Taylor Dugger

COM S 311 Programming Project 1 Report

25 February 2017

Programming project 1 consisted of writing several classes all for helping in the end to find nearest points of a list of points and creating a recommendation system for movies. One of the key classes for NearestPoints was buildDataStructure(). First you have to create our HashTable class which this structure will use. First, I had to create a new hashTable of size m > 1.5*n. Since we need an integer for the hashTable construction, I used the ceiling function after I multiplied it by 1.5. The next step is to go through the list of points, getting your first point and setting as p. I then take p and use the Math.floor function on it to obtain our key. Next you create a new tuple with our k and p values and add to the hash table. Hash table takes care of most of the work here.

We need to find the nearest points from a point p. So, I created a function called npHashNearestPoints which uses our HashTable and buildDataStructure to find nearest points in a very efficient time. First, you must convert our input p to a key value to look in our table. So, we use the floor function again to obtain g, our key to search for. Since a nearest point is < 1 from our point, the floor function could place our nearest points into three different buckets. One below our point, at our point, and one after our point. Now we need to just search these three buckets instead of every bucket. So, we just go through each one and do a hashTable.search() on it. Then we need to see if it's a nearest point since we could potentially get a point slightly outside our 1 range. So, we just do a Math.abs(p-q) <= 1. If that is true, then add it to our new array of closest points.

When we look at run times of project 1 we need to compare how allNearestPointsHash and allNearestPointsNaive run. If you do a brute force way we can see that the Naïve function runs in about 36 seconds. When we use our hash table, all nearest points hash runs in just around 500 ms, less than a second. Using an efficient data structure like hash compared to a brute force way speeds up our run time exponentially.