Project 2\_2

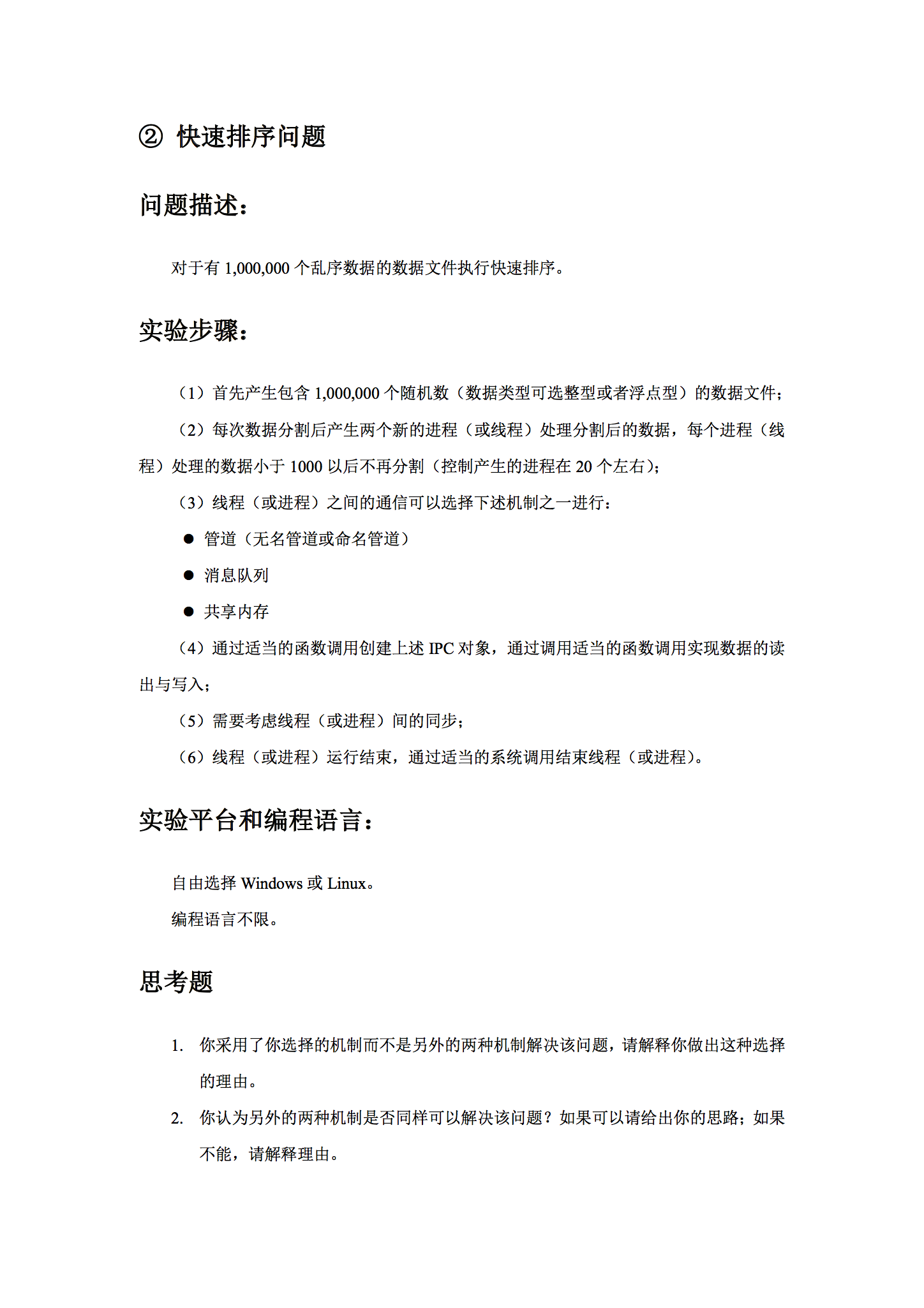
快速排序问题

班级：无53

姓名：陈相宁

学号：2015011033

Email：[c.xiangning1997@gmail.com](mailto:c.xiangning1997@gmail.com)



1. **实验思路：**

采用多线程处理本实验。

在快速排序问题中，采用分治思想：首先找到其中一个基数的正确位置，从而将序列分割为比基数小的一段以及比基数大的一段，而这两段序列的排序相互独立，因此可以交给两个线程分别完成。

许多多线程编程方案中，线程会即时创建，即时销毁，尽管与创建进程相比，创建线程的时间已经大大的缩短，但是如果提交给线程的任务是执行时间较短，而且执行次数极其频繁，那么服务器将处于不停的创建线程，销毁线程的状态。本实验中，限定线程数量为20个左右，并且很多线程完成的任务只是将序列一份为2，执行时间较短。因此，我就自然地想到了利用线程池来解决本问题。

线程池采取这样的思路：确定数量的线程被预先创立并置于线程池中，另外有一任务队列处理任务的生成及结束，线程处理完任务后并不被销毁而是继续从任务队列中领任务，这样便能够避免多次创立线程，从而节省线程创建和销毁的开销，能带来更好的性能和稳定性。

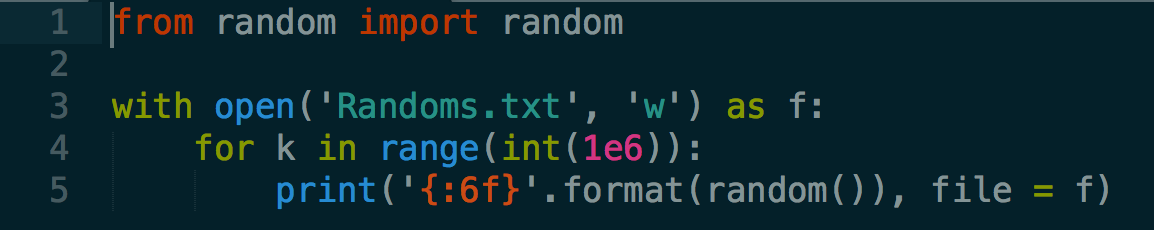
排序问题本身没有特别大的难度，不过由于涉及到共享空间，还是需要利用互斥控制同一时刻只有一个线程读写序列。

1. **代码实现：**

编程语言为python，采取queue库中的Queue类实现线程池中的任务队列，采取threading中的Thread类实现线程，Lock类实现互斥访问，Semaphore类实现信号量。

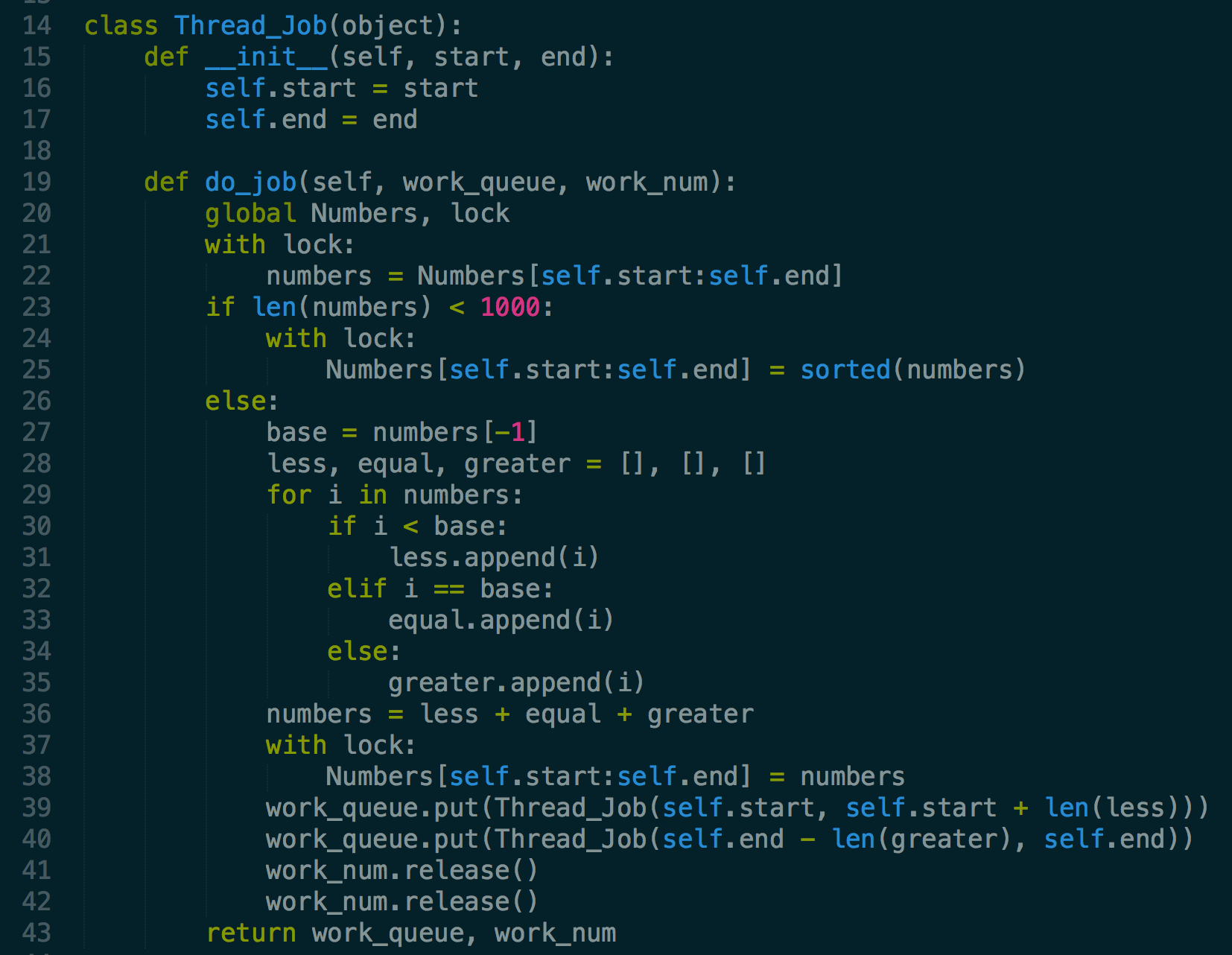
1. Generate\_Randoms.py生成随机数：

比较trivial。



1. Thread\_Job类：

自建Thread\_Job类抽象任务。初始化时输入该任务针对的子序列的起止点。do\_job方法中，首先，利用with lock语句实现互斥读取子序列。之后，如果子序列长度小于1000，则不再分割，直接排序并互斥写回原序列。否则，将子序列中最后1个数作为基数，找出小于基数的所有数作为less，大于基数的所有数作为greater，等于基数的所有数作为equal。将原序列对应位置更改为less + equal + greater，同时在任务队列中添加两个新的排序任务并对表示任务量的信号量执行2次V操作。



1. Thread\_Pool\_Manager类：

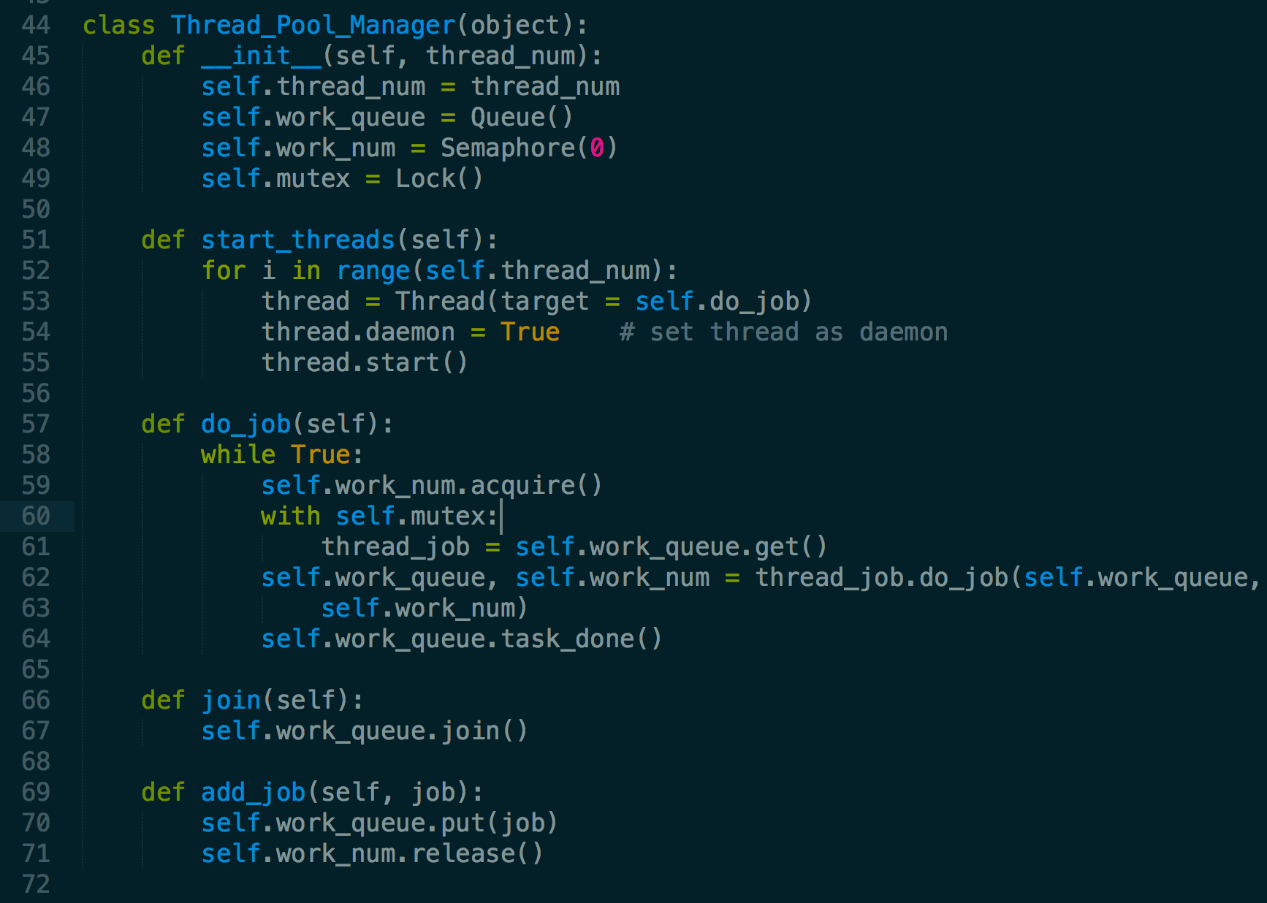
自建Thread\_Pool\_Manager类抽象线程池。初始化时生成空任务队列，初始值为0的表示job数量的信号量work\_num用于防止从空队列中提取任务，以及名为mutex的锁保证同一时间只有一个线程从work\_queue中提取任务。其中有4个方法：

① start\_threads用于启动线程，启动既定数量的线程，执行函数为do\_job。其中关键的一步是将线程池中的线程设置为守护线程daemon，这样所有任务都完成后，所有线程自动退出。

② do\_job中设置一个无限循环，首先对work\_num执行P操作，解除阻塞后在锁变量mutex的保护下从任务work\_queue中提取一个任务，完成任务同时更新work\_queue以及work\_num。之后关键的一步是调用work\_queue的task\_done函数告诉队列这一任务已经完成，这一步是之后的join函数能够正常退出的保证。

③ join函数即执行work\_queue的join操作，必须等到所有任务都完成后程序才继续执行。

④ add\_job函数用于将第一个任务加入work\_queue并对work\_num执行V操作。



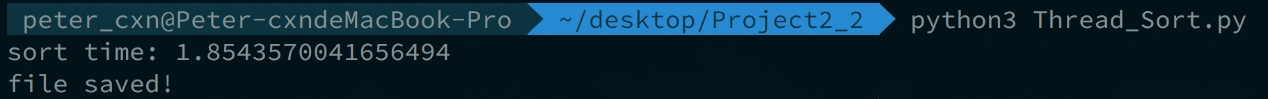
全部代码：

Thread\_Sort.py：



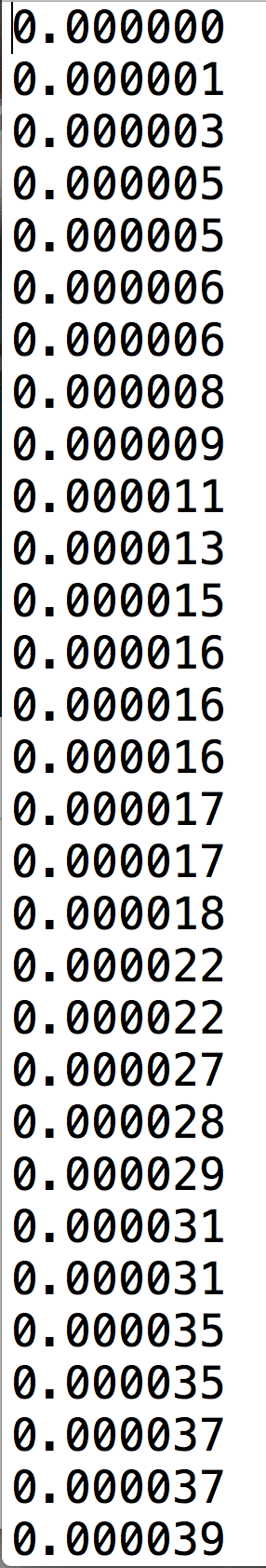
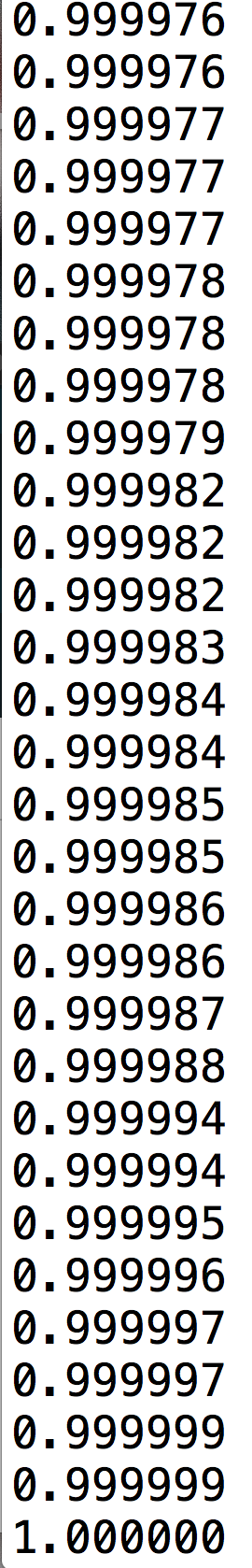


实验结果：



可见，线程池中放置20个线程，则1e6个float型数据排序需要1.85秒，排序好的数据输入到Sorted\_Numbers.txt中。从速度来讲，线程并行排序的速度高于c语言标准库中的函数。

前30个数： 后30个数：



通过循环逐一检测以及抽样的方法确认所有数据均正确排序。

1. **思考题：**
2. **你采用了你选择的机制而不是另外的两种机制解决问题，请解释你做出这种选择的理由。**

我采取的是“共享内存”的方式。因为采用多线程，内存本身就是共享的，最为直观，且不会带来额外的开销。我在本实验中设置了2个全局变量，就是共享内存最为直接的体现。如果使用管道或者消息队列的方式都涉及消息的复制以及传播，都会带来时间及空间上的损耗。

1. **你认为另外的两种机制是否同样可以解决该问题？如果可以请给出你的思路；如果不能，请解释理由。**

同样可以解决。

① 管道：采用多进程，当父进程开启两个子进程时，通过管道将数据传播至子进程即可，当所有进程完成时，排序任务也就自然完成了。

② 消息队列：将需要完成的任务塞入消息队列中，由线程或者进程接收消息完成排序任务即可。