Final Project Report

Kofi Boampong Benefo – Bandoh -51562022

Papa Kwame Twumasi-Ntiamoah – 37352022

Ashesi University

**ABSTRACT**This is a project about the process of hashing. The project required the use of the python programing language to implement a hashing algorithm and also investigate the efficiency of the algorithms.

Two methods were used to resolve collision on the hash tables, namely the Direct Chaining method and the Tree Overflow method. Both methods were compared to find their runtimes and as such determine which is more efficient in handling this problem.

**PROBLEM DESCRIPTION**

This project seeks to test the concept of hashing, binary trees, array lists and also Big-O notation. The problem revolves around implementing a hashing scheme by generating uniform random numbers in the interval of 16384 to 65535. The numbers generated are then to be inserted in to a hash- table of size 509 which is represented by the variable “M”. The hashing scheme is to be represented with two collision resolution methods. Namely the Simple Direct Chain method and the Tree Organized Overflows. The average times for insertion as the keys go from 1 to 8192.

The insertion would be done in sets of 1024. After the first 1024, the average insertion time is calculated and reinitialized to 0 and the while process is repeated in increasing sets of 1024 till 8192 insertions are made. The average insertion times is computed during this process. The process should be performed on both hashing algorithms.

The results of the average time computations are then plotted on graph using MatPlotLib.

**HASHING ALGORITHM**

The hashing algorithm is one that uses mathematics to encrypt data in a hash function. According to blog.jscrambler.com, a hashing algorithm is a mathematical algorithm that maps data of arbitrary size to a hash of a fixed size. The hashing algorithm consists of a hash table. The hash table was implemented using a linked list of size M = 509. Modulo is used in calculating where each key goes on the list. This helps to realize the hash function, which is a mapping between an item and the slot where it belongs.

For this project mid-square hashing was used for the hashing process. In mid-square hashing, a seed value is selected and then squared, two numbers of the result are selected to be to be the new seed.

Due to the fact that various locations are selected for the placements of the keys it is possible that an occupied index is selected to place a key, when this happens this is referred to as a collision. And this can be resolved by some various methods.

**COLLISION RESOLUTION METHODS**

**Direct chaining**

The direct chaining method was one of the methods used in resolving collision in this project. In direct chaining method, the hash table is an array of linked lists, this means that every index in the list has a linked list and as such when more than one item is hashed to the same index, it forms a link with the item already in that position. This implementation was used in the development of our program.

“LinkedList” was imported from the file linkedList.py to help facilitate the implementation of the direct chaining method. The method run faster in the early testing stages of the program as compared the tree overflows method of resolving the collision.

Tree Overflow

This method uses a binary tree to resolve collision in the hash table. In our implementation we used a recursive insert function to insert the data into the binary tree. The items in the tree were then printed also with a recursive print function. The recursive insert and print functions were used in this function to make it easier to especially insert into the tree and print respectively.

DESCRIPTION OF EXPERIMENT ENVIRONMENT

The experiment was conducted using the Python programming language. The program was tested and run on two local computers which both have at least 8GB of RAM. As a result, the testing of the program was not a problem since there was enough memory to run the programs.

Github was used in the of the project to store the code and protect it from possible damage.

DISCUSSION OF RESULTS

During the early stages of running and testing the program, the simple chaining had faster run times than that of the tree overflows. We expected that that would remain the same, but eventually the Tree Overflow method had a faster insertion time than Direct chaining.

Looking at the graph it is seen that in the early stages the time difference between the two resolution methods is large but as the number of iterations of the program increases, the times seem to converge almost becoming equal.

A Graph of Average Time VS Number of Runs 


Description automatically generated

**CONCLUSION**

In conclusion, we found that, the tree overflow method is more efficient in terms of runtime and also implementation. After 10 iterations of the whole program without errors we are confident that the conclusion of the report is valid and can be used for further research.

References

Hashing Algorithms | Jscrambler Blog. (2020). Retrieved 8 August 2020, from

https://blog.jscrambler.com/hashing-algorithms#:~:text=A%20hashing%20algorithm%20is%20a,hashing%20algorithms%20have%20been%20compromised.

What is chaining in hash tables?. (2020). Retrieved 8 August 2020, from

https://www.educative.io/edpresso/what-is-chaining-in-hash-tables