學號：**R08946006** 系級：**資料科學學程碩一** 姓名：**周逸平**

1. Comment out the printf statements. Run it with 1, 2, 4, 8,… threads and report the execution time

Threads	Time	Threads	Time	Threads	Time
1	0.000178	64	0.004282	4096	0.185498
2	0.000282	128	0.008258	8192	0.334155
4	0.000347	256	0.016194	16384	0.675680
8	0.000485	512	0.026895	32768	Failed
16	0.004535	1024	0.049806	65536	Failed
32	0.002089	2048	0.110409	131072	Failed

可以看到除了 thread 16 耗時異常高以外，其餘時間幾乎都按照著前一個的兩倍左右增長，這表明了不是 thread 開越多耗時就越低，相反的在矩陣不大的時候，由於開的 thread 過多，反而會使得耗時增加。

2. Double the values of NRA, NCA, and NCB to observe the execution time.

3. Repeat Step 2 until a problem happens to the system. Report your observations.

2 跟 3 整理如下表格:

Threads \ Multiple	2	4	8	16	26(max)
1	0.001075	0.007466	0.033478	0.175396	0.795551
2	0.000298	0.004078	0.020975	0.110924	0.424379
4	0.000314	0.002213	0.013934	0.059409	0.231431
8	0.000729	0.001768	0.008431	0.035248	0.124674
16	0.002855	0.004931	0.009275	0.028420	0.108240
26 (max multiple)	0.001961	0.002633	0.008456	0.029601	0.106922
32	0.002331	0.003003	0.008855	0.030768	0.106047
64	0.004127	0.005101	0.010197	0.032229	0.106629

可以看到矩陣處理所需最短時數通常會落在 threads = multiple 附近，相反的離最短時數越遠的的選項(thread 過多或過少)就相對地耗時更久，故針對檔案大小配適適當的 thread 數量也是很重要的。

奇怪的是，低 multiple 實驗在 thread 16 的時候所需時間通常不會如預期成長，而是突然暴增，所需時間甚至比 thread = 32 還多，但我無法找到原因為何。