CSCE 221 Cover Page

Programming Assignment #5

Bonus Due Date: November 27th, 11:59pm

Final Due Date: December 2nd, 11:59pm

First Name: Ian Last Name: Matson UIN: 927003542

**Any assignment turned in without a fully completed cover page will receive ZERO POINTS.**

Please list all below all sources (people, books, webpages, etc) consulted regarding this assignment:

CSCE 221 Students Other People Printed Material Web Material (URL) Other

1. Alex Labbane 1. 1. 1. 1.

2. 2. 2. 2. 2.

3. 3. 3. 3. 3.

4. 4. 4. 4. 4.

5. 5. 5. 5. 5.

Recall that University Regulations, Section 42, define scholastic dishonesty to include acquiring answers from any unauthorized source, working with another person when not specifically permitted, observing the work of other students during any exam, providing answers when not specifically authorized to do so, informing any person of the contents of an exam prior to the exam, and failing to credit sources used. Disciplinary actions range from grade penalties to expulsion. Please consult the Aggie Honor System Office for additional information regarding academic misconduct – it is your responsibility to understand what constitutes academic misconduct and to ensure that you do not commit it.

I certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received nor given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment.

Today’s Date: 11/27/2019

Printed Name (in lieu of a signature):

Ian Matson

**PA5: Hash Tables**

**Ian Matson**

**CSCE 221 510**

**1 Theoretical Statement**

**1.1 Single Operation Time Complexities**

**Insertion:**

Linear Probing: *O(1)*

Double Hashing: *O(1)*

Chaining: *O(1)*

**Removal:**

Linear Probing: *O(N)*

Double Hashing: *O(N)*

Chaining: *O(N)*

**1.2 N-Operations Time Complexities**

**Insertion:**

Linear Probing: *O(n)*

Double Hashing: *O(n)*

Chaining: *O(n)*

**Removal:**

Linear Probing: *O(N\*n)*

Double Hashing: *O(N\*n)*

Chaining: *O(N\*n)*

**2 Experimental Analysis**

**Figure 1**

**Number of Elements Inserted vs. Time for Various Hash Table Collision Resolution Strategies**

**Linear Scale**

**Figure 2**

**Number of Elements Inserted vs. Time for Various Hash Table Collision Resolution Strategies**

**Logarithmic Scale**

**2.1 Discussion of Graphs**

As demonstrated in figures 1 and 2, all three collision resolution techniques performed extremely similarly. The most efficient implementation was linear probing, which was somewhat surprising due to the fact this it is also the most rudimentary. Second was separate chaining, and last was double hashing. Because all three were so concentrated in performance, however, it is likely that even small changes in how each strategy was implemented could have a significant change in the outcome of the experiment. Consequently, it is unrealistic to draw widespread conclusions from this experiment, such as “linear probing is the fastest collision resolution method.”

**3 Discussion**

Although all experimental runtimes performed according to their established theoretical runtimes, a few surprising features can be observed. Firstly, linear probing was the fastest when inserting items. Although at first, this may seem counter-intuitive, after the realization that a simple addition is more computationally efficient than nearly all other operations, the results begin to make relative sense.

Interestingly enough, for small values of n, the relative efficiency for each implementation is reversed. Double hashing becomes the most efficient, chaining second, and linear probing third. This is largely due to the double hashing function becoming slightly less efficient after 10000 items. This is likely due to my implementation of the double hashing function and not reflective of double hashing as a whole, as there numerous established double hashing implementations.