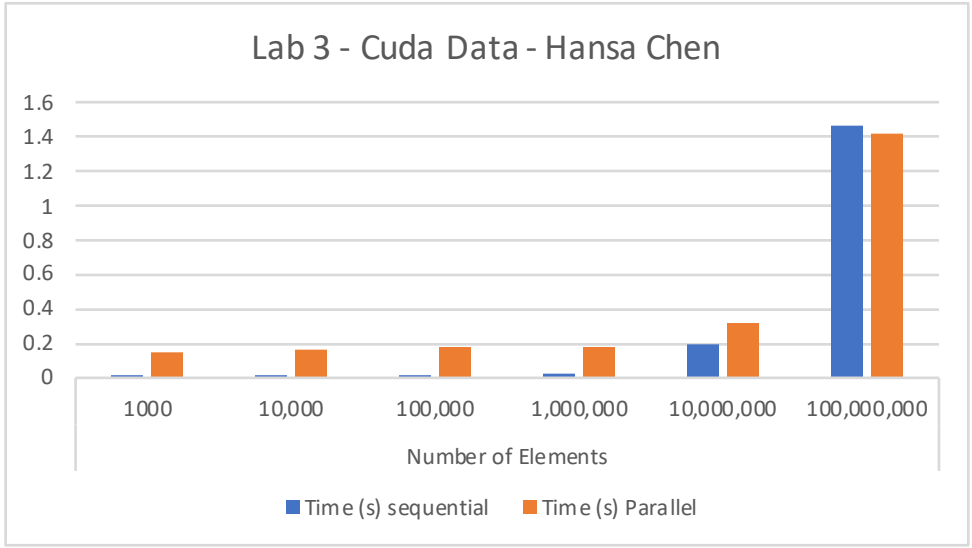
Lab 3 - Cuda Data

Hansa Chen (WC1369)

I used Cuda1

		Time (s)	
		sequential	Parallel
Number of Elements	1000	0.002	0.149
	10,000	0.002	0.158
	100,000	0.006	0.172
	1,000,000	0.032	0.172
	10,000,000	0.199	0.322
	100,000,000	1.457	1.425



I used cudaGetDeviceProperties .maxThreadsPerBlock to get the max threads per block to maximize threads per block usage. Once I have gotten the max threads, I can calculate how many blocks are needed by size / max threads. To compile, I ran the command: nvcc -arch=sm_30 maxgpu.cu -o maxgpu

From the graph, it is not hard to see prior to 10 million elements, the sequential version is much faster than my parallel version. Because of overhead cost to divide elments into block, warps, and threads, afterwards have to gather all elements and find the biggest number, it is not very efficient to run parallel version with smaller numbers. As more and more elements need to be calculated, the parallel version becomes more efficent as more elements can be computed in parallel in different blocks. Hence at 100 millions elements, we see the parallel version is slightly faster than the sequential version.