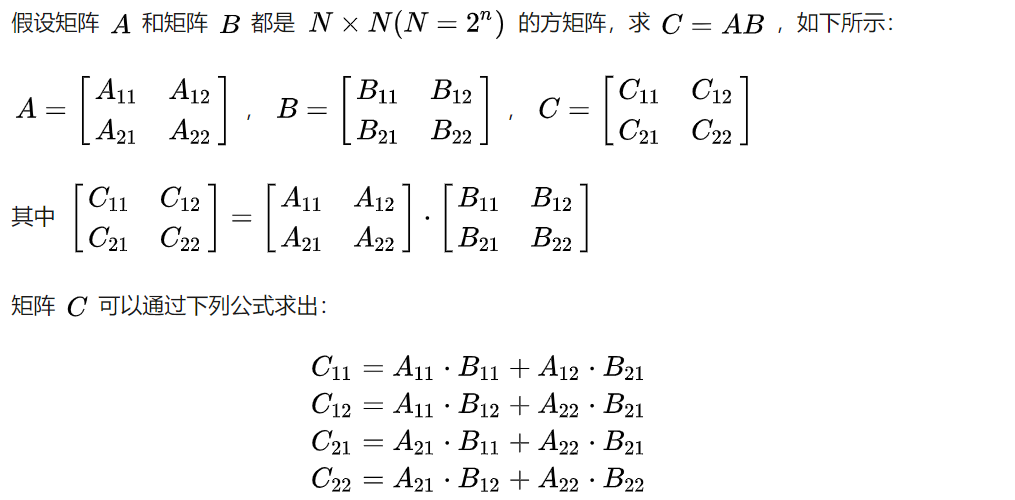
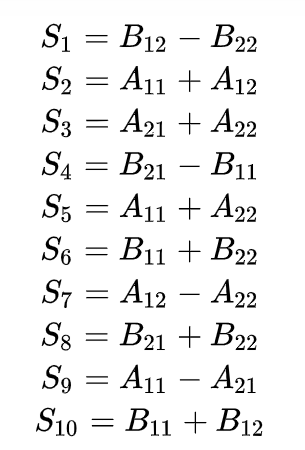
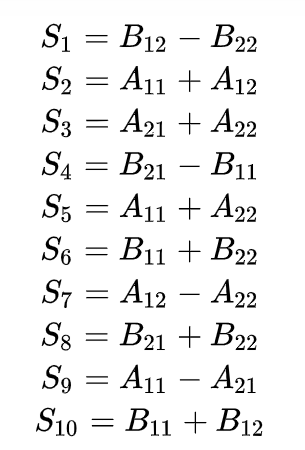
实验报告

一、Strassen矩阵乘法的实现

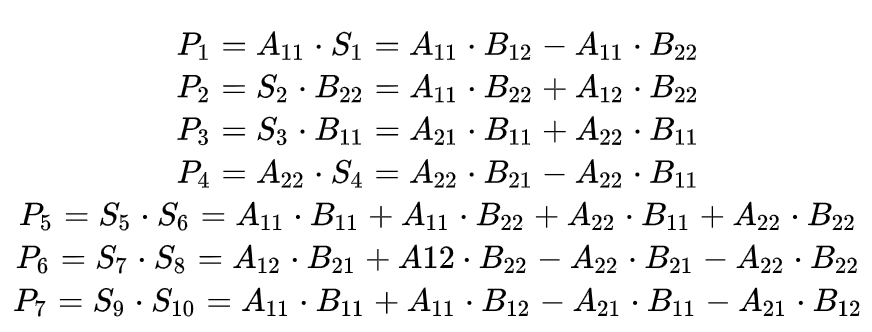
1.原理



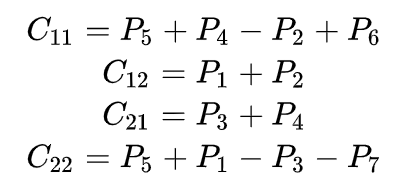
如上分解完矩阵ABC之后，创建10个N/2xN/2的矩阵S1-S10



然后由此计算P1-P7



最后根据Pi计算C11-C22，拼成矩阵C得到结果



2.思路

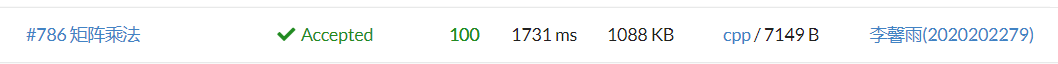
由于Strassen矩阵乘法只适⽤于N x N（N=2n）的⽅矩阵，所以对于⼀般的矩阵乘法需要将两个矩阵补零⾄该情况再计算，最后只输出需要部分即可。

3.代码

见附件“786.cpp”

4.正确性检验

提交YOJ#786，证明结果正确

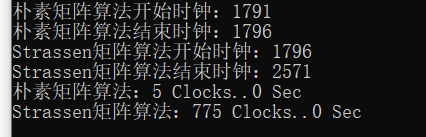


二、Strassen矩阵乘法与朴素算法对比

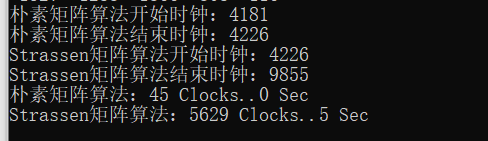
1.结果展示

用所给数据进行试验，得到如下结果：（代码见compare.cpp）

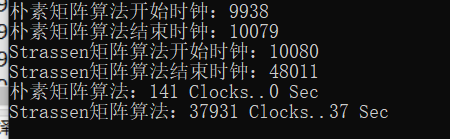
100 100 100



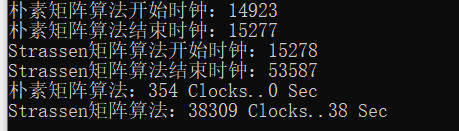
200 200 200



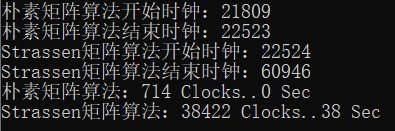
300 300 300



400 400 400



500 500 500



得到如下表：

|  |  |  |
| --- | --- | --- |
| 矩阵大小 | 朴素算法（时钟数） | Strassen算法（时钟数） |
| 100 | 5 | 775 |
| 200 | 45 | 5629 |
| 300 | 141 | 37931 |
| 400 | 354 | 38309 |
| 500 | 714 | 38422 |

2.对比分析

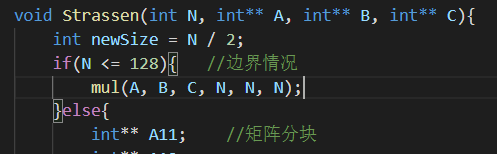
朴素算法的时间复杂度为O(n3)，而Strassen算法的时间复杂度为O(n2.375),本应具有性能优势。但是根据1中的结果，可以发现在所给数据下，相比朴素算法，Strassen算法的耗时不但没有减少，反而剧烈增多，效果更差。

3.原因与改进

通过实验和查阅相关资料，猜测导致Strassen算法性能较差的原因可能是：

采用Strassen算法作递归运算，需要创建大量的动态二维数组，其中分配堆内存空间将占用大量计算时间，从而掩盖了Strassen算法的优势。

针对这一点，对Strassen算法进行改进。为其设置一个界限，当矩阵的维度小于这个界限时，采用朴素算法计算矩阵而不再分治递归，以此节省多次分配空间所花费的时间。通过实验，选择N=128作为界限值，改动如下，改进后计算时间大幅下降。



三、结语

本次实验中，成功实现了矩阵相乘的Strassen算法并通过YOJ验证了其正确性。还通过对比朴素算法与Strassen算法的运行情况，发现了Strassen算法的缺陷并进行了改进，完成了实验要求。