Casey Nordgran
CS 2420 – Section 004
Summer 2014
U0887369

Assignment 8 Analysis

CS 2420 – Summer 2014

1. Who is your programming partner? Which of you submitted the source code of your program?

   Cody Cortello is my programming partner. Of the two of us, <> will be submitting the source code of our program.

2. Evaluate your programming partner. Will you work with this partner again? Have you worked with more than one partner? Switching at least once is a requirement, and there are just two assignments left.

   Cody is a good hardworking person and a great partner. He is very easy to work with and we have never had confilicts with our assignments. As I did not know java before this class, he has also taught me a lot about java programming and has been very patient with me. He knows the material very well is creative with his implementations. I have not worked with more than one partner yet. I would like to work with Cody again and we will have to try and switch partners once for the next assignment possibly.

3. 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?

   The load factor for the hash table that uses quadratic probing to resolve collisions means the fraction of the table capacity that is filled with elements. For the table that uses separate chaining the load factor represents the average size of the LinkedLists in the LinkedList array. The hash table using quadratic probing has good performance as long as the load factor stays below 0.5. The hash table using separate chaining maintains good performance for load factors of a much greater size, on average it should be around 1.0 as the textbook states but can go much higher and still maintain great performance.

4. 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).

   For the BadHashFunctor, the hash method simply takes the string object and returns an integer that is the size of the string. This method should certainly perform badly because there could be many different strings that are not equal to each other but have the same length.

5. 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 the MediocreHashFunctor, or in our source code it is call FairHashFunctor, the hash method takes the string object, then sums all the integer values of the separate characters in the string, than multiplies that final sum with the length of the string. This hash functor is expected to perform moderatly because it produces a hash code that is more unique than the BadHashFunctor because many strings will have the same length but produce a different integer with this method. However, it is still not the best method because there could be many different strings the will still produce the same hash code given this method.

6. 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 GoodHashFunctor, the hash method is to convert every character in the specified string into it's equivalent decimal representation in ASCII, then create one large integer as a BigInteger data type by placing all the integers of the characters side by side in one large string. Before returning the value, the BigInteger is converted to the type 'int' then the absolute value of this int is returned. This hashing function is expected to be a good hash function and better than the previous two because not only does it produce an integer that depends on the length of the string and the characters used, but it also depends on the order those characters are placed in the string.

7. 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

growth of 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.

8. 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.

9. 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.)

The cost of each of our BadHashFunctor is O(1), this is because all it does is return the length of the string object which is a constant time operation. The cost of our FairHashFunctor is O(N) because it mush visit each character in the string and sum their ascii values. It also calls the length of the string but this is a constant operation and so the over all cost is still O(N). The cost of our GoodHashFunctor is expected to be also O(N) because it too mush visit each character in the string as explained questions six. This functor also calls multiple other operations which are constant time but this still should leave the overall cost as O(N).

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

It does not effect the performance as much of our hash table that uses separate chaining because collisions just add the element to the end of the linked list which is done in constant time. Even if the load factor is quite high the time to access the item in a linked list is still very quick if the lists themselves are quite short.

For the hash table that uses quadratic probing the performance will drop quite quickly if the load factor goes above 0.5. This is because collisions occur more and more and the problem of clustering becomes much more apparent. When trying to access data the probing method many times will still have to probe and visit many different indexes before it finds the correct data.

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

    A remove method would be implemented by instead of having the data type be of string, we could create nodes that hold a data type of string, and also hold a member variable of data type boolean that states whether the data is considered deleted or not. This way we can know if it is deleted data, but it will still be in place in the array as to not disrupt the clustering that may occur and allow us not to visit data that is placed in sequence.

12. 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?

    As mentioned in the questions above, this could be done by creating nodes that could accept a string or any other type of data so it is generic, but than still has a data type of boolean stating whether the node is deleted or not.

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

    15 hours.