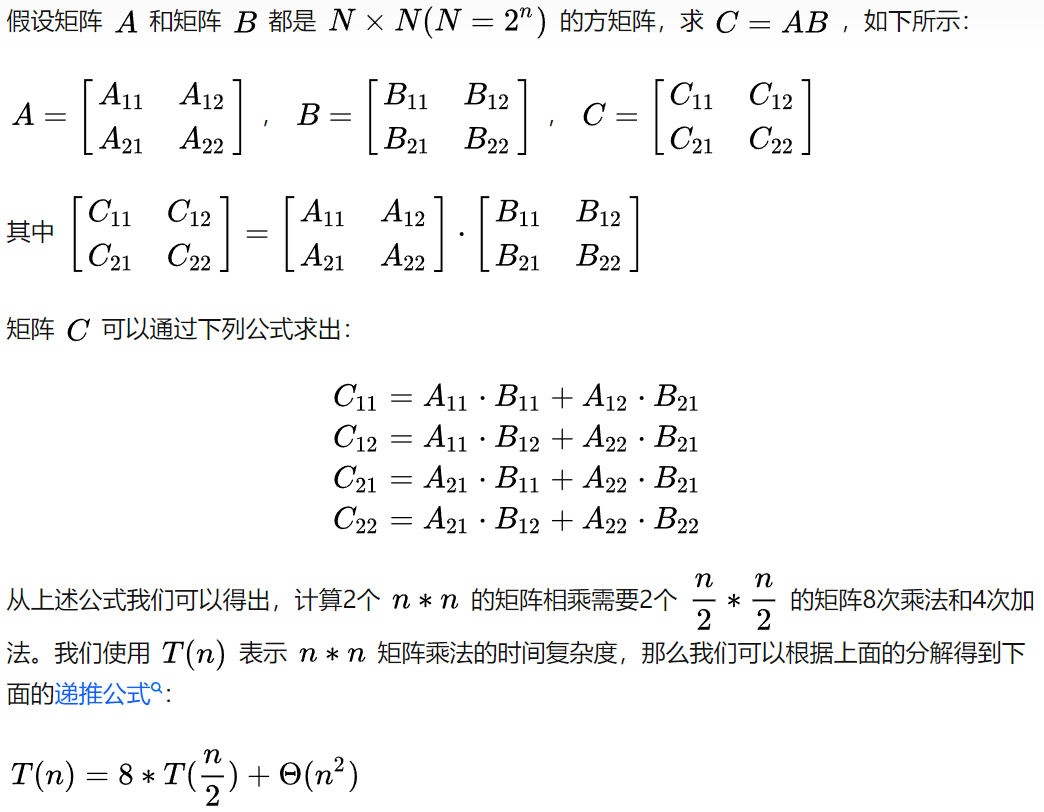
1. **分析**

首先，A矩阵为m \* n大小，B矩阵为r \* s大小，C为乘后的矩阵，大小为m \* s。由矩阵乘法定义可知，n = r，否则无法相乘（让操作者重新输入B矩阵）。

****其次，对于矩阵乘法，很容易想到一种最基本的运算法则，即我们在线性代数中学到的矩阵乘法的定义，而该定义也可以自然而然地使用计算机语言写出，我在这里称作矩阵相乘的朴素算法。朴素矩阵乘法算法的缺陷在于，当A和B的行数和列数很大时，该算法所花费的时间会以很快的速度变长。以下展示了朴素算法的核心代码，运用了三个for循环，显然时间复杂度相对较高，为**Θ(n^3)**。

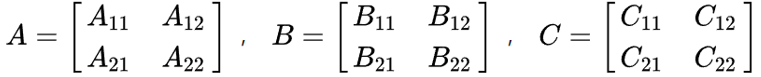
因此，我们考虑到优化矩阵乘法的算法。

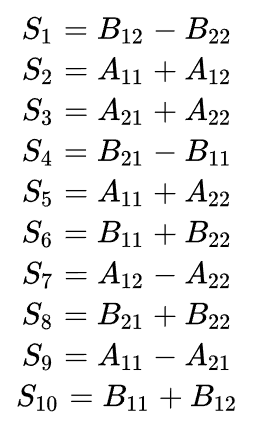
在线代中，矩阵分块可以让矩阵运算在某种情况下减少运算量，而且每一次分块都可以将原矩阵的行数和列数减少，每一次计算的时间减少，矩阵越大，减少的时间越多。所以我们考虑将矩阵分块之后再进行计算，我们称之为分治算法。

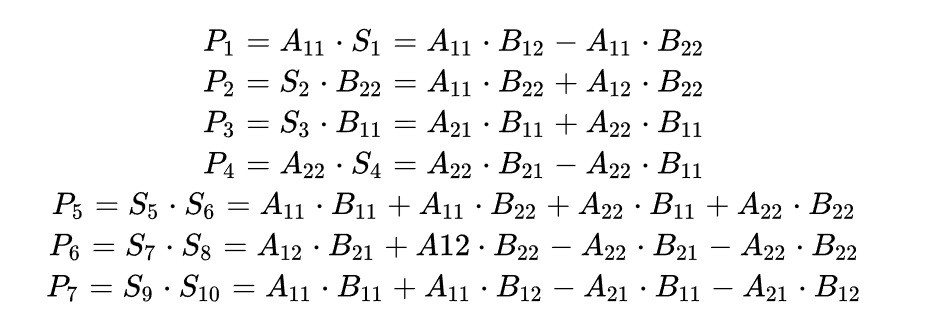
事实上，运用最简单的分治算法，在时间上的复杂度依旧为**T(n)= Θ(n^3)**。

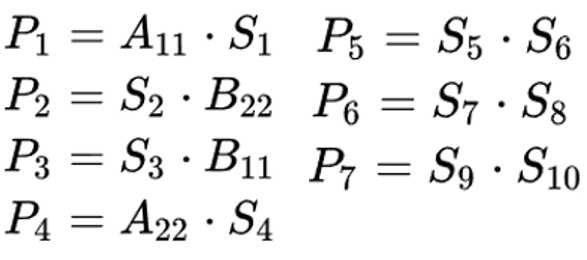
通过以上推导我们发现，每一次矩阵乘法运算的时间复杂度和T(n/2)前的系数密切相关：系数越大，时间复杂度越大。所以我们需要想办法减少系数的大小。

***Strassen提出了这样一种减小系数的方法。***

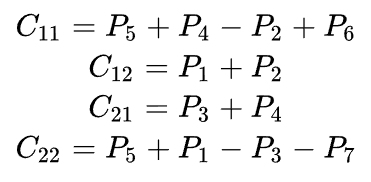
（1）按照以下方式将A,B,C三个矩阵分块。

（2）创建十个大小为（n/2 \* n/2）的方阵。

（3）计算以下七个积。

实际上计算的只是

（4）通过P来表示C。

****这样表示，则将系数由8降到7，相应的，时间复杂度也由**Θ(n^3)**，降到了 。

以上的算法中，朴素算法适合所有的矩阵运算，分治算法和Strassen改进的算法只适用于方阵相乘且方阵为（2^n \* 2^n）大小。

1. **设计思路**

算法1：朴素算法。（核心段落展示）

for(j = 0; j < m; j ++)

{

for(k = 0; k < s; k ++)

{

for(i = 0; i < n; i ++)

{

C[j][k] += A[j][i] \* B[i][k];

}

}

}

算法2：Strassen分治算法。（核心段落展示（递归单元））

//计算M1-M7（只展示M1，其余类似）

//M1[][]

ADD(A11, A22, AResult, HalfSize);

ADD(B11, B22, BResult, HalfSize);

Strassen(AResult, BResult, M1, HalfSize);

1. **程序清单**

[Strassenc.h](../代码/strassenc.h)(将使用的函数全部写入该头文件中)

[Strassenc.cpp](../代码/strassenc.cpp)(主程序)

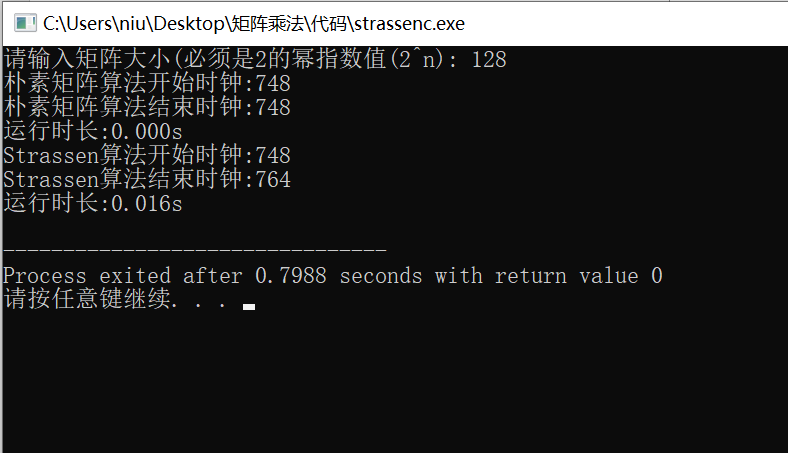
1. **运行结果(为了方便，我们把矩阵的每一个元素设置成一位数)**

多次测试，取平均值填入表格：

|  |  |  |
| --- | --- | --- |
| **大小** | **朴素算法时长** | **Strassen算法时长** |
| 2,4,8,16,32 | 0.000s | 0.000s |
| 64 | 0.001s | 0.001s |
| 128 | 0.007s | 0.005s |
| 256 | 0.046s | 0.038s |
| 512 | 0.405s | 0.274s |
| 1024 | 4.231s | 1.942s |
| 2048 | 55.715s | 13.592s |

1. **探究结果**

我们发现，在方阵的大小小于128时，Strassen算法和朴素算法的计算时长上的差距很小，但当矩阵越来越大，朴素算法和Strassen算法的时长暴增，但显然Strassen算法要优于朴素算法。

但是在测试数据的过程中，我也发现了部分反常于规律的数据。

当矩阵大小是256 \* 256时，虽然平均数据吻合规律，但是某一次的实验数据显示Strassen算法时长比朴素算法长，如果按照正常的分析和解释，这样的状况是不可能出现的，优化后的算法一定比之前的算法要快。

所以我们撇开数学层面，考虑代码层面，我发现，Strassen算法采用了多次递归，动态分配了多次空间， 这样可能将算法的时长拉长。但随着矩阵变大，动态分配的时长显然小于了算法本身复杂度带来的时长。

当然也有计算机时钟带来的误差产生的影响。

1. **收获**

通过本次对Strassen算法优越性的探究，我明确了算法优化的重要性，即使是一小点的优化也会随着数据的不断庞大而愈发显示其优越性。

同时，Strassen算法中采取的递归计算方法，还有动态分配空间的方法都是c语言中很重要的方法，通过本次学习，我加深了对这两种方法的理解，练习了这两种方法。

再者，我在本次大作业中也练习了文件的相关操作，头文件的编写，这些技能也是不可或缺的。

最后，探究的循序渐进也让我理解了探索精神。