**Assignment 5**

**Design:**

The core design of this project is by collecting time data used in different situation.

**Hash design:**

The hash class is designed to contain 2 arrays which can take keys and values. It also has a some instance variables like length of hash table and load factor.

**Other design:**

To make the result more objective, I use random integer varying from 0 to 100000.

**Analytics:**

1. *Same Input, Different Hashing (Without Resizing):*

All of these hashing has length 1000. From the graph we can see that the slope of three hashing are different. These data are collected in the case that inputs are done from 10000000 to 100000000. Linear Hashing which is the blue line has the highest slope because with the extend of data, linear search because more time-consuming. The fastest hashing is quadratic hashing in my research because it can efficiently accelerate finding position to insert the key-value pair. Double hashing behaves medium in this case in which I choose 97 as R of hash2().

1. *Same Input, Different Hashing (With Resizing)*

All of these are done with load factor 0.5. Each line represents a type of hashing(), number of input increasing from 100000 to 1000000. This time, I add resizing, which apparently, cause the results flapper violently. Because resizing can avoid hash map becoming too crowded. This is an effect way to improve performance when the data piles too many.

1. *Same Linear Hashing, different array size (With Resizing)*

All three line come from linear hashing with same load factor-0.5, but their array size(at the beginning) differ from 100, 1000 to 10000. As you can see, they are very similar. with resizing, hash table can effectively resist the impact of length of hash table.

1. *Same Array Size, different load factor (With Resizing)*

In this case, I let all array size to be 1000 while load factors vary from 0.1 to 1 by pace 0.01. All these data are collected from three kind of hashing. As you can see, they are almost the same. The breakpoint appear at the position where load factor is about 0.67. After this value, the performance of the hash map will become extremely worse. Before that are almost the same.

1. *Same load factor, different Array Size(With Resizing)*

As you can see with resizing, the size of array at the beginning has nearly no impact on performance.

1. *Create New table instead of Resizing*

As you can see, the higher one is the time of hashing by creating a new class of hashing.

Because this will make hashing class recursive including copying large number of data and using large spaces, the time consumed is obviously more than resizing.

**Conclusion:**

There are four elements may affect the performance of a hashing.

**Load factor:** usually under 0.7. When too small, it will increase the time to resize the hash map. When too large, time consumed will increase violently when table is almost full.

**Array size:** (Without Resizing) When array size is too small, the table will be filled full immediately. When array size is too large, there will be waste of memory.

(With Resizing) With resizing, the impact of array size at the beginning will become very small.

**Number of inputs:** More input, more time.

**Resizing:** Resizing can minimize the impact of initial array size.