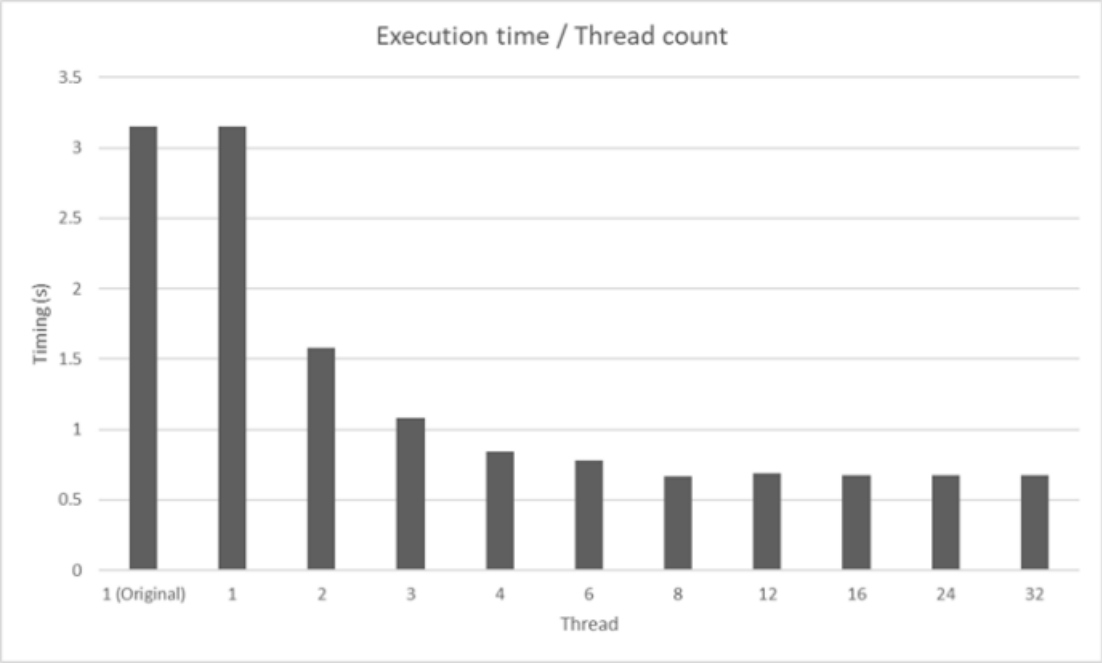
Assignment 3 Report

Yunfan Yang 30067857

Q2 - Written answer

A. Make a table with these timings, and a bar graph

Threads	Timings (s)
1 (Original)	3.153
1	3.153
2	1.581
3	1.083
4	0.841
6	0.778
8	0.664
12	0.691
16	0.676
24	0.676
32	0.674



B. When you run your implementation with N threads, you should see N-times speed up compared to the original single threaded program. Do you observe this in your timings for all values of N?

No. It starts with 2x and then slowly become lower and lower.

C. Why do you stop seeing the speed up after some value of N?

Swiching threads takes time for CPU, as there are more threads, each threads will get less workload, when the workload is too small, it takes more time because CPU is frequently switching threads, and it basically cancels out the benefit by having multiple threads.

Q4 - Written question

medium.txt

# Threads	Time Observed	Speed Up
Original	45.8739s	1.0
1	45.5852s	1.0006
2	23.7622s	1.9305
3	15.9656s	2.7688
4	12.4775s	3.6765
8	12.3818s	3.7049
16	12.0849s	3.7959

hard.txt

# Threads	Time Observed	Speed Up
Original	16.8609s	1.0
1	15.5224s	1.0862
2	8.0834s	2.0858
3	5.4188s	3.1115
4	4.1351s	4.0775
8	4.1271s	4.0854
16	4.1228s	4.0896

hard2.txt

# Threads	Time Observed	Speed Up
Original	15.5519s	1.0
1	15.5137s	1.0024
2	8.0834s	1.9239
3	5.4106s	2.8743
4	4.0844s	3.8076
8	4.0623s	3.8283
16	3.9302s	3.9570

Are the timings what you expected them to be? If not, explain why they differ.

The timing is different from expected, the speeding-up gets levelling off. As mentioned before, there are more threads than actually needed, each threads get too few workloads and CPU is taking time frequently switching between them. There is a balance between thread amount and workload, only falling into the balance will get the optimized time. According to the table, this balance is around 4-threads.