Comparing Linear, Quadratic and Chaining Probing.

Author: Bilal Hasan Aslan

02.03.2019

Bilalhasanaslan@gmail.com

CONTENTS PAGE

Bilalhasanaslan@gmail.com

«EXPERIMENT»

Aim

Aim of this project is to test efficiency of searching and inserting when we using hashing with linear, quadratic and chaining.

Method

If testing is not chosen we are going to ask which hashing is wanted to use, table size, which file to read also number of keys wanted to search. Then we will insert all data from csv file to asked table. While inserting to given table size we will gather number of probes are made to see their effectiveness for inserting. Then we going to search keys if it is asked from user, through stored data in table and gather number of total probes, average of probes and key that has maximum probe, to see effectiveness for searching.

If testing is chosen we are going to ask 5 table sizes, which file to read also number of keys wanted to search. Then we will insert all data from csv file to all different tables. While inserting to given table sizes we will gather number of probes are made to see their effectiveness for inserting. Then we going to search keys for each table, through stored data in tables and gather number of total probes, average of probes and key that has maximum probe for each, to see effectiveness for searching.

So we need Insert, SearchKeys, and testing methods for all hashing tables.

Classes

I created five different java classes where each has unique function for different hash tables:

1. **hashTable** – This java class is main class that connects all different java classes, this class is used to see effectiveness for a certain table for certain table size or for testing all tables for various number of table sizes.
   Main() : calls **ask()** method to get input what is wanted. after that it does according to what is asked and calls needed classes.
   Ask() : Asks if user wants testing or not then asks table size and other stuff.

2. **linearProbing** – This java class is for linear hashing table that inserts given data to given table size while counting probing also searches given number of keys in stored data again while counting number of probes are made.

insert (String file, int Size): reads file with the given name, creates table with the given size and inserts data with linear hashing. Also counts number of probes made.

Search (String key): Searches through table with hashing for given key. Also counts numbers of probes are made.

searchKeys(int nKeys, ArrayList <String> allData): Searches with hashing, the number of asked keys that is given with list, through stored data.

3. **quadraticProbing** – This java class is for quadratic hashing table that inserts given data to given table size while counting probing also searches given number of keys in stored data again while counting number of probes are made.

   insert (String file, int Size**):** reads file with the given name, creates table with the given size and inserts data with quadratic hashing. Also counts number of probes made.

   Search (String key): Searches through table with hashing for given key. Also counts numbers of probes are made.

   searchKeys(int nKeys, ArrayList <String> allData): Searches with hashing, the number of asked keys that is given with list, through stored data.

4. **chainingProbing** – This java class is with linkedList table that inserts given data to given table size's linkedLists while counting probing also searches given number of keys in stored data again while counting number of probes are made.

   chainingProbing(String file,int tSize): this is a constructor reads file with the given name, creates table with the given size and inserts data to linkedList. Also counts number of probes made.

   Search (String key): Searches through table's LinkedLists for given key. Also counts numbers of probes are made.

   searchKeys(int nKeys, ArrayList <String> allData): Searches linkedLists inside table, the number of asked keys that is given with list, through stored data.

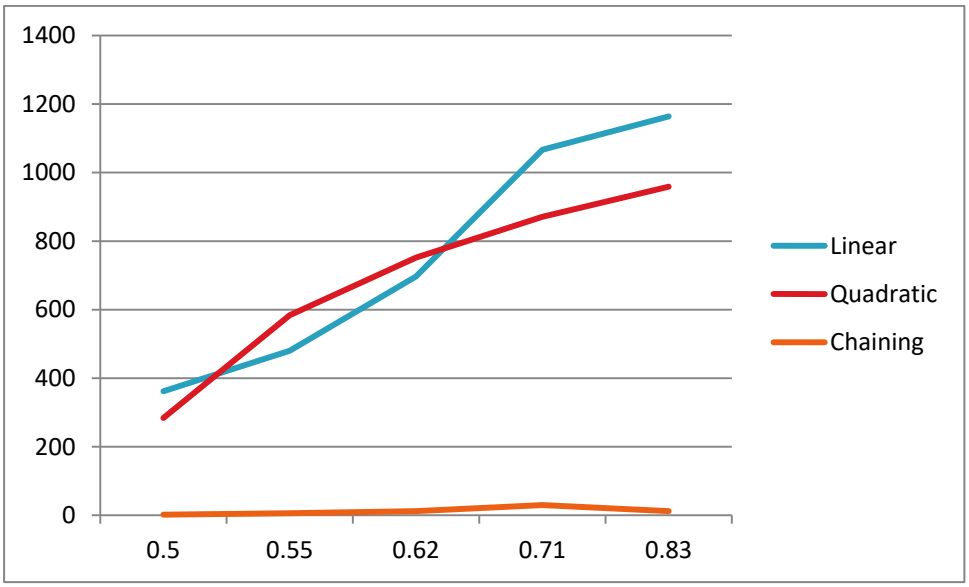
5. **Node** – This java class is a linkedList class.

   printList(): prints everything stored in linkedList.

Bilalhasanaslan@gmail.com

« GRAPHS FOR INSERTION »

Linear, quadratic and chaining total insertion probes compare to load factor



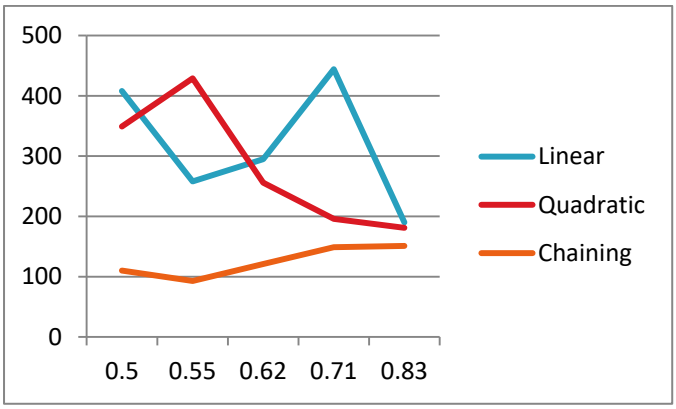
Discussion of results

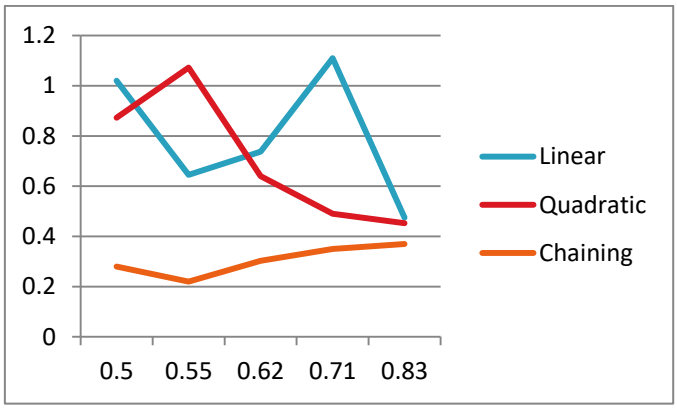
I inserted data from csv file to all structures separately and printed their probe count with load factor. When I compare chaining to linear and quadratic, chaining has much less probing which makes it more efficient over others. When I compare linear and quadratic they don't have much difference on efficiency, but in long run, quadratic seems more efficient over linear. If I compare their complexity for insertion coding while linear is very basic coding, chaining has most complexity.

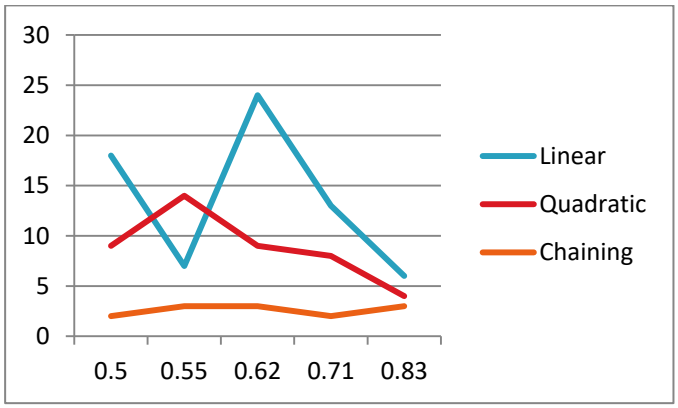
Bilalhasanaslan@gmail.com

«Graphs for Searching»

Graphs of total searching probes, searching probes average and maximum probe made







Discussion of results

The data is inserted earlier from csv file. I searched 400 keys that is randomly picked but searched same key for all tables then printed their probe count of total search count, average search probe count and key with maximum probe. When I compare chaining to linear and quadratic, chaining has less probing which makes it more efficient over others. When I compare linear and quadratic they don't have much difference on efficiency but, in long run, quadratic seems more efficient over linear. If I compare their complexity coding for searching while linear is very basic coding, chaining has most complexity.

Bilalhasanaslan@gmail.com

«CONCLUSION»

Results

As we can see from graphs, inserting and searching with chaining is more efficient when it is compared to inserting and searching in linear and quadratic hashing.

When it comes to very large data sets quadratic hashing becomes better than linear hashing whilst chaining is still much more efficient.

Coding for chaining is more complex compare over linear and quadratic chaining.

Extras (Creativity)

Code is made in a way that is asks users if they want to test or not also brings many options where user can choose to use.

Testing option is made for checking insert and search efficiency to get efficiency information to create a graph.

The way testing method works is even list of keys are shuffled it uses same keys when searching so test is fair for every structure.

Method made in Node class that prints out everything that is stored in linkedList.

«REFERENCES»