矩阵乘法比较实验报告

洪方舟
Student ID: 2016013259
Email: hongfz16@163.com

March 6, 2018

1. 实验目的

a. 实现两种矩阵相乘的方法（常规法与Strassen方法）
b. 比较两种矩阵相乘方法在不同数据大小（矩阵维数）下所需时间
c. 分析实验结果并给出在实际操作中选择哪种方法建议

2. 实验环境

操作系统：macOS High Sierra (Version 10.13.1)
处理器：1.6GHz Intel Core i5
编程语言：C++
编译器：g++

3. 实验方法

a. 编写两种不同的计算矩阵的函数，将两种方法的计算结果进行比较来保证正确率
b. 随机生成两个维数为$2^n, n = 1, 2, 3, ..., 11$的矩阵，使用上述两种方法进行计算，并计算时间
c. 记录不同维数下两种方法所耗时间，进行比较分析

4. 实验结果

Table 1: Strassen方法实现中，递推的结束点设置为矩阵维数为1×1

矩阵维数	常规法耗时(s)	Strassen方法耗时(s)
2	$7e-06$	$3.5e-05$
4	$2e-06$	0.000178
8	$6e-06$	0.001375
16	$3e-05$	0.008691
32	0.00023	0.058942
64	0.001829	0.281699
128	0.010066	2.17835
256	0.149735	15.0091
512	1.35245	105.976
1024	33.5958	717.901
2048	307.071	$\times$

显然，虽然理论上Strassen方法的时间复杂度$\Theta(n^{2.81})$低于常规方法的$O(n^3)$，但是实际运行结果表明后者效率远高于前者。

Table 2: Strassen方法实现中，递推的结束点设置为矩阵维数为128×128

矩阵维数	常规法耗时(s)	Strassen方法耗时(s)
2	$3.7e-05$	$2e-06$
4	$3e-06$	$3e-06$
8	$8e-06$	$9e-06$
16	$4.6e-05$	$4.6e-05$
32	0.00035	0.000352
64	0.002738	0.002013
128	0.016678	0.016214
256	0.100411	0.078303
512	1.05642	0.57834
1024	10.6463	4.3006
2048	213.335	29.9343
4096	2218.87	216.887

对递归结束点进行调整之后，Strassen方法的运行效率明显提高，理论上的复杂度优势得到了明显的体现。

5. 分析与总结

a. 上述第一个实验中Strassen方法明显慢于常规方法，可能的原因包括：

i.Strassen方法在计算中需要动态分配大量的内存，每递归一层就需要分配$\Theta(n^2)$的内存，大量的时间耗费在分配内存空间上

ii.Strassen方法使用递归的思想，多次的递归调用耗时较多

b. 第二个实验中将Strassen方法递归的结束点设置为128×128之后，减少了递归调用层数，从而大量减少了所需的分配空间的时间，这样做可以减小复杂度中的常数，从而在与常规方法的比较中体现出复杂度的优势，在较大维数下可以达到常规方法十倍的速度。

c. 综上，在实际运用中，对于维数小于256的矩阵应该使用常规方法，而当维数高于256的时候，应当使用Strassen方法，且须将递归的结束条件设置为维数为128×128

6. 源代码及可执行文件说明

源代码存放在$/bin$目录下，可执行文件存放在$/src$目录下，由于在$macOS$环境下进行编译，所以可执行文件只能在$macOS$环境中运行。运行后自动开始测试不同维数的矩阵相乘，在控制台中输出当前维数信息并且输出两种方法所用时间。