

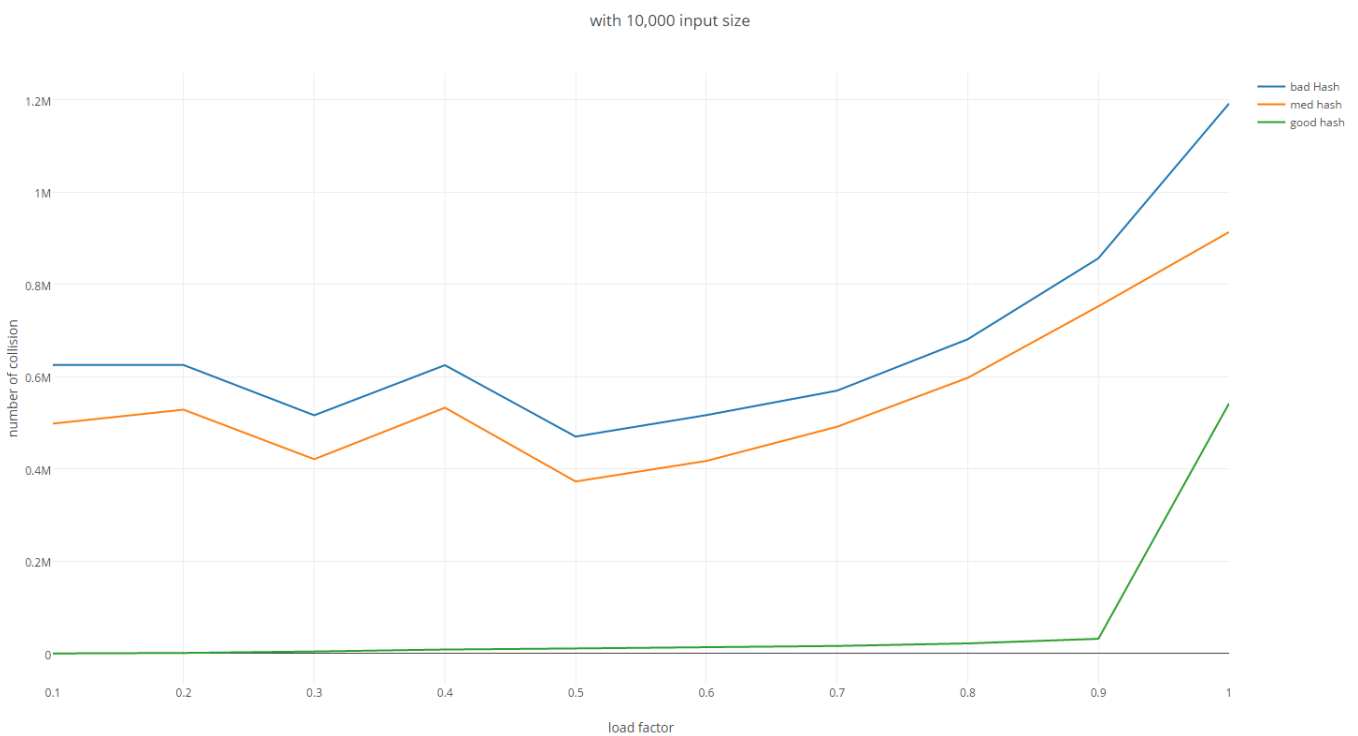
Jiwon Nam(u0950753)

When you are satisfied that your program is correct, write a brief analysis document. The analysis document is 30% of your Assignment 10 grade. Ensure that your analysis document addresses the following.

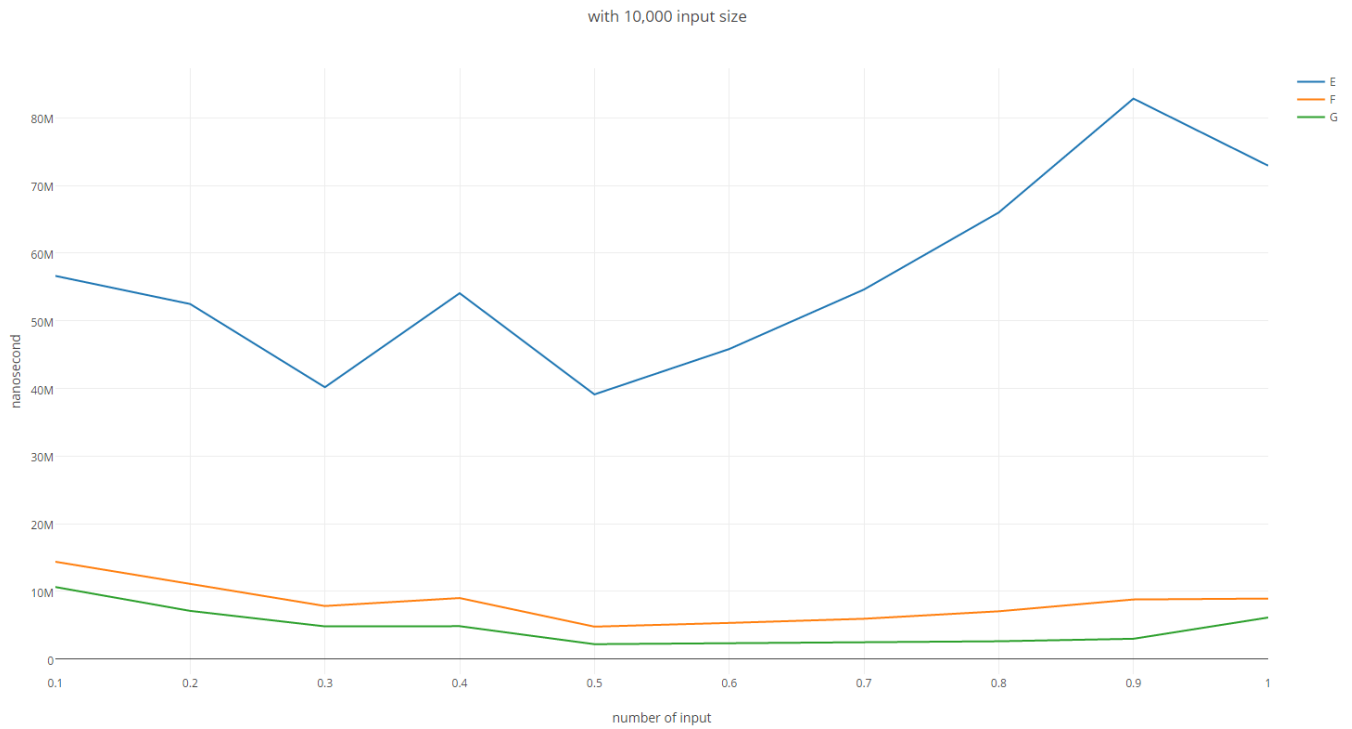
1. What does the load factor λ mean for each of the two collision-resolving strategies (quadratic probing and separate chaining) and for what value of λ does each strategy have good performance?

For quadratic probing, λ is the portion of the space filled in the specific capacity. Depending on the load factor, the quadratic probing hash table should rehash since it prevent lots of collision. The reason is that if the hash table have filled 9 space out of 10 capacity, even though the hash functor is really nice, it will occur collision to find index position. smaller capacity frequently occur collision if the spaces almost filled.

Therefore, λ controls preventing lots of collision, so if load factor is bigger than specific strategy portion, the hash table should rehash with larger capacity. Here is my graph to check depending on load Factor, how affect to run-time with same input size (1,000 to 10,000) increasing with 1,000. I set load factor 0.1 to 1.0 increasing 0.1



From this chart, I can say that at load factor is 0.5 was the lowest collision occurred.

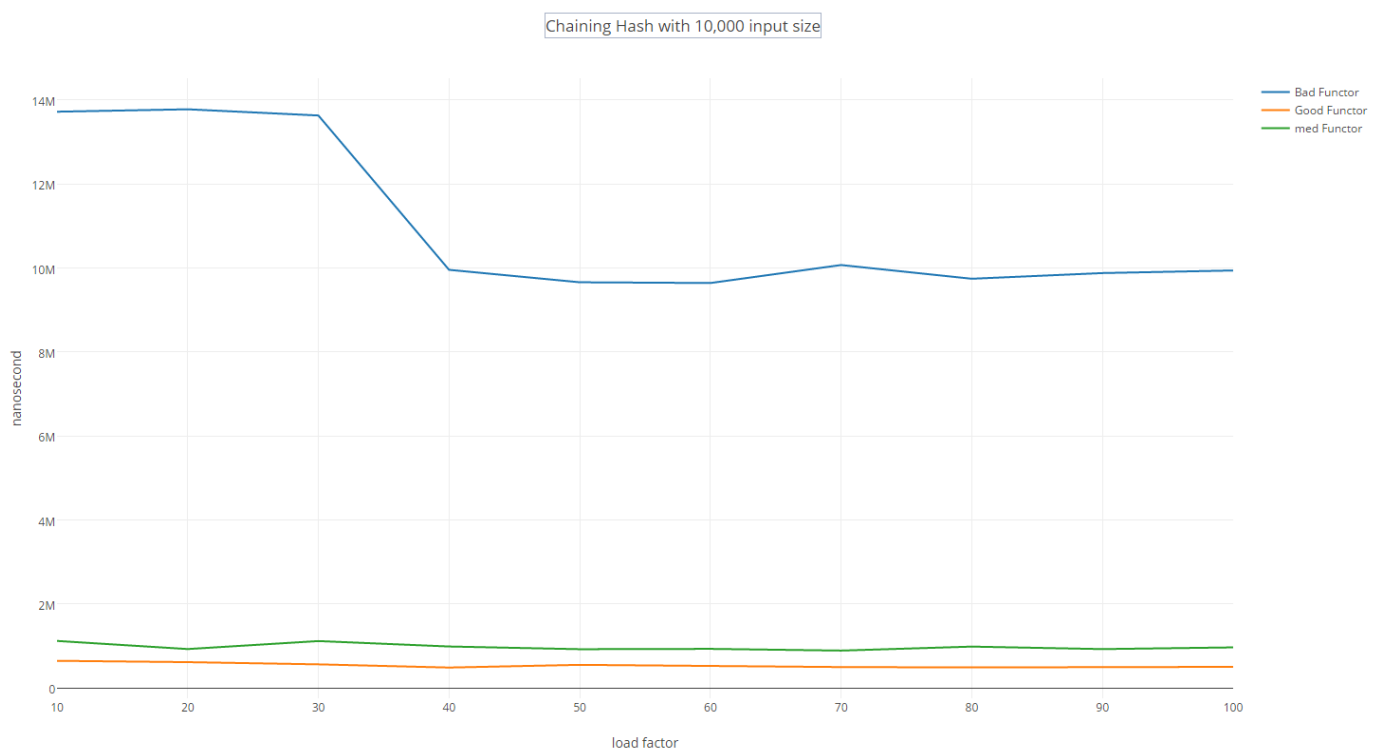


This is the result of timing graph for each load factor.

From those graph, at load factor is 0.5, it is the best case for each cases.

For operate chaining, λ (load factor) is the average length of linked list. it prevent lots of rehash operating to avoid big run-time processing. the load factor is not a portion, but it should include size and capacity to find average length of all linked list in the set. if I don't set load factor, the hash set will have huge size of linked list with same capacity, and it causes huge run-time to find item with linked list ($O(N)$). However, if i set specific number to load factor, and rehash if load factor is bigger than. Here is graph to check how the relation between load factor and run-time is.

I set 1,000 to 10,000 input items increasing with 1,000. I set load factor 10 to 1,00 increasing by 10.



From 40 to 100, it doesn't matter to get how much load factor is, it doesn't affect to speed of add method for chaining hash set.

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

I set bad hash functor as sum of each character's ascii numbers divided, and then dividing the sum with three. The reason is that if I input every character, it will have every different numbers. However, the bad thing will happen if sum of characters are easy to have same number with other combination of characters. It cannot get specific number each like "ay" will same with "mm". even if capacity is smaller, it is easy to make collision for this case. Therefore, I set as bad hash functor like this.


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

To prevent same index case in bad functor class, I set each character multiple by 2 or 3 depending on their string position. For this progressing, it will return even or odd number depending on their position and characters. It return better collision numbers than bad hash functor.

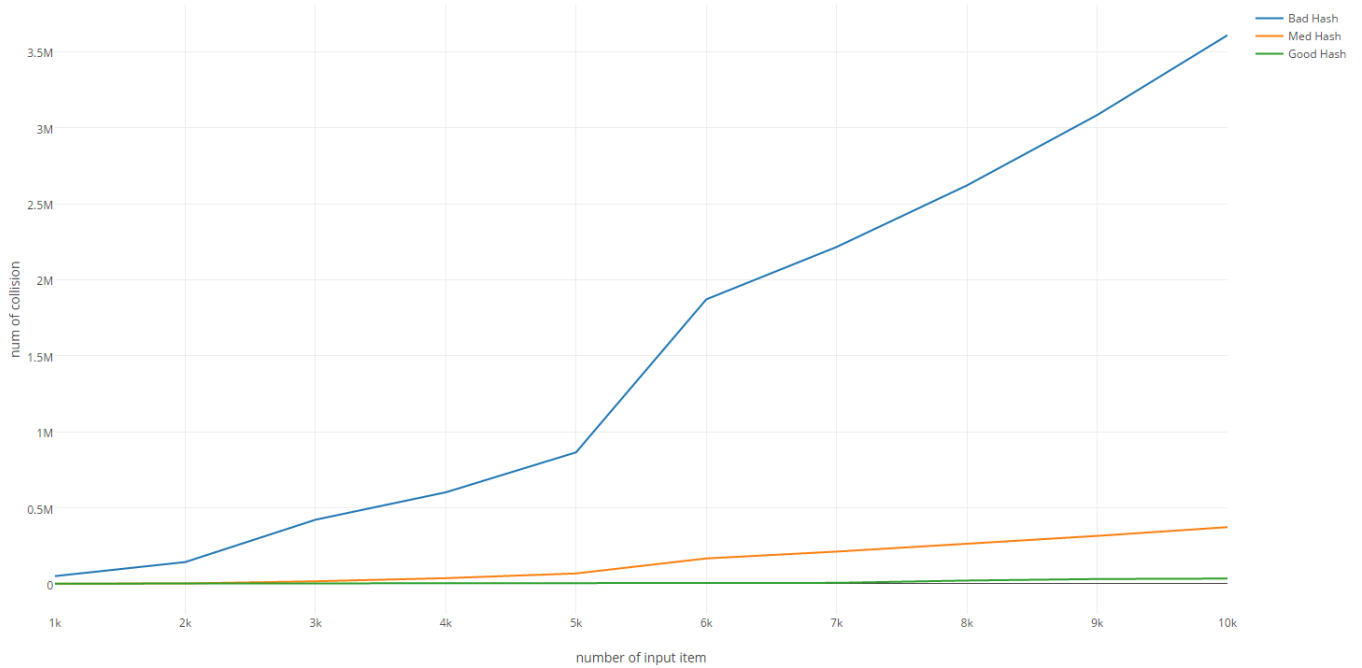
4. Give and explain the hashing function you used for GoodHashFuncutor. Be sure to discuss why you expected it to perform well (i.e., result in few or no collisions).

For good hash functor, I made new integer array that are prime number from 2 to 23. For the hash index, sum of each character times first prime number and last prime number. From this result, I can multiple each character with different number end with 0 to 9 every cases. $2 * 23 \% 10 = 6$, $3 * 19 \% 10 = 7$, $5 * 17 \% 10 = 5$, $7 * 13 \% 10 = 1$. like this, I don't have any same number to multiple each character so that it can get unique value for each word.

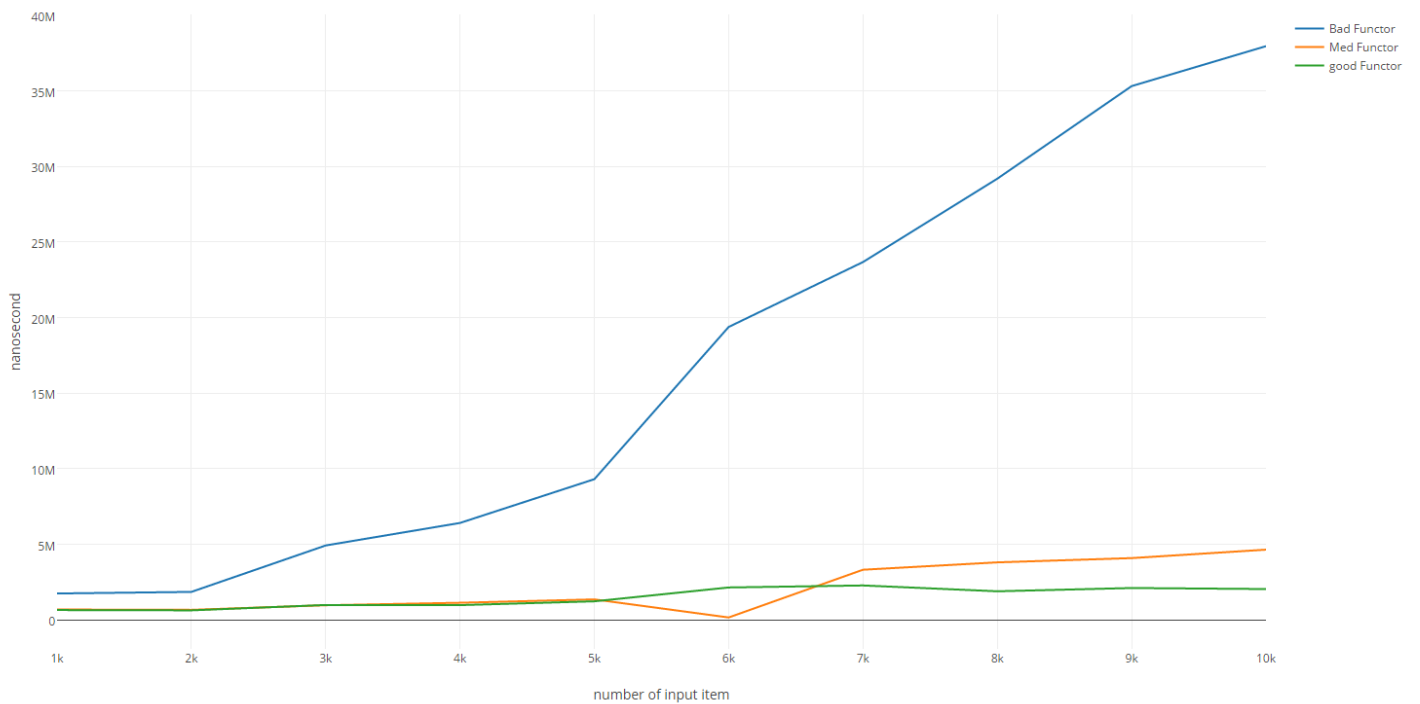
5. Design and conduct an experiment to assess the quality and efficiency of each of your three hash functions. Carefully describe your experiment, so that anyone reading this document could replicate your results. Plot the results of your experiment. Since the organization of your plot(s) is not specified 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.

quad Hash number of collisions input 1,000 to 10,000



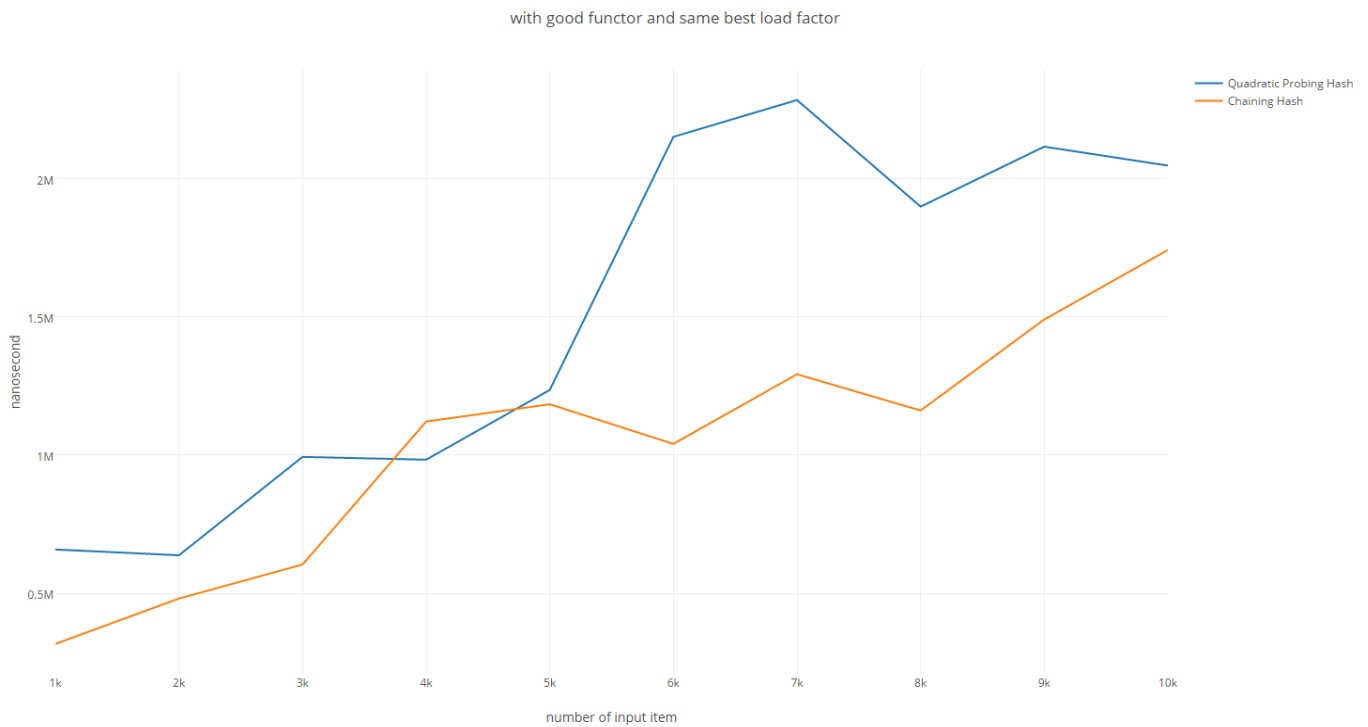
quad Hash time graph input 1,000 to 10,000



From the graph, those two graph looks really similar shape. If it is bad hash functor, it will occur lots of collision. From lots of collision, it also cause lots of time to perform the add method. The Bad hash functor looks like $O(N)$ graph, but med and good functor look like less than $O(\log N)$, and approach as $O(c)$ graph.

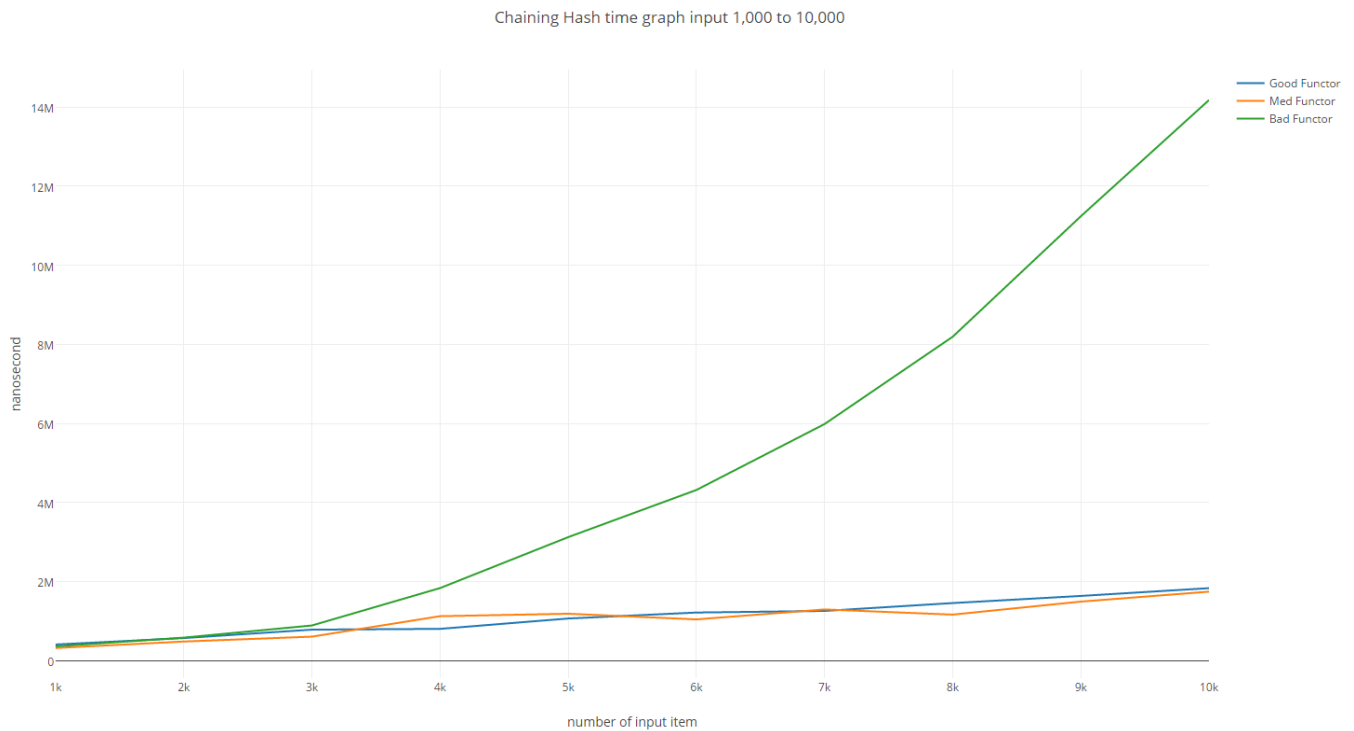
6. Design and conduct an experiment to assess the quality and efficiency of each of your two hash tables. Carefully describe your experiment, so that anyone reading this document could replicate your results. Plot the results of your experiment. Since the organization of your plot(s) is not specified 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.



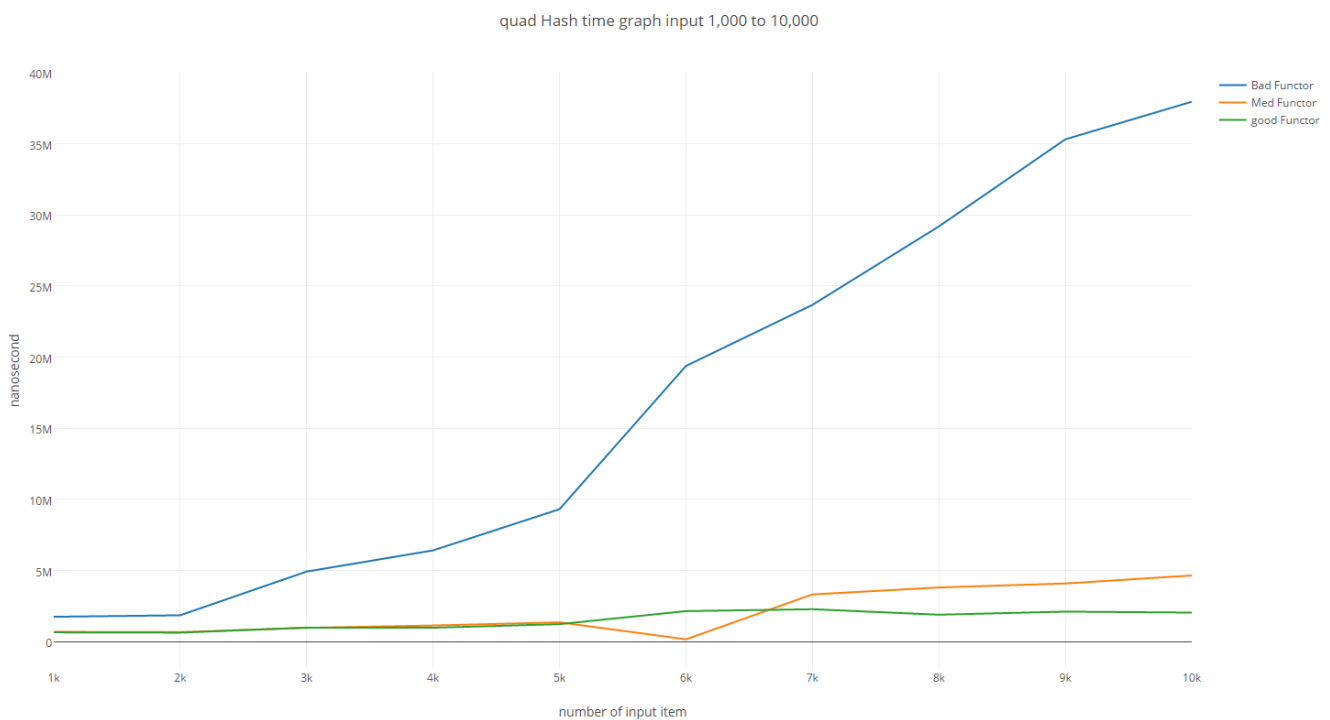
From the result of graph, chaining hash table is faster than quadratic probing hash table slightly. The reason is that because quadratic probing need to rehash more time than chaining hash table. From rehashing, it cause run time complexity lise $O(N)$, so it charge more time to complete the program.

7. 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.)



From the graph, Bad hash functor looks like quadratic increasing, but med and good hash functor are similar graph approaching to less than $O(\log N)$ shape. It is not $O(C)$, but it approach to constant.

From question 5 second graph, I can get graph like this too.



From those two graph, for my Bad Functor, the big O complexity is $O(N)$. Med and Good functor graph looks like $O(\log N)$ because when they rehash, the big-O complexity will be $O(N)$ to copy all table, but not often. Therefore, I can say $O(\log N)$ for two cases.

8. How does the load factor λ affect the performance of your hash tables?

Load factor affects to performance if it is extremely bad such as 0.1, 0.2 or 0.8, 0.9. From this load factor, it occur really lots of collisions, so it affects to slow performance to hash table.

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

If I made remove method for hash table, to find index if it has same item in the table, get an index for the item. In my hash table, I can get index position with returning negative value of index except index 0. First check index 0 item, and then if not, find negative index of the position to escape the loop in quadprobe method. Set the negative value to absolute value to indexing in array, and then remove the value in the array. Also, reduce size and return true. This is my strategy.

10. 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?

It is possible. I can make the LinkedList with generic type or make object array with some generic type. Comparing method doesn't need to be positioned in hash set class, but for Hash functor method, I need to make instance thing to return some integer value for input generic type value.

11. How many hours did you spend on this assignment?

15 hours.

Programming partners are encouraged to collaborate on the answers to these questions. However, each partner must write and submit his/her own solutions.

Upload your solution (.pdf only) here by 11:59pm on November 9.