实验报告

矩阵相乘的常规法和Strassen方法比较

实验环境

CPU：Intel Core i5-7200U

内存：8GB

平台：Windows 10 64位教育版

编程语言：Python 3.6

算法分析

常规法

常规法采用三重循环计算矩阵乘法，其时间复杂度$T(n)=\Theta(n^3)$

Strassen方法

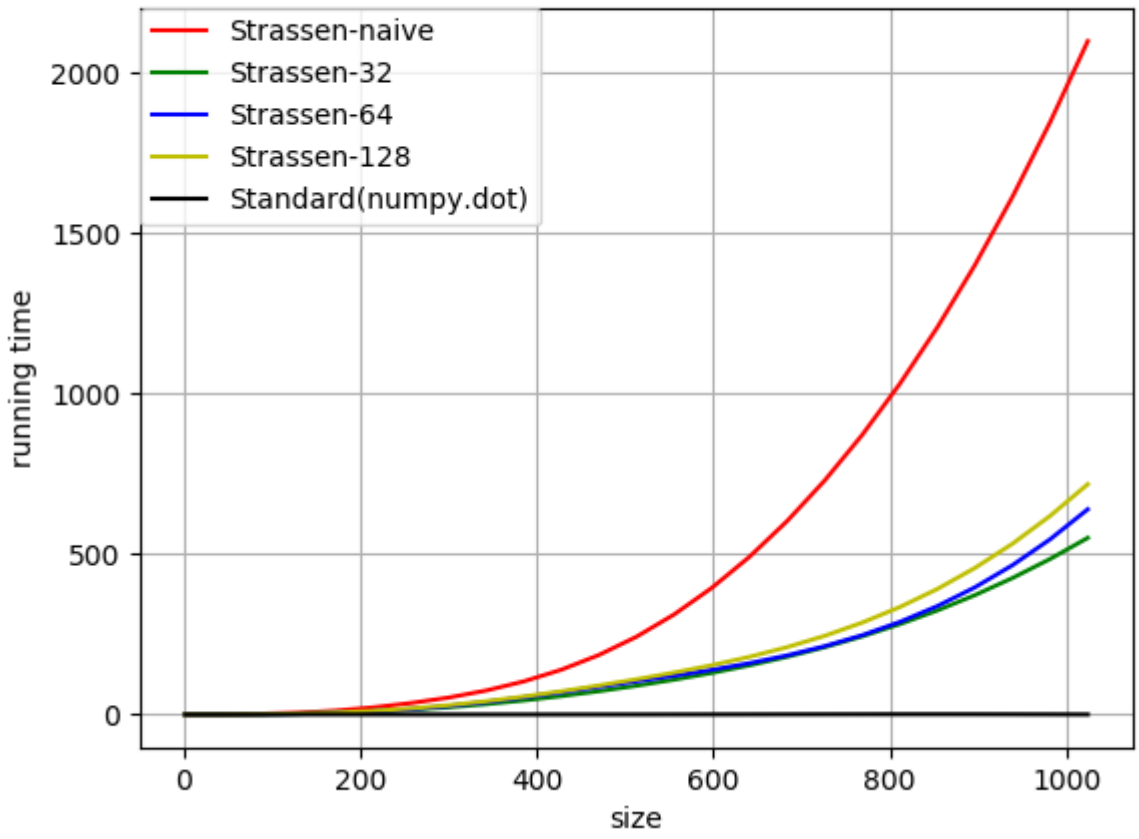
Strassen方法采用分治与递归的思想，将相乘的矩阵均分为4个子阵，利用数学方法将子阵相乘的次数由8次减少为7次，其时间复杂度$T(n)=\Theta(n^{log_2 7})\approx\Theta(n^{2.81})$

结果分析

取矩阵维数$n=2^i(i=1,2,3,\cdots,10)$，使用两种方法进行实验，同时以numpy.dot()函数作为基准时间，每组实验均进行5次取运行时间的平均值，并记录其运行时间如下表（单位为秒）：

$n=2^i$	常规法	Strassen方法	numpy.dot()
1	3.28495e-05	0.00012	6.83421e-05
2	0.00017	0.00039	8.30677e-06
3	0.00060	0.00722	9.43951e-06
4	0.00881	0.01725	1.32153e-05
5	0.06326	0.12123	5.73922e-05
6	0.50343	0.87960	9.17521e-05
7	3.32507	5.04043	0.00034
8	19.8800	34.9376	0.00442
9	160.977	240.878	0.01483
10	1629.66	2098.84	0.06088

在实验中发现，虽然Strassen方法$T(n) \approx \Theta(n^{2.81})$比常规法小，但由于其中不必要的递归过程耗时过多，总体仍然比常规法慢了不少。为了减少递归过程耗时，进一步的实验对Strassen方法进行下界优化，使得递归函数中当$n \leq 32$(或64或128)时终止递归过程，直接调用常规法计算矩阵乘积，进一步的实验结果见下图：



实验结果表明，优化后的Strassen方法运行速度有了很大提升，其中$n \leq 32$优化的效果最好，在size=1024时运行时间仅为优化前的26%，常规法的34%，但相比numpy.dot()函数的基准时间仍然有非常大的差距。猜想numpy.dot()函数采用了多线程并行、调用C等方法才能取得这样的优化结果。