Sawyer Cowan
CSCE 435

Homework 6 - GPU Programming

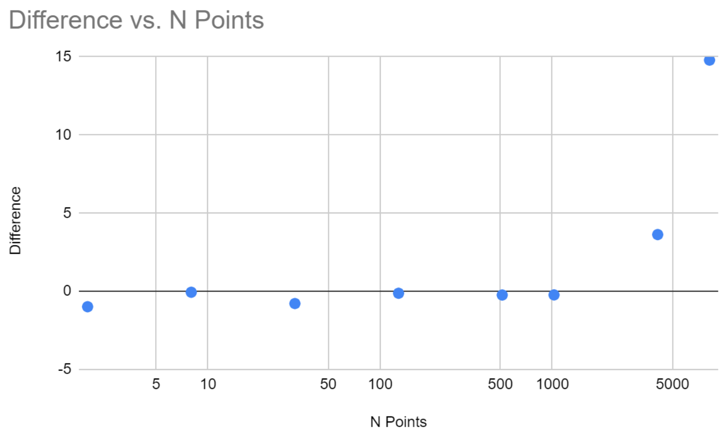
The Approach

First, I copied the minimum_distance_host function for the device version, but modified it for GPU utilization. Then, I made sure that each thread would be used per pair of points, which then required more blocks of threads if the number of threads needed was larger. However, one issue I had to resolve was finding the global min across all the threads. One way I found this is to use an atomic function that would save the distance value to the device memory. Credit for this atomic function (global_dev_min) goes to an old Stack Exchange forum that I found.
nvidia - How do I use atomicMax on floating-point values in CUDA? - Stack Overflow
One more addition to the code was line 199-200 and this was for the device minimum variable.

Data

N Points	CPU time (ms)	GPU time (ms)	Difference	Weighted Favor
2	0.000312	0.98624	-0.985928	-0.999683647
8	0.000632	0.060832	-0.0602	-0.9896107312
32	0.001685	0.78528	-0.783595	-0.9978542685
128	0.009817	0.134048	-0.124231	-0.9267650394
512	0.112691	0.349888	-0.237197	-0.6779226495
1024	0.433486	0.663776	-0.23029	-0.3469393289
4096	6.632746	3.002912	3.629834	1.208771353
8192	26.41729	11.62532	14.79197	1.272392502

If the weighted favor is negative, then the CPU was favored, and positive indicates the GPU was favored. Favor is the GPU time - CPU time and then divided by the GPU time.



(*The CPU could not keep up*)