Ryan Bolander u0016911

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 Î» mean for each of the two collision-resolving strategies (quadratic probing and separate chaining) and for what value of Î» does each strategy have good performance?

> The load factor just tells us the percentage of the array that is currently filled. So our number of slots filled with items over the number of available slots in the array. For our quadratic probing we kept the load factor below 50% or 0.5. We did this in order to make sure that during the quadratic searching to place an item that not only do we find a spot but that it doesn't end up in an infinite cycle trying to find an empty spot. Having the load factor at this percentage helps ensure that we always have spots but that we don't have too many or too little.

2. 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).

> I just used the length of the string for the bad hash function. This would be bad because you could easily end up with a lot of string lengths being the same and running into a lot of collisions. Especially if the strings are something like a dictionary word list.

3. 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).

> For my mediocre hash function I used the value of the first character plus the length of the word. This will result in fewer collisions than the bad hash function. However, since it is still dependent on the length and there could be a lot of words that are the same length with the same first letter. Or even the same sentence with the same first letter and length. You could still run into a lot of collisions but less than the bad function.

4. 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).

> For the Good hash function I added up the value of each character in the word. Considering all the different possibilities you will run into much fewer collisions using this method than the previous two.

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. â ¨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.

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. â ¨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.

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.)

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

The load factor assures that we have the available space to always be able to add the items. It also helps make sure that we don't have too many spots available as it creates wasted memory. But we also need to have enough spots that we can actually find a spot as well. Otherwise you run the risk of not having a spot to add your item that needs inserting. So, this affects performance because it also affects how often you have to change your size of the array as this takes more time to re-hash everything.

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

I would use the contains method to find the item and then remove that item.

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?

To make this generic I feel like there would have to be quite a few changes. Obviously making it a generic item. But also making the hash functions perform differently because they are all assuming that it's a string at this point. So you would have to determine what the generic item was and then based on that choose the best hash strategy for good functionality.

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

8 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.