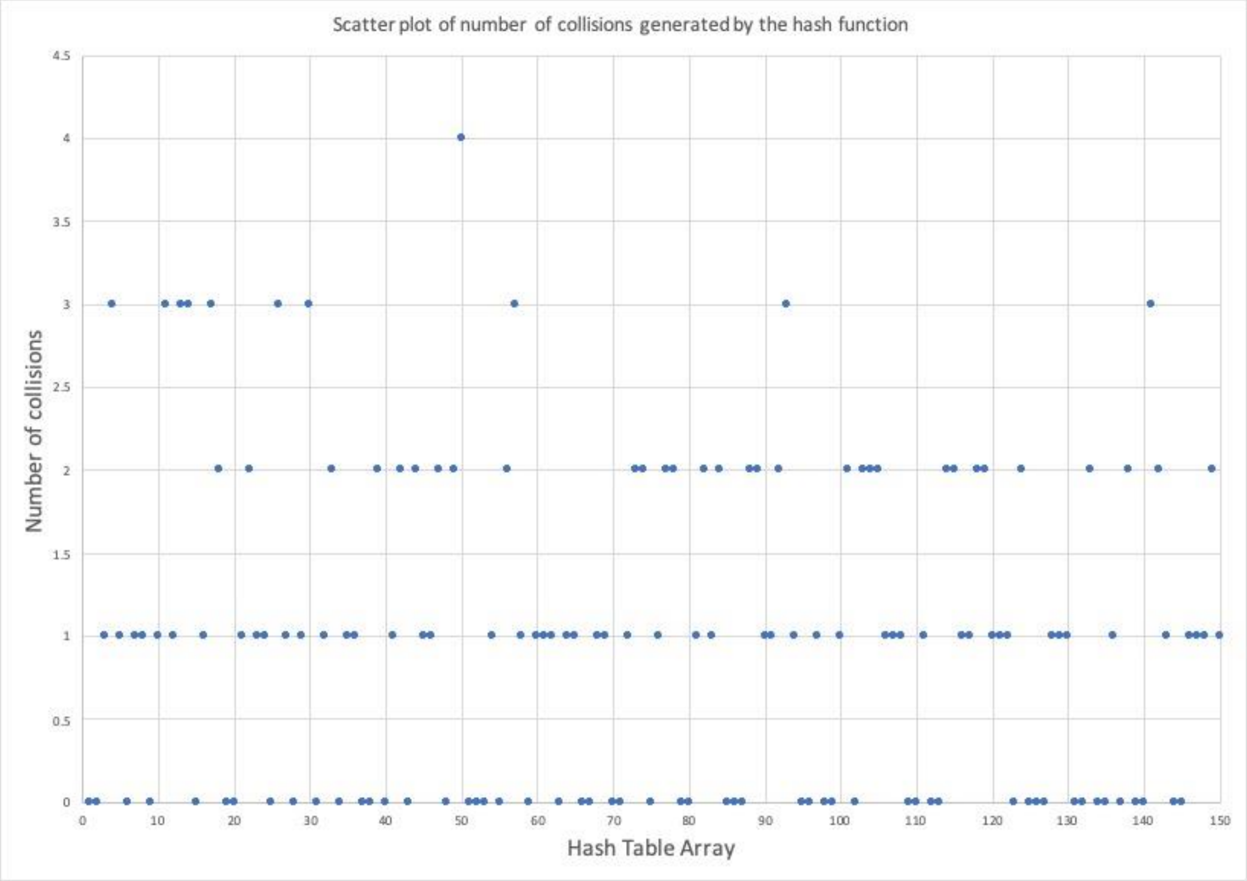
Part 2 - Hash Function Analysis

a) Analyzing the hash table containing the pointers to the all the student objects in the institution object
b) The indexing key used is the student number
c) Student number was chosen because every student will have a unique student number

Results:

What does the range of hash table array indices produced by our hash function look like?



The hash function creates a decent spread of indices across the hash table. Unfortunately, there are a lot of indices that are not used. However, the majority of the indices are only hashed once and twice. The maximum number of collisions is 4.

Can we predict where all the other indices (not produced in our experiment) would "land"?

It is not possible to predict where the indices will land because there is no clear pattern in the type of student number that would make it easy to predict. However, it is noted that the indices that have collided all end with the same digit. Since a folding technique is used, similar looking student numbers will have different indexing keys.

Does our hash function evenly distribute indexing keys across the range of hash table array indices? Do we obtain a 1-to-1 mapping?

The hash function does a mediocre job distributing the indexing keys across the range of the hash table array. There are 54 empty indices in the hash table array implying that there is no 1-to-1 mapping. The hash function produces a maximum of 4 collisions per index. From the scatter plot above, it is clear that the indexing keys are distributed evenly, and the majority of the keys are created only once.

Is our hash function producing collisions? Why?

Yes, the hash function is producing collisions. This is happening because arithmetic on the student number may not be unique. Since we are using a folding technique, the arithmetic can be identical to a student number is that looks different, but because of the arithmetic the hash function performs on it, it produces the same hash code.

What do synonyms look like? Give some examples of synonyms.

It is hard to tell what synonyms looks like as explained above in the section regarding the prediction of indices. This is due to the folding technique and how the arithmetic plays out. There is no “pattern” that generates synonyms.

Examples of synonyms:
The following 8-digit student numbers produce the same indexing key 49:

1. 19988709
2. 54274459
3. 84388009
4. 74717279

How easy each version is to compute, by expressing its time efficient using the Big O notation.

The hash function is very easy to compute. It can be computed in O(1) because we are using folding hash functions that does simple arithmetic on parts of the indexing key.