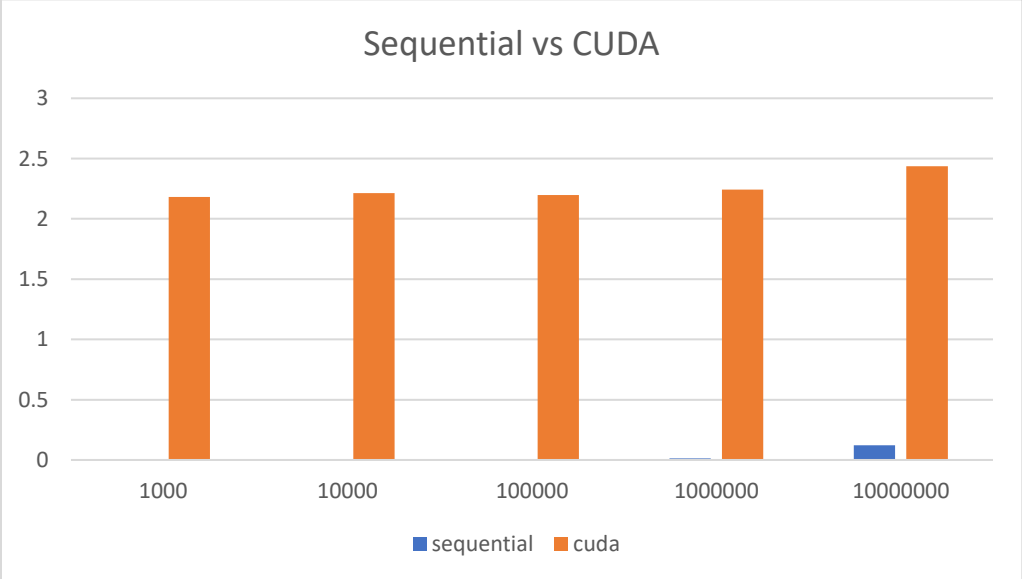
Cuda3

1. The grid size/dimension is always 1 because the nature of the question, which is to search the max number in a 1D array, is 1D, and it is easier to write code that reflects that instead of forcing the program to behave as if there is more than one dimension.
   The thread number is the smaller of the two, the maximal number of threads the GPU can handle per block, and remaining elements to be scanned. Since the maximum thread count per block here happens to be the multiple of 32, which is the warp size, threads are fully utilized until the last iteration, because paddings are not added.
2. nvcc maxgpu.cu
3. (graph and data below)
   The program divides the numbers array into multiple, smaller arrays and assigns them to the GPU to be searched in place. The search is done in a tree like manner, which takes O(log n) time, where n is the smaller array's length. Then, the local maximum is placed on the left and then compared with the original array's first element, where the largest integer will ultimately reside. If the newly found maxima is bigger, it is placed in the first index. Because of the design choice, the growth of search time is slow, but the graph shows that the CUDA version takes a much longer time. The most likely reason, if not the quality of code written by an undergrad, is the overhead from multiple kernel calls. They contribute the most, yet consistent, overhead to the overall time. If the array length is n, there will be ceil(n/maxThreadPerBlock)*2 kernel calls. Also, the assignment is not GPU friendly because there are not many computations, if there are any at all. The core operation here is comparison, which is better done on the CPU, which is where the sequential version is run. The locations and difference of latencies play a major role in the difference in performance.



(The bars for sequential time are not obvious because CUDA takes significantly more time and outshines the sequential version)

			sequential time		
trial 1	0.001	0.003	0.003	0.014	0.121
trial 2	0.002	0.002	0.002	0.014	0.122
trial 3	0.001	0.002	0.002	0.015	0.122
trial 4	0.001	0.004	0.002	0.015	0.121
trial 5	0.002	0.002	0.002	0.014	0.122
Average	0.0014	0.0026	0.0022	0.0144	0.1216
length	1000	10000	100000	1000000	10000000

			cuda time		
trial 1	2.208	2.241	2.188	2.174	2.381
trial 2	2.179	2.248	2.15	2.253	2.455
trial 3	2.165	2.162	2.171	2.265	2.36
trial 4	2.156	2.225	2.317	2.206	2.514
trial 5	2.204	2.189	2.159	2.312	2.476
Average	2.1824	2.213	2.197	2.242	2.4372
length	1000	10000	100000	1000000	10000000

aggregate time (Seq vs CUDA)

1000	0.0014	2.1824
10000	0.0026	2.213
100000	0.0022	2.197
1000000	0.0144	2.242
10000000	0.1216	2.4372