Final Project

Parallel Programming & Architectures

Consideration

- ✓ Your code is automatically graded using a script, and therefore, if your file/folder names are wrong you will receive a grade of **zero**. Please read and follow the instructions carefully. Common mistakes include
  - Different file or folder names
  - Different formatting of input or output
  - Not paying attention to case sensitiveness of C++ and Linux
- ✓ Go to the folder ~/the/project/ in your home directory on the server and put your codes in this directory and remove any compiled binaries and test cases.
- ✓ Make sure your code compiles and runs without any error **on the server**. Your grade will be **zero** if any compile or runtime error occurs on the server. **Any!**
- ✓ The provided test cases, examples and sample codes (if any) are only to better describe the question. They are **not** meant for debugging or grading. It is your responsibility to think of and generate larger and more complex test cases (if necessary) in order to make sure your software works correctly for all possible scenarios.
- ✓ Start early and don't leave everything to the last minute. Software debugging needs focus and normally takes time.
- ✓ Just leave your final programs on the server. **Don't** email anything!
- ✓ Your grade is divided into several parts. In all cases, if you miss **correctness** (i.e. your code doesn't satisfy desired functionality), you miss other parts (e.g. speed, coding style, etc.) too. This rule is applied separately for each section of the project. For example, your code might not be correct for K=100 but still you will get your grade for K=10.
- ✓ Talking to your friends and classmates about this take-home exam and sharing ideas are *OK*. Searching the Internet, books and other sources for any code is also *OK*. However, copying another code is **not OK** and will be automatically detected using a similarity check software. In such a case, grades of the copied parts are **multiplied by -0.5**. Your work must be 100% done only by yourself, and you **should not** share parts of your code with others or use parts of other's codes. Online resources and solutions from previous years are part of the database which is used by the similarity check software.

Nearest-Neighbour Distance Histogram

Grading (100):
Report: 15
Correctness: 25
Speed: 60
You will receive the speed grade only if the result is correct.

Nearest neighbor Search (NNS) is one of the most well-known learning algorithms. It is defined as a problem of finding the closest point among references to a query point q. The definition is shown below.

$$nearestneighbor = \underset{x}{argmin} d(q, x)$$

Where d is a distance function, and x is a vector in the D-dimensional reference points.

In some problems, all distances to the neighbors have information instead of jsut the nearest one. In this case, calculating the histogram can be a possible solution. A histogram is a plot that lets you discover the underlying frequency distribution of a set of continuous data.

To find the histogram of distances between a query point q and reference points x, we can divide the algorithm into three major stages. First, the distance between q and each reference vector should be calculated. Second, we should find the maximum and minimum of the distance array to get the lower and upper range of intervals, called bins. The third step is to split the data into the bins.

In this project, you have 1,000,000 reference vectors which have 128 dimensions. The test query consists of 10,000 vectors with similar dimensions. The distance function is Euclidean distance (L2-norm). It defined for each a = (a0; a1; ...; an-1) and b = (b0; b1; ...; bn-1) vectors as shown below.

$$Euclideandistance = \sqrt{\sum_{i=0}^{n-1} (a_i - b_i)^2}$$

Fig.1 shows an example in a two-dimensional space. The query point q is (1,2). The nearest point is (2,2) whose distance to the query point is 1. Similarly, the maximum distance is 5, which belongs to the point (4,-2). The number of bins is given 4; so, the bins array is [1, 2, 3, 4, 5]. By counting the points in each interval, we can find the final histogram as [5,7,5,3]. Note that all but the last (righthand-most) bin is half-open. In other words, in the above example, the first bin is [1, 2) (including 1, but excluding 2), the second is [2, 3), and the third is [3,4). The last bin, however, is [4, 5], which includes 5.

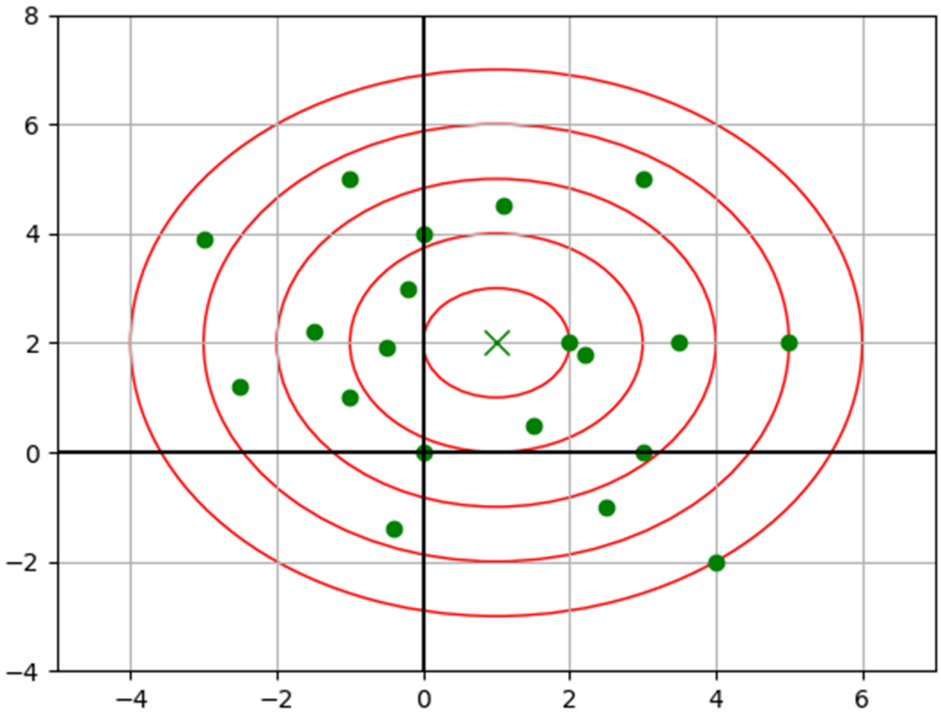

Fig.1

Implement the above "Nearest-Neighbour Distance Histogram" algorithm on GPU. You should calculate the defined distance for each query point q with all reference vectors. Then, the histogram should be returned as an array of K elements. K is the number of bins, and it is given as an argument in the command. You should write the outputs array for all query points to a file in your project directory (□/the/project/outputs.ivecs) for evaluation.

You will earn the speed's grade in comparison to other students, i.e., the student who achieves the best time will receive a full grade. To improve the speedup you can use the following ideas:

- Exploit the shared memory to reduce the main memory access.
- Try to find the best parameters for the number of blocks and threads. You can use the same idea as the blocked matrix multiplication which you had in previous take-home-exams.
- Consider various K in the range of $1 \leq K \leq 5000$ with which the code will be tested. Use some special cases for smaller K.

You are also encouraged to use any ideas which are not mentioned above. However, you are not allowed to use the approximate approaches to find the distances. Your task is developing a code to find the exact histogram of neighbors.

You should report all the ideas which you have utilized to improve your code's speed. Write it in a pdf file named nn_studentID.pdf and put it in the project directory (□/the/project/nn_studentID.pdf). The "studentID" should be replaced by your own student ID.

Use the provided gpuerrors.h, gputimer.h and nn.cu files to start your work. You can modify any part in nn.cu. For grading, we will execute the code with N = 10000 and various values for K smaller than 5000. N denotes the number of test vectors in the query, and K is the number of histogram bins in the algorithm.

Compile: *nvcc -O2 nn.cu -o nn*

Execute: *./nn N K*