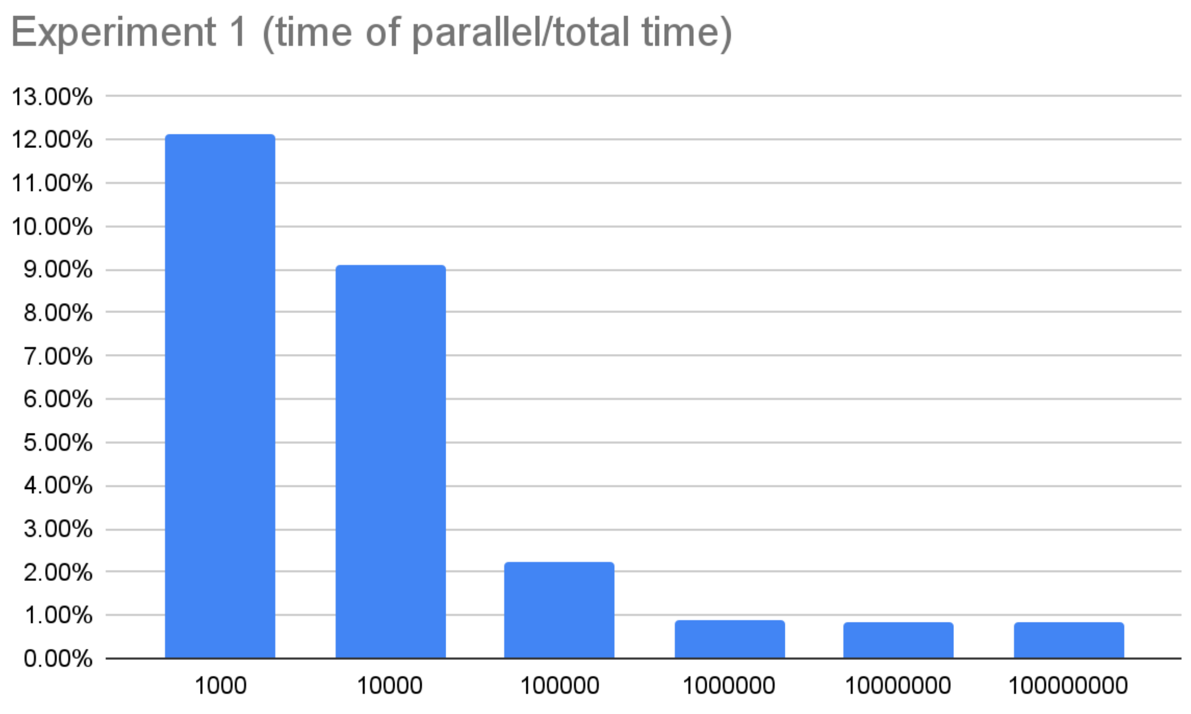
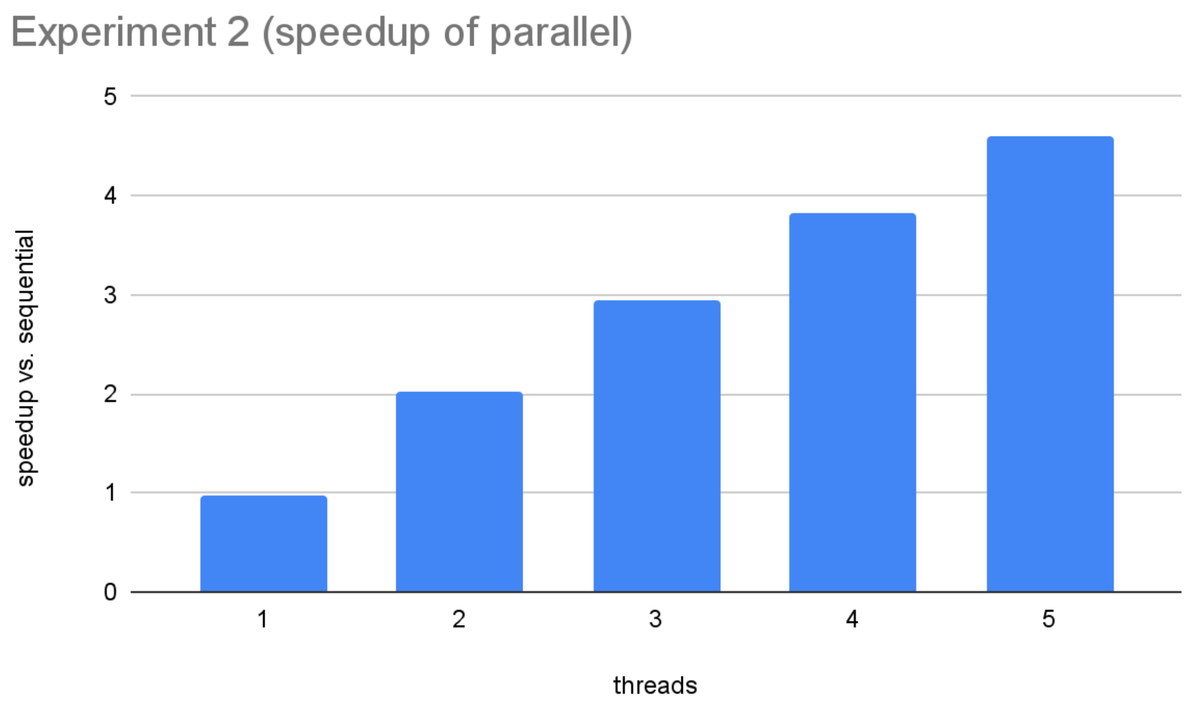
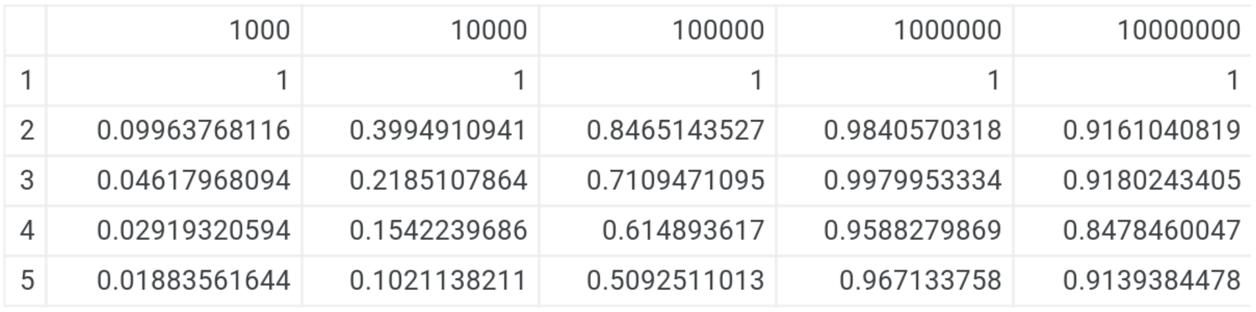
Graphs
(20 bins, 10 threads)



(50 bins, 1 - 5 threads, 1000 000 floats)



Experiment 3 (Efficiency - speedup/# threads): (20 bins, 1 - 5 threads)



	1000	10000	100000	1000000	10000000
1	1	1	1	1	1
2	0.09963768116	0.3994910941	0.8465143527	0.9840570318	0.9161040819
3	0.04617968094	0.2185107864	0.7109471095	0.9979953334	0.9180243405
4	0.02919320594	0.1542239686	0.614893617	0.9588279869	0.8478460047
5	0.01883561644	0.1021138211	0.5092511013	0.967133758	0.9139384478

Analysis
As the problem size increases, the fraction of the parallel part decreases substantially, as the majority of the time is taken up by I/O from reading the large float files, which was done sequentially in the program. Unfortunately, any attempts at parallelizing reading the file with computations were unsuccessful, resulting in slightly more time taken than even sequentially reading the file and then doing parallel computation, because of the usage of expensive system calls. Additionally, calculations take much less time than I/O, so parallelization of the two at these problem sizes may be unproductive.

The speedup as the number of threads increases for 1 million floats is approximately linearly increasing, though it drops most significantly for 5 threads. As the number of threads increases, the speedup falls off slightly due to the overhead of thread creation and synchronization points within the parallel section.