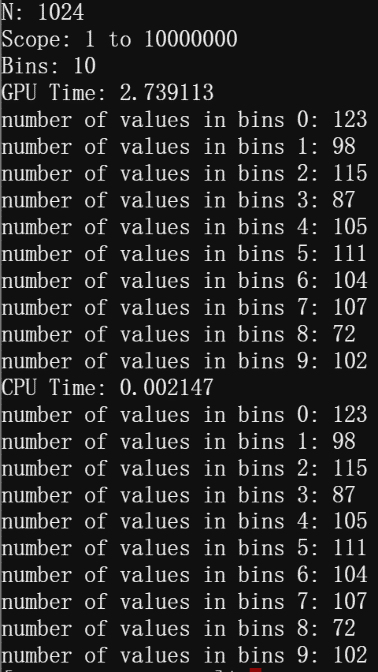
1. Let us revisit the histogramming problem assigned in Homework 4. Your input to this program will be integers in the range 1-10,000,000 this time (use a random number

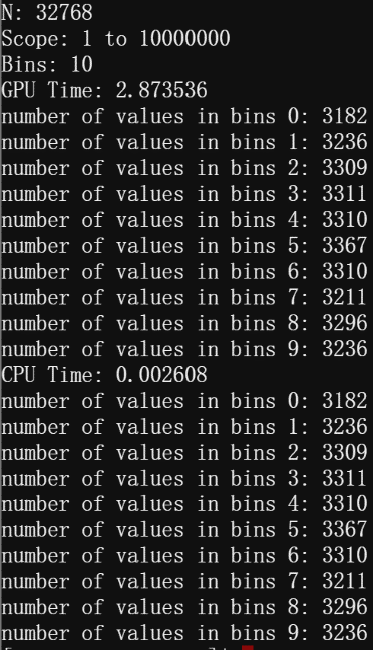
generator that generates the numbers on the host). Your host-based data set should contain N integers, where N can be varied.

1. This time you will implement a histogramming kernel in CUDA and run on a GPU. You can choose how to compute each class of the data set on the GPU. Attempt to adjust the grid size to get the best performance as you vary N. Experiment with N = 210, 215, 220 and 225. When the GPU finishes, print one element from each class in class ascending order on the host (do not include the printing time in the timing measurements, though do include the device-to-host communication in the timing measurement). Plot your results in a graph.

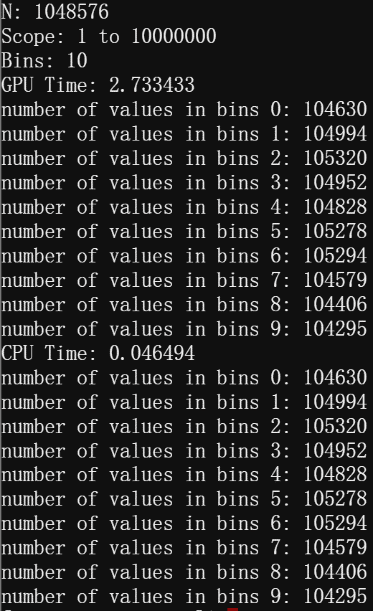
210:



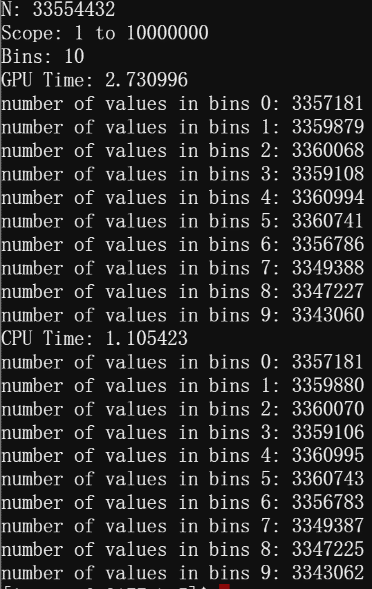
215:



220:



225:



1. Compare your GPU performance with running this same code on a CPU using OpenMP.

We can see that the GPU's runtime basically does not change drastically with a large increase in N and it is very stable. The CPU's runtime, on the other hand, rises significantly after a certain level of N. When the value of N is small, it runs faster with omp than with cuda, while at very large values of N, the advantage of gpu comes through.