Nickolas Komarnitsky u0717854

04/07/2017

CS 2420 Assignment 10 Analysis

1. Describe the testing you did to verify that your hash tables each worked.

   What were some of the boundary cases you tested? Was your testing exhaustive?

   The testing that I did was to add in multiple items that collide, and then determine that they can be found and the insert completes. The boundary cases were for an item that could not be entered and for a full hash map. It was somewhat extensive, but I could have done more. You can always do more.

2. For all three hash table implementations. Disable the resize method. Set the size to 79. Insert the first 79 strings from the name file.

   What, if any, problems do you run into?

   Is this a fair test (a fair comparison between implementations)? Explain why or why not.

   I didn't run into any problems, all names were inserted into the table. This is a fair comparison because it tests the exact same amount of insertions into the table for each implementation.

3. Based on your testing, comment on what you may have learned about the differences between chaining, linear and quadratic probing. Which one is better? Is this true for all cases?

   I have learned that linear and quadratic are basically the same, just a different way of handling collisions, while chaining is drastically different in how it handles everything. Chaining is better for almost everything, as the collisions are minimal and all data ends up in it's hash's location. This is true for almost all cases.

4. Describe how would you go about implementing a remove function for each of your hash table implementations.

   For all of the implementations to delete a key, I would hash the key to remove and go to that index, if the key is there, delete it. If it is not there then for chaining I would return that the key is already not there, for linear and quadratic I would probe the table until I find the key or get back to the hash location.

5. How dependent is each hash table on the size of the array?

   How dependent is each hash table on a good hash function?

   If you had to choose, would you want a larger array or a better hash function? Explain why.

   Linear and quadratic are very dependent on the size of the array and having a good hash function. Chaining is not as dependent on the size of the array, rather it is more dependent on a good hash function. For linear and quadratic I would want a larger array to store as many values as possible. For Chaining I would want a very good hash function so that one spot in the table doesn't fill up with keys and it would have it dispersed between each spot in the table.

6. Give any other thoughts you have on this project. In particular, comment on the time it required, as well as, whether and why it was more or less time than you expected.

   This project was very interesting and fun to write. I particularly enjoyed the password cracking part of it. This project took a little more time than I expected to get it just right.

7. Discuss how effective the brute force and dictionary attacks were on the password cracking problem.

   Discuss the effect of using an array vs. a hash set in the computational speed of these attacks.

   The brute force attack was very effective until around 10 character passwords and then it slowed down immensely. The dictionary attack took the same amount of time regardless of the length, but it didn't always find the password. Using an array list for the brute force vs a hash set for it was immense. It took a very long time to check for hashes in the array list when checking the large hash list.