并行编程

1. 在原程序中，子线程创建后进入了一秒的sleep，但是主线程不会等待子线程退出后再执行，而是会直接执行之后的return，主线程退出后主进程也退出，子线程也会相应退出，因此不会执行打印。

   而取消注释后，主线程会被阻塞在join函数，等待子线程退出，因此可以正常打印。

2. 可能会。

   case: 当线程 1 执行完a.lock() 但未执行 b.lock() , 线程 2 执行完 b.lock() 但未执行 a.lock()时， 两个线程会互锁，因此陷入死锁。

3. 思路：

   由于fn的执行顺序不会影响结果，所以我们可以将循环分成若干子循环，并将每个子循环分发给一个线程处理。 具体而言，我们设置线程数为 n，然后统计从 first 到 last 迭代器中总共有多少个，计数为 m，为每个 thread 设置步长为 $step = (m + n - 1)/n$ ，并分别执行 for_each，即可得到 for_each 的多线程版本。

4.

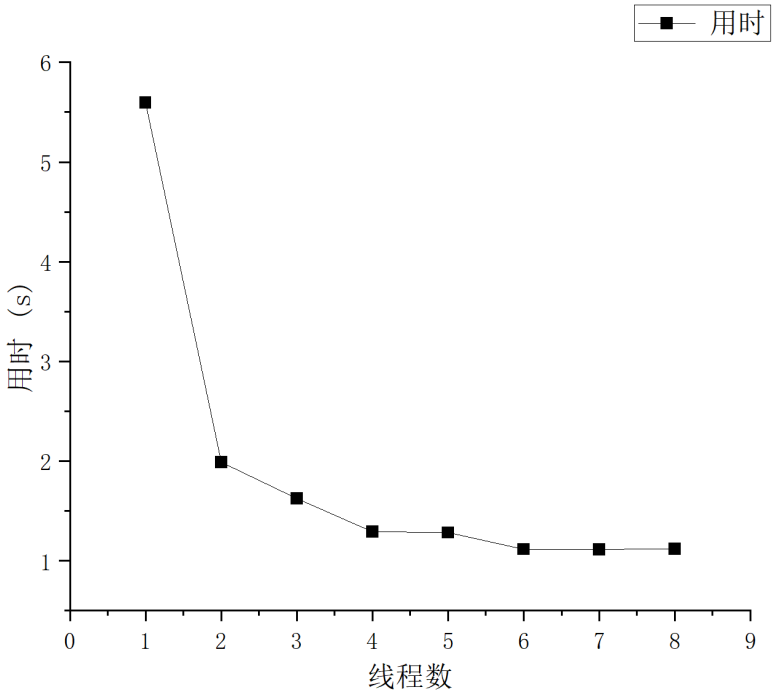
思路：

对于并行化归并排序算法，我们先对排序区间平均分成 n（线程数） 份，然后把每个区间交给一个子线程处理。当所有结果返回后，我们再对每个区间进行合并。

在不同的线程数量下，利用并行归并排序算法对N = 30,000,000数进行排序的程序执行时间如下表所示。

线程数	1	2	3	4	5	6	7	8
用时（s）	5.596	1.989	1.625	1.292	1.258	1.166	1.115	1.119

绘图得：



可见随着线程数的增加，归并排序的用时逐渐减少，且随着线程数接近8，用时逐渐稳定。