Analysis for Assignment 10

CS 2420

Name: Shuhao Wu

UID:0793268

1)
Q: What does the load factor λ mean for each of the two collision-resolving strategies (quadratic probing and separate chaining) and for what value of λ does each strategy have good performance?

A: For quadratic probing, load factor means the proportion of the whole memory space occupied by added items. And we need $\lambda < 0.5$ to guarantee that every spot will be examined at least once and no cell is visited twice. For separate chaining, load factor means average length of linked lists. We want a smaller λc to have a good performance.

2)
Q: Give and explain the hashing function you used for BadHashFunctor. Be sure to discuss why you expected it to perform badly (i.e., result in many collisions).

A: The bad hashing function I used always returns to 0, it makes everything's hashcode to be 1, result in many collisions.

3)
Q: Give and explain the hashing function you used for MediocreHashFunctor. Be sure to discuss why you expected it to perform moderately (i.e., result in some collisions).

A: The mediocre hashing function I used returns the length of the input string, it better than last one, but it will result in some collisions because most of English words' lengths are approximately from 1 to 10.

4)
Q: Give and explain the hashing function you used for GoodHashFunctor. Be sure to discuss why you expected it to perform well (i.e., result in few or no collisions).

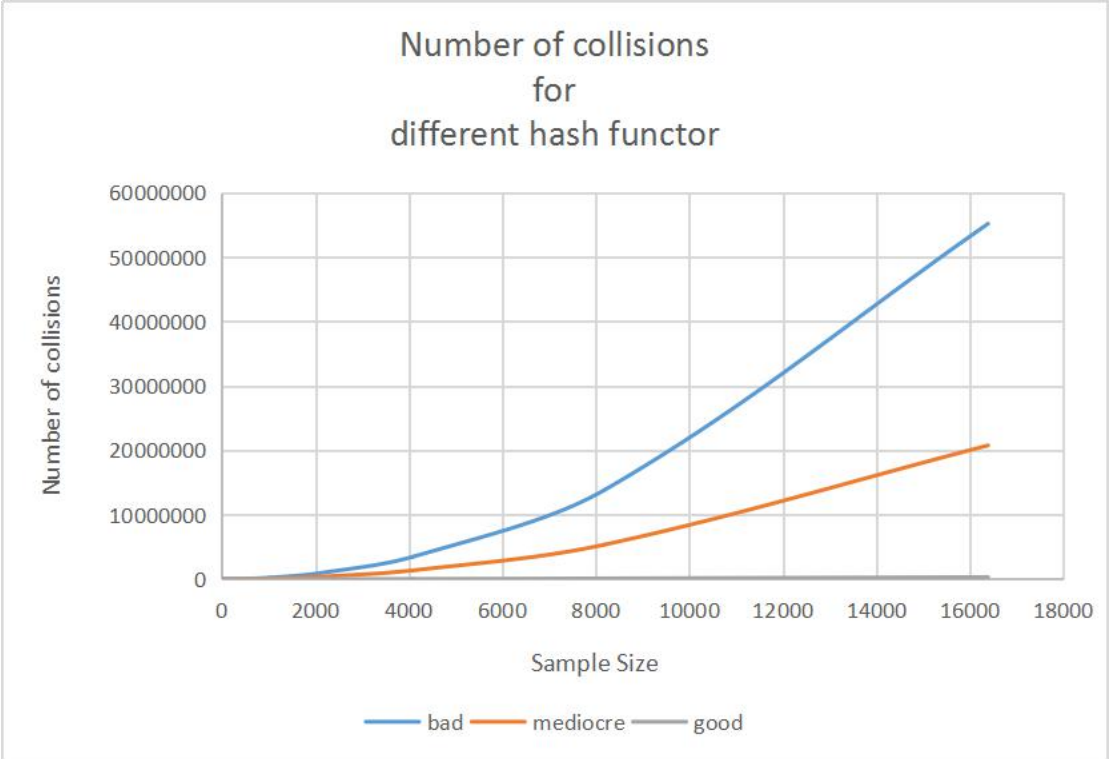
A: The good hashing function I used returns the sum of ASCII chart index number of every single char in the input string. This one is better than above two, every word was composed by different chars, so they have different hashcode, it results in few collisions, collisions only occurs when words are anagram to each other.

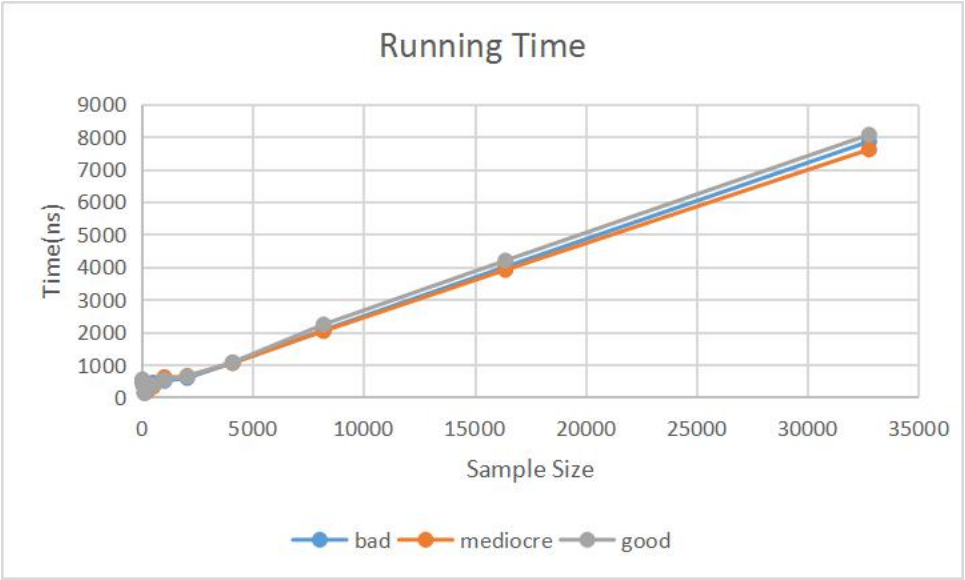
5)

Q: Design and conduct an experiment to assess the quality and efficiency of each of your three hash functions. Carefully describe your experiment, so that anyone reading this document could replicate your results. Plot the results of your experiment. Since the organization of your plot(s) is not specified here, the labels and titles of your plot(s), as well as, your interpretation of the plots is critical.

A recommendation for this experiment is to create two plots: one that shows the number of collisions incurred by each hash function for a variety of hash table sizes, and one that shows the actual running time required by each hash function for a variety of hash table sizes. You may use either type of table for this experiment.

A: I choose QuadProbeHashTable to do this experiment. First I wrote a method can generate random String, than I created the increased sized(form 2^5 to 2^15) random String into an ArrayList, then I call hastable's addAll method, and print its number of collisions and the average running time. The output is below.



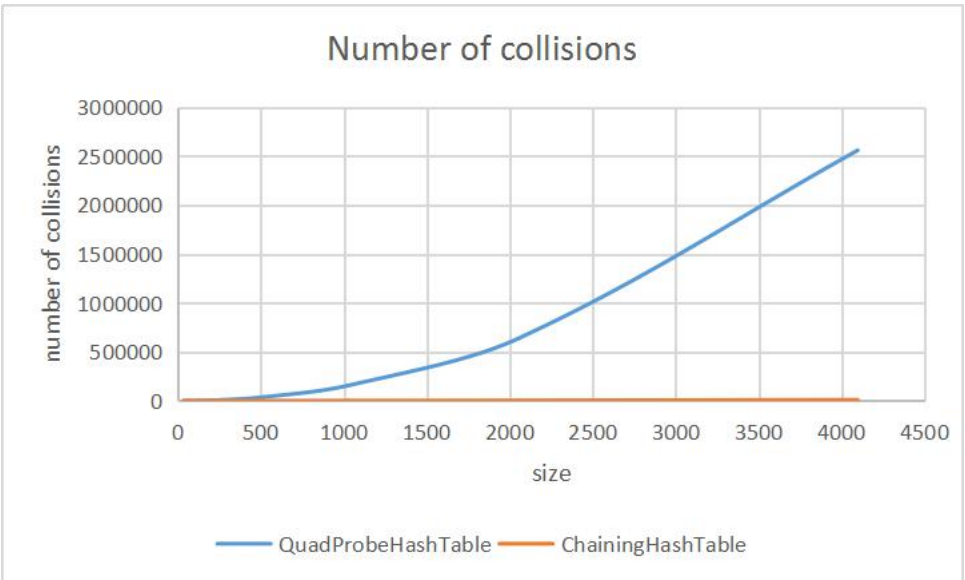


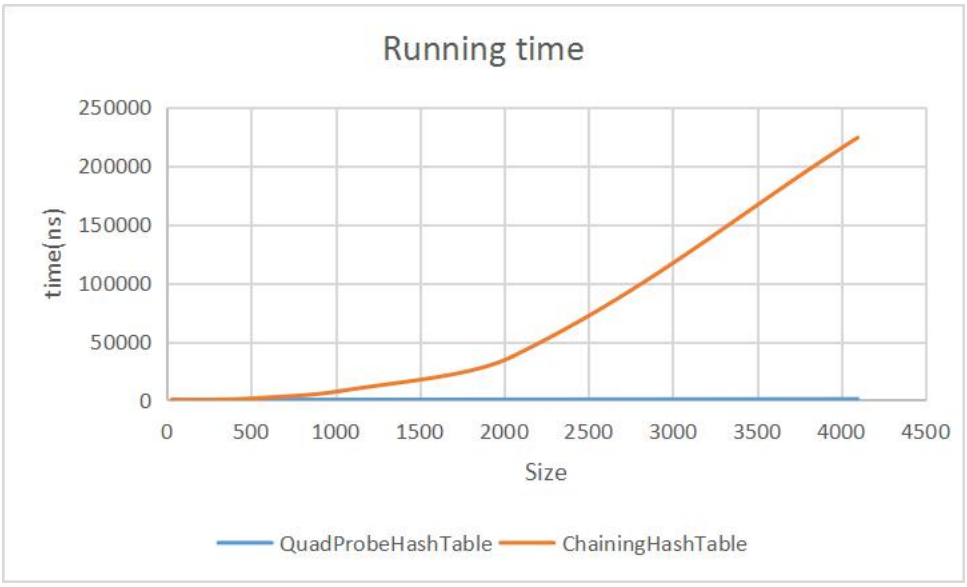
6)

Q: Design and conduct an experiment to assess the quality and efficiency of each of your two hash tables. Carefully describe your experiment, so that anyone reading this document could replicate your results. Plot the results of your experiment. Since the organization of your plot(s) is not specified here, the labels and titles of your plot(s), as well as, your interpretation of the plots is critical.

A recommendation for this experiment is to create two plots: one that shows the number of collisions incurred by each hash table using the hash function in GoodHashFunctor, and one that shows the actual running time required by each hash table using the hash function in GoodHashFunctor.

A：The process is similar with the last experiment. Output is below.





7)

Q: What is the cost of each of your three hash functions (in Big-O notation)? Note that the problem size (N) for your hash functions is the length of the String, and has nothing to do with the hash table itself. Did each of your hash functions perform as you expected (i.e., do they result in the expected number of collisions)? (Be sure to explain how you made these determinations.)

A:

Bad : O(C) its directly return a int.

Mediocre: O(C), it calls string.size().

Good: O(N), it checks every single char in the String.

Yes, it is as my expect. Bad has many, Mediocre has less, Good has few.

8)

Q: How does the load factor λ affect the performance of your hash tables?

A: It obviously increases the performance, I did not set a Load Factor for Chaining Hash Table, it has a really bad performance, but QuadProbe has, so it is much better.

9)

Q: Describe how you would implement a remove method for your hash tables.

A: Same as insertion, check the index hashcode, if target is not there, use QuadProbe, like +1, +4,+9, until you found it, and set it to null.

10)

Q： As specified, your hash table must hold String items. Is it possible to make your implementation generic (i.e., to work for items of AnyType)? If so, what changes would you make?

A：I think I just need to change the functor to make sure it can calculate the hash code for every type of input.

11) 10 hours.