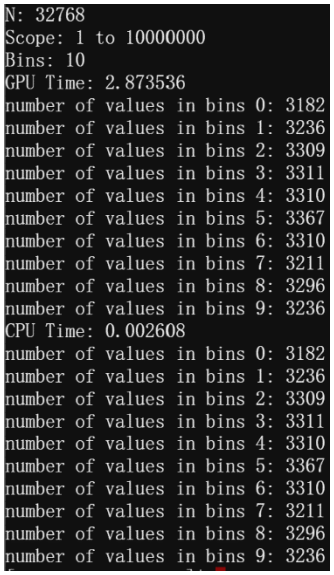
1. Let us revisit the histogramming problem assigned in Homework 4. Your input to this program will be integers in the range 1-10,000,000 this time (use a random number generator that generates the numbers on the host). Your host-based data set should contain N integers, where N can be varied.

a) 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 = 2^{10}, 2^{15}, 2^{20} and 2^{25}. 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.

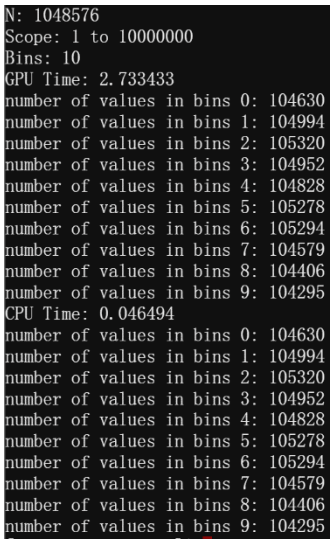
2^{10}:

```
N: 1024
Scope: 1 to 10000000
Bins: 10
GPU Time: 2.739113
number of values in bins 0: 123
number of values in bins 1: 98
number of values in bins 2: 115
number of values in bins 3: 87
number of values in bins 4: 105
number of values in bins 5: 111
number of values in bins 6: 104
number of values in bins 7: 107
number of values in bins 8: 72
number of values in bins 9: 102
CPU Time: 0.002147
number of values in bins 0: 123
number of values in bins 1: 98
number of values in bins 2: 115
number of values in bins 3: 87
number of values in bins 4: 105
number of values in bins 5: 111
number of values in bins 6: 104
number of values in bins 7: 107
number of values in bins 8: 72
number of values in bins 9: 102
```

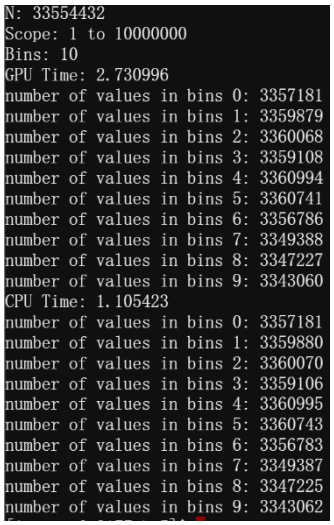
2^{15}:


N: 32768
Scope: 1 to 10000000
Bins: 10
GPU Time: 2.873536
number of values in bins 0: 3182
number of values in bins 1: 3236
number of values in bins 2: 3309
number of values in bins 3: 3311
number of values in bins 4: 3310
number of values in bins 5: 3367
number of values in bins 6: 3310
number of values in bins 7: 3211
number of values in bins 8: 3296
number of values in bins 9: 3236
CPU Time: 0.002608
number of values in bins 0: 3182
number of values in bins 1: 3236
number of values in bins 2: 3309
number of values in bins 3: 3311
number of values in bins 4: 3310
number of values in bins 5: 3367
number of values in bins 6: 3310
number of values in bins 7: 3211
number of values in bins 8: 3296
number of values in bins 9: 3236

2^{20}:


N: 1048576
Scope: 1 to 10000000
Bins: 10
GPU Time: 2.733433
number of values in bins 0: 104630
number of values in bins 1: 104994
number of values in bins 2: 105320
number of values in bins 3: 104952
number of values in bins 4: 104828
number of values in bins 5: 105278
number of values in bins 6: 105294
number of values in bins 7: 104579
number of values in bins 8: 104406
number of values in bins 9: 104295
CPU Time: 0.046494
number of values in bins 0: 104630
number of values in bins 1: 104994
number of values in bins 2: 105320
number of values in bins 3: 104952
number of values in bins 4: 104828
number of values in bins 5: 105278
number of values in bins 6: 105294
number of values in bins 7: 104579
number of values in bins 8: 104406
number of values in bins 9: 104295

2^{25}:


N: 33554432
Scope: 1 to 10000000
Bins: 10
GPU Time: 2.730996
number of values in bins 0: 3357181
number of values in bins 1: 3359879
number of values in bins 2: 3360068
number of values in bins 3: 3359108
number of values in bins 4: 3360994
number of values in bins 5: 3360741
number of values in bins 6: 3356786
number of values in bins 7: 3349388
number of values in bins 8: 3347227
number of values in bins 9: 3343060
CPU Time: 1.105423
number of values in bins 0: 3357181
number of values in bins 1: 3359880
number of values in bins 2: 3360070
number of values in bins 3: 3359106
number of values in bins 4: 3360995
number of values in bins 5: 3360743
number of values in bins 6: 3356783
number of values in bins 7: 3349387
number of values in bins 8: 3347225
number of values in bins 9: 3343062

b) 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.