CS205 C/C++ Programming Report Of Project 5

Name: 杨博乔
SID: 12112805

Part 1. analysis

因为是基于project3&4的，本次project的其他部分可以看前两次的报告，此外，我进行了以下优化（上次没实现的 1

1. 使用了模板
2. 将大量指针转换为引用，这会优化掉我project3苦心孤诣做的输出双指针模式。（使用引用作为输出变量可以有效处理）
3. 大量使用运算符重载
4. 使用了转置的strassen算法进行矩阵乘，较上次的strassen在常数上优化了（主要是缓存命中率部分）

题目需求：使用c++写一个类，如project4的矩阵，但是需要运算符重载和支持多类型，同时满足于老师的「七条建议」主要上做的是将上一次的代码翻译为cpp并对特殊数据类型进行实例化，但对于矩阵提速我有了更新的看法

strassen优化，考虑分治但实际作用不大，因为有进出栈过程容易常数暴毙（实际上也是，使用了block的方法几乎没有对于上一种的优化，哪怕开了O3）

这个是对于算法复杂度的优化，对于n阶方阵，其时间复杂度为O(pow(n,2.8))，stl库说得好，数据小可以尝试复杂度更小的暴力，因此经过试验，对于128以下的分治出来的矩阵，直接走寻址优化返回，（因为多线程会导致错误结果，写保护又失去了多线程的意义。

Part 2. code

您可以在这里看代码 https://github.com/Cu4water/CS205project5BestMatrixCalculator

Part 3. Difficulties & Solutions

1. 对于用户输入，我们使用了fscanf,但是后来发现如果自己重载了输入流的话，用户自己处理会比我们手动处理快得多，于是我换用了cin/cout进行输入输出
2. 对于一些矩阵，考虑到修改矩阵时可能导致的大量删除重写，请用户自行判断什么时候传值什么时候传址，等号（以及copy constructor和copy()函数）默认传址，而如果使用copy_num的话，可以传值，但是受限于算法的复杂度与数据规模，尽管我尽可能的提升了缓存命中率，效率可能还是会较传址更为低下。
3. 对于SIMD优化，我开始使用的方法是"256/sizeof(T)"，后来发现可能由于用户自定义类型过大导致死循环，因此SIMD优化仅针对int, char, float与double类型
4. 关于ROI:由于题目需求的计算能力只是==, +, -, *一类的简单计算，因此ROI的存在其实是内存空间的冗余
5. 关于omp：由于某些用户自定数据类型可能线程不安全，我只优化了一些基本数据类型。
6. 对于一些安全的操作，我进行了额外的运算符重载，如在上次报告中证明的：转置以优化矩阵乘效率