Time Analysis of Open and Closed Hashing Functions

Jan-Michael Carrington
College of Charleston ‘18
12 Bogard Street
Charleston, SC 29403
(864) 558- 5247
carringtonjo@g.cofc.edu

ABSTRACT
In this paper, I provide research of the time efficiencies of open and closed hashing functions.

General Terms
Data Structures, Open Hashing, Closed Hashing, Time Efficiency, Performance, Experimentation.

Keywords
Insert, Search, Delete,

1. INTRODUCTION
I have developed my own versions of the data structures open hashing and closed hashing. My open hashing data structure utilizes a previously developed Array List, which contains Singly Linked Lists, also previously developed, within it. The closed hashing data structure utilizes a previously developed Array List. Both the open and closed hashing data structures have the methods insert, search, and delete which will be empirically tested to see which hashing data structure is the better choice. The hashing function I am utilizing was found from a user named jonathanasdf on the website Stackoverflow.com. I slightly modify the hash function to better suit my open and closed hashing data structures.

2. Methods
Each open and closed hashing data structure utilizes the methods insert, search, and delete. In my open hashing, insert will take in a string and put it through the hash function. The hash value received from the function determines where it will be in the Array List. If a value is already at the hash location, the new value will be placed next in line in the Singly Linked List within the Array List cell. If the value is a duplicate, nothing will occur, as there is already the value in the data structure.

The insert method in my closed hashing data structure is relatively similar. If no value is found at the hash key’s corresponding cell, the string is placed. If a duplicate is found, nothing happens. If a value is already there, linear probing occurs until an empty cell is reached.

The search function in the open hashing data structure runs the taken in string through the hash function. Going to the corresponding Array List cell of the hash value retrieved, if there is nothing there, false is returned. If the key is found, it returns true. If a different value is found, the Singly Linked List is traversed until it either finds the value and returns true, or does not and returns false.

For the closed hashing data structure, the search function does the same for finding the hash key. At the corresponding cell, if nothing is there it returns false. If the value is found it returns true. If a different value is found, it linear probes until finding the value, returning true, or finding an empty cell, returning false.

The final method is delete. In my open hashing data structure’s delete function, it calculates the key value of the given string and goes to the Array List’s corresponding location. If nothing is there, the method does nothing. If the value is found, it is deleted. If a different value is found, the Singly Linked List is traversed until either the value is found, and deleted, or not found, and does nothing.

Similarly in the closed hashing function, the key value is calculated and the corresponding Array List cell is found. If nothing is there, the method does nothing. If the value is found, a lazy deletion occurs replacing the string with the integer -1000. If the string is not found, linear probing occurs until an empty cell is found, where the method does nothing, or the string is found, performing another lazy deletion.

3. Results
3.1 Insertion Results
Experimentation with the insertion method shows that the open hashing data structure is much faster than the closed hashing data structure. Empirical analysis shows that closed hashing can have load times of around six times that of open hashing. On average closed hashing on the large text file asp5202.txt took about 2000 milliseconds while open hashing only took about 350 milliseconds.

3.2 Search Results
The time analysis of the search function gave back almost identical results. Searching for a few words, the same words in each search test, almost always took 0 milliseconds for closed hashing and always took 0 milliseconds for open hashing. The search function is extremely efficient with closed hashing only fractionally slower.

3.3 Deletion Results
Time analysis of the delete function was just like the search function. Closed hashing almost always took 0 milliseconds and open hashing was always 0 milliseconds. The deletion function is extremely efficient with closed hashing only fractionally slower.

4. Conclusion
After performing all the experiments for insert, search, and delete, I have come to the conclusion that open hashing is a better data structure to use. A text input file of small size shows that both data structures will be almost identical in efficiency, but when large files are inputted, the open hashing data structure is much more time efficient at insertion. When it comes to search and delete, both data structures are almost equal with open hashing just slightly better.

5. REFERENCES

[1] Jonathanasdf, Good Hash Functions for Strings. *Stack OverFlow. Lang.* (Apr. 2010), http://stackoverflow.com/questions/2624192/good-hash-function-for-strings.