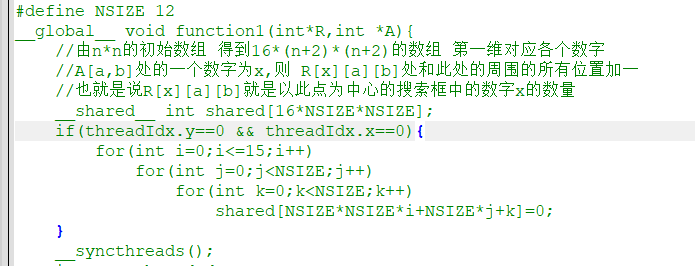
//给的代码的框的大小为3，因为pdf图示为3。而且3最有代表性，//即使变成5,7，也是完全同理

1. 注释解释的很清楚，核函数也分别解释了
2. 

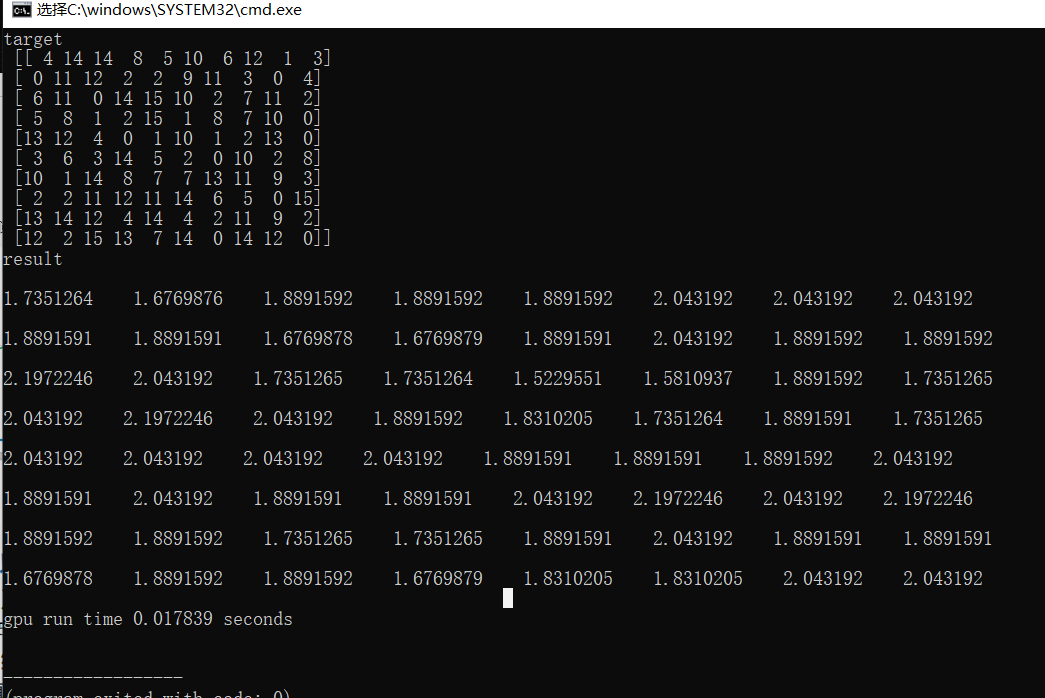
对于，框内某一数字的统计使用共享内存。因为其里面的内容要多次提取和修改。共享内存读取速度快，因为在GPU中保存。

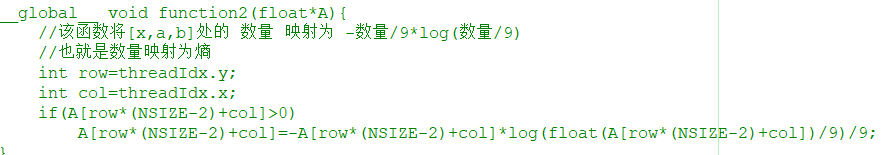


此外\_\_syncthreads();对不同方向的区域的操作进行分段。保证每次不会同时修改同一个位置的内存。

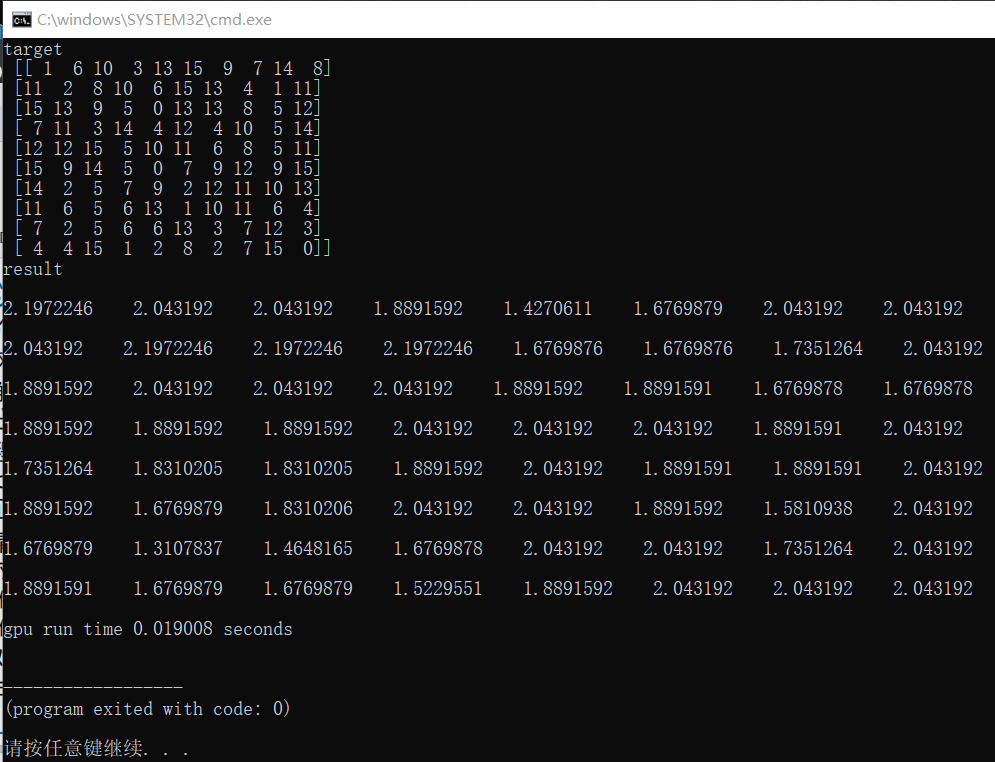
3.

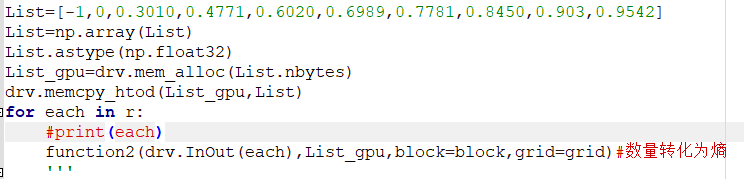
直接用log函数的结果：

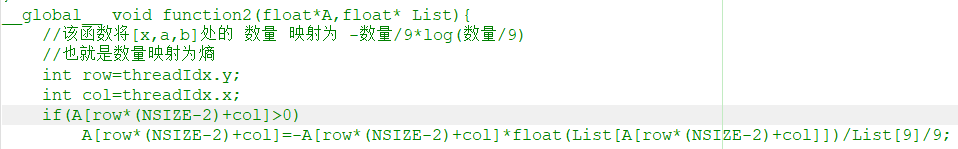




使用对照表的结果：







两者差别不大，说明cuda在运算过程中会保留一些小计算的结果，不会重复计算。

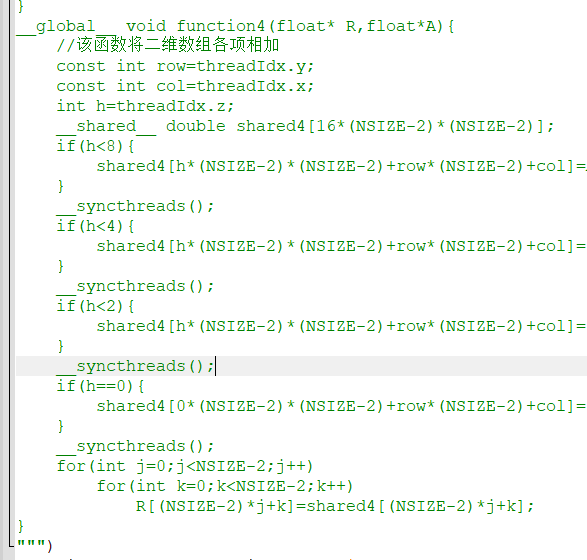
4.basline版本：hw1\_baseline.py

优化版本：hw1\_plus.py

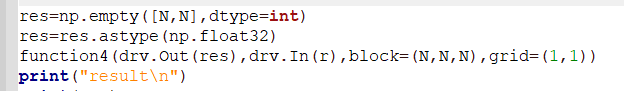
使用更多的共享内存



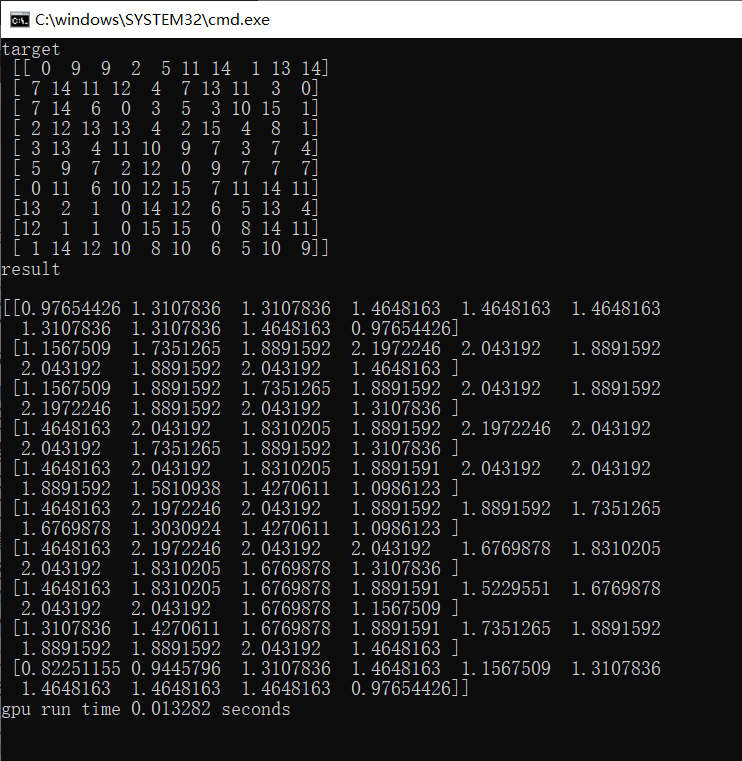
增加函数function4



用一个函数完成各数字对应项的相加。同时相加运算的数量是当前集合大小的一半。



效果是运行时间缩短了



5.影响cuda性能的因素，结合试验分析

在实验中最开始我没有设置同步，其速度很快，但各部分相互影响，导致错误。

说明同步会降低运算的效率，因为一部分运行好的程序需要等待其他程序完成。

并行度越高，运算越快。

在未改良的版本里，每次只运行一个矩阵加法，但改进后的代码，同时运行多组矩阵加法。用的时间也有明显的缩短。说明，程序的并行程度对效率至关重要。拿最极端的情况来说明，所有指令都不并行，就是线性执行，时间最长。所有程序都并行，就是只花一个程序的时间，用时最短。

读取，修改数据所在的内存的位置也会影响运算效率。从实验中我发现，使用共享内存时所用时间较短。这是因为减少了数据读取和储存所用的时间。

一些算式的运算途径，并不影响效率。这可能是因为cuda运算时，会保留中间量的结果，且用于之后的运算。