Brian Burton
04/07/17
CS 2420

Assignment 10 Analysis Document

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

My partner and I did our best to test every method in our hash tables. Although, most of the time we had a hard time knowing how to test each method in our Hash Table classes. The way that we got around this problem is defiantly a way that you would not want to do it in the real world. All we would do is call the method, and then debug it to make sure that the method was working the way that we designed it to. We also through in some asserts just for kicks and giggles, but we felt that it was better than nothing.

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

One of the boundary cases that we tested was adding multiple keys that were the same key value pair. It is a good thing that we did this test as well since our code was not implementing this case correctly. After a making a few changes to our code however, we were able to solve the problem. Another boundary case that we checked was the rap around concept in the Quadratic Hash Table class. This was hard to know if it truly was working correctly, but we did our best debugging skills and feel that our code is efficient enough.

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:

```
      ------------ Chaining Hash Table ------------
  Average collisions: 31  Average Hash Function Time: 0.0
  Average Insertion Time: 0.0  Average Find Time: 0.0
  Percent filled : 0.01  Size of Table  : 79
  Elements       : [[<Peppi Elder 39 -324095288>], [], [<Serina Haynes 30 308581981>,
<Margaretta Wolff 38 -949548165>], [<Josephine Hatfield 49 1376236567>, <Sharalyn
Schreckengost 44 1098505589>], [<#Names Age 670766884>, <Kaylyn Priebe 30 -1519894118>],
[<Leatrice Bonner  34 1100415574>], [<Paulina Mckinnon 45 1475227362>], [<Braeden Bould 43
-1620101434>], [], [<Chrissie Brown 36 387945941>], [<Joyce Warrick 45 -1963291420>],
[<Karaugh Levett   35 -1788490965>, <Gaynor Yeskey 32 238063193>], [], [<Hollie Barrett 48
-701712773>, <Deeann Cowart 33 -942624379>], [<Wat Cressman     32 -937782628>], [<Willis
Neely 35 -1693564885>], [<Bettye Dickinson 30 1794531310>], [<Coleman Mcelroy  39
1155374938>, <Clifford Ritter 46 842366431>], [], [], [], [], [<Darla Paynter 33 -
691696293>], [<Marjory Stainforth 42 -378205808>], [<Sunny Throckmorton 43 455221882>,
<Jerrold Herrold 35 272888460>], [<Avis Wise 40  1011899017>], [<Pheobe Hardy 50
977897576>, <Alexina Beard 31 -1511615888>], [], [<Demelza Knisely  37 376298595>], [<Joy
Bailey 49 933514115>], [], [<Sacheverell Walker 47 -1845311103>], [<Chantel Williams 33
523047577>, <Dorris Huston 46 -56875766>], [<Cayley Mcdonald  31 -1517462843>, <Pierce
Dunlap 40  -352260006>, <Cyan Lowstetter 31 -371642972>], [], [<Harriette Isemann 46
```

```
49442659>], [<Cicely Zeal 50 318996506>, <Humphry Isaman 30 -755819975>], [<Jesse Seidner
35 -1569205398>, <Benedicta Weeks 37 1472393268>], [], [], [<Warren Grant 41 170477774>],
[<Dell Zeal 50 -1405262074>, <Kilie Davis 36 -1402801066>, <Petronel Goodman 42
1233733640>], [<Delice Burney 44 -1068855427>, <Lillie Hynes 34 608719453>], [], [], [<Sue
Leichter 36 1185495770>], [<Erin Staymates 38 -1784521250>], [], [<Ilean Tedrow 45 -
2052322167>], [], [<Dennis Tripp 33 2058831690>, <Coleen Alcocke 37 1389444624>],
[<Penelope Beck 41 -1763732010>, <Cameron Whittier 44 1980457213>], [], [], [<Eugene Day 32
-568548834>, <Marge Fields 48 -181670745>], [], [], [], [<Kristia Trout 43 -244585164>],
[<Alexia Welty 40  -1323533827>], [<Harriet Baum 47 1806094347>, <Eldreda Cowper 39 -
651481855>], [<Christianne Simmons 40  -233663022>, <Lisa Orbell  42 -950704231>, <Wendi
Burnett 34 -2022942554>], [], [<Twila Mary 39 -1229810522>], [<Arn Ehret  41 -351211944>],
[], [], [], [<Lissa Hatcher 32 361933038>], [<Connie Read 31 559386192>], [<Nolene Saline
48 581000435>, <Marcus Ewing 47 1850113552>], [<Mercia Dealtry 49 -718849195>], [],
[<Malina Ratcliff 42 -1103958713>], [<Felix Adams      36 467906883>, <Rosannah Potter 34 -
283560857>], [<Jordyn Kistler 43 -961965063>], [<Reuben Winton    38 -896291021>, <Winthrop
Bennett 37 1768383871>, <Marjorie Jenner 38 1088071816>], [<Jere Christopher 44
2050267643>, <Huey Reed 41 1271501996>], [], ]
  Actual Capacity: 79
----------------------------------------
```

What, if any, problems did you run into:

We noticed that not every index in the hash table array was being filled with elements. Instead, some of the elements had two or three elements while some had none. The Linear Hash Table was of course the exception since it only steps once. This is not a problem however, since the more empty spaces that you have, the less collisions you have. Therefore, we fill that we did not run into any problems in the end.

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

I believe that this is a fair test for a few reasons. First of all, it allows us as the programmer to see what is going on in the computer. This allows us to better analyze our algorithm. The second reason that I believe that this is a fair test is because math works. When we receive the statistics of our Hash Tables, we can reasonably see if the numbers are even a possibility.

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:

What I personally learned about the different Hash Tables that you can implement is that some are easier than others. They all have the same ideas however. Linear and Quadratic probing made more since coding wise for me while chaining took a little more work. As far as which one is better, I believe it depends. That is Professor Jim's favorite answer. If you can almost guarantee that you will not have any duplicate hash values, then Linear Probing would be the way to go in my opinion since there is no need to do extra work trying to make it so there is always an empty spot. On the other hand, chaining would be better in the case where you don't know what kind of data you will be receiving. Sure it takes up more memory but it is a more efficient algorithm in this case. Quadratic can also be a good choice however if you are not wanting to use extra memory.

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

The way that I would go about implementing a remove function for the Linear and Quadratic Hash Tables is I would first find the value or key that is to be removed, then I would replace that key with null. If it was a Chaining Hash Table, then I would find the key or value that is to be removed and then make the parents reference change to the node after the removed node or to null.

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

I believe that all of the Hash Tables are extremely dependent on the size of the array because they would all thrive on a larger array and all suck on a small array. It would not matter which implementation that you are using. If you had a lot of data, you would have a lot of collisions no matter what.

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

I believe that all of the Hash Tables are also extremely dependent on a good hash function. If you don't have a good hash function then you stand the risk of loosing your constant add and find algorithms. All of your keys could be placed in the same spot as the Hash Table making it harder to find and insert at that index. This would also mean a lot of collisions.

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

I would want a larger array because of what I said before. No matter which Hash Table you are using, if you have a lot of data and a small array to fit it in, it makes it extremely difficult and even impossible to have an efficient algorithm.

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:

I thought that this would be an easier assignment that usual but I was completely wrong. When the code seemed to be working, it actually wasn't. My partner and I did a lot more debugging that I ever have with anyone else. Either we didn't understand the concepts good enough or we didn't think through the algorithms enough. Our consequence was how long it took us to complete this assignment, 16 hours.

7.

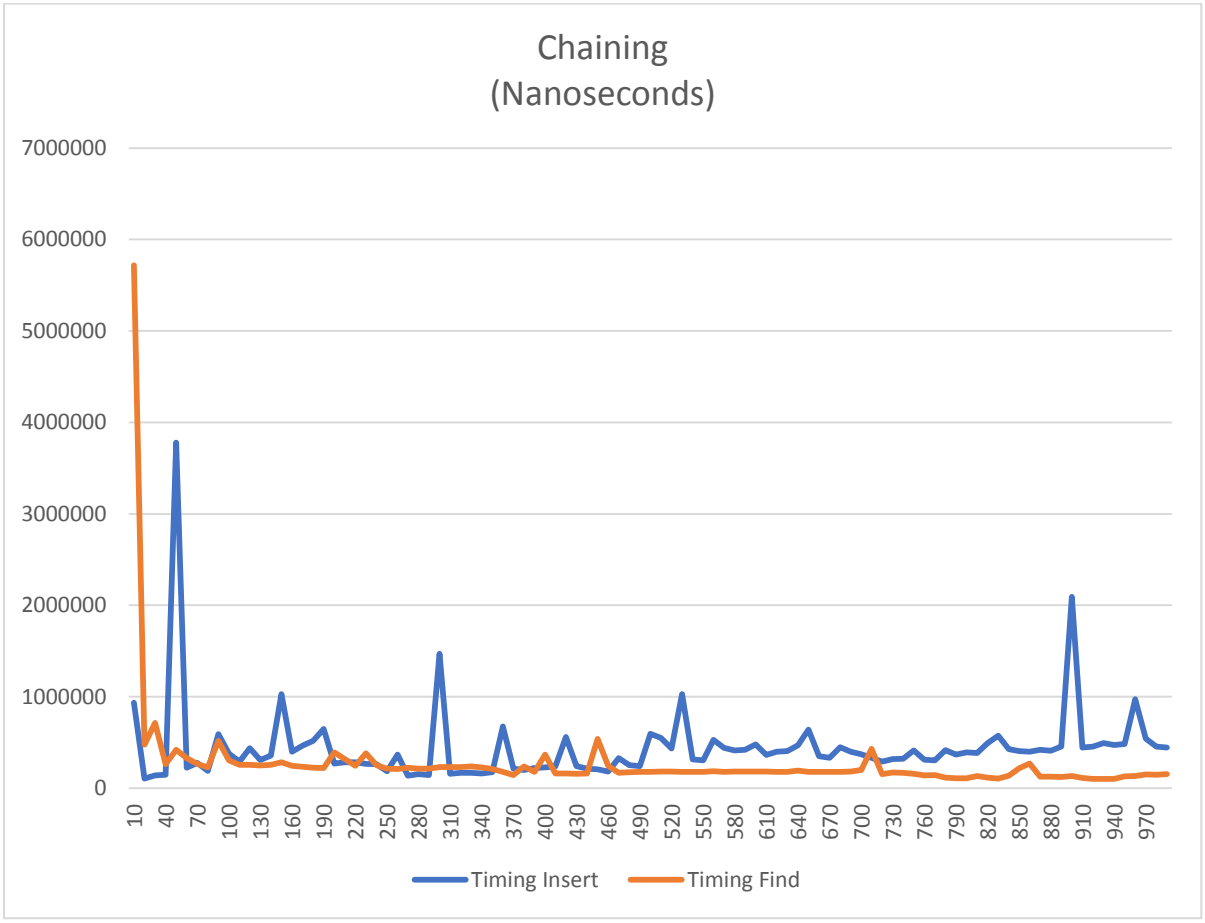
Discuss how effective the brute force and dictionary attacks were on the password cracking problem:

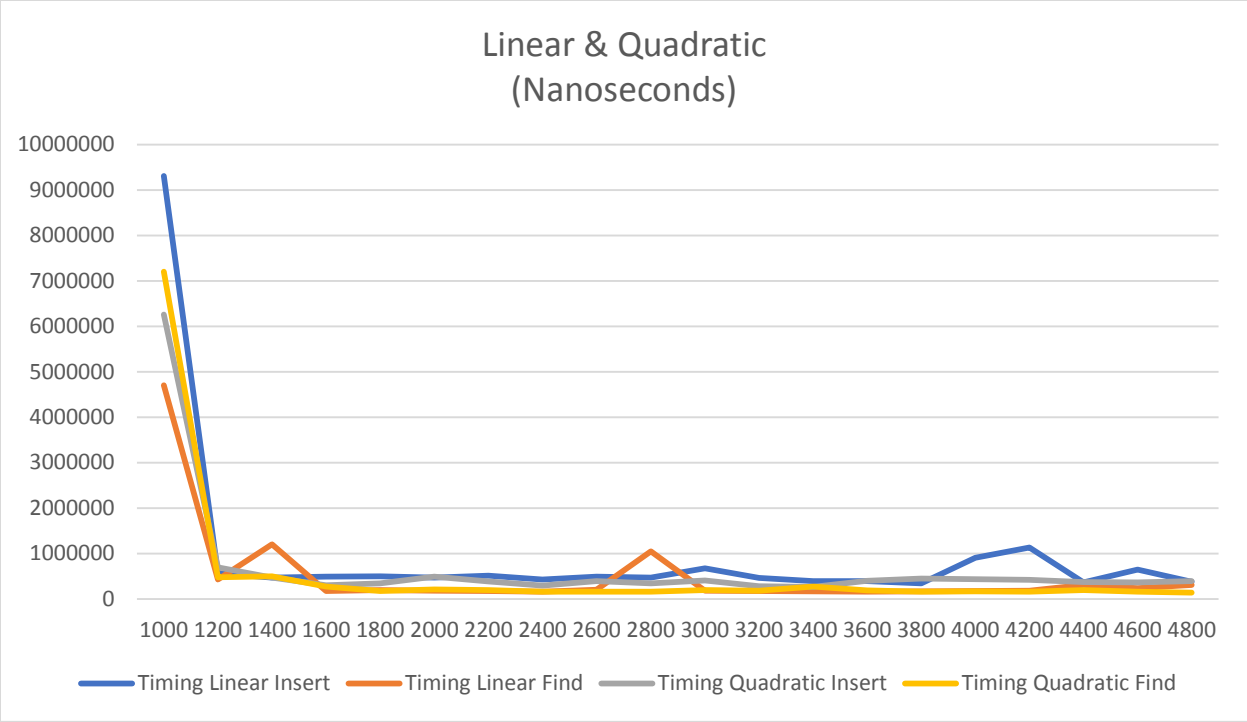
The brute force and dictionary attacks where very effective, it just took way to long to run the code. Once a program takes longer than five seconds max, I and every other user is no longer interested. Especially if it took a few days to calculate. That is how long brute force and dictionary attacks have the potential of taking.

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

Hash sets are the best way that I know of to implement the password cracking program. This is because all of it's operation can be accomplish in constant time no matter how large the table is. This can also be said with arrays, however the downfall with arrays is that its operations is not constant time. And unfortunately, it can be as bad as an N squared algorithm.

Timing Analysis Graph


Chaining
(Nanoseconds)
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
10
40
70
100
130
160
190
220
250
280
310
340
370
400
430
460
490
520
550
580
610
640
670
700
730
760
790
820
850
880
910
940
970
Timing Insert
Timing Find


Linear & Quadratic
(Nanoseconds)
10000000
9000000
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
1000 1200 1400 1600 1800 2000 2200 2400 2600 2800 3000 3200 3400 3600 3800 4000 4200 4400 4600 4800
Timing Linear Insert
Timing Linear Find
Timing Quadratic Insert
Timing Quadratic Find