Joe Kuczek
CS 311 PA1

CS 311 Programming Assignment 1 Report

1.) Algorithm for the method buildDataStructure()

- First calculate the required size of the hash table, size > 1.5n where n is the number of points from the input.
- Create the Hash Table by finding a prime number p >= size, set the size of the hash table to be this prime number.
- For each point *q*, calculate *floor(q)* and use this as the key for a Tuple *t*. The value of the tuple will be the point itself. Store *t* in the hash table in position h(*floor(q)*), where h is the hash function for the table.

2.) Algorithm for the method npHashNearestPoints(float p)

- Create an empty ArrayList for storing nearest points.
- First, find the 3 possible buckets any potential neighboring points could lie in. Do this by calculating h(*floor(p-1)*), h(*floor(p)*), and h(*floor(p+1)*), and looking the keys up in the hash table to get the corresponding buckets.
- For each bucket, loop over all the Tuples. Compare the value of the Tuple (q), with p. if Abs(p-q) <= 1, then we have found a nearest point, so we add it to our list.
- Return the ArrayList

3.) Runtime comparison

- Naive: 212,614 ms => 212.61 s => 3.5 minutes
- Hash: 156,055 ms => 156.05 s => 2.6 minutes
- As our input size increases, we will see a greater separation between the two runtimes. This is because the naïve solution is O(n) while the hash solution is O(N(p)) where N(p) is the number of nearest points to p.
- Note: we are including the time it takes to write a file in the runtimes.