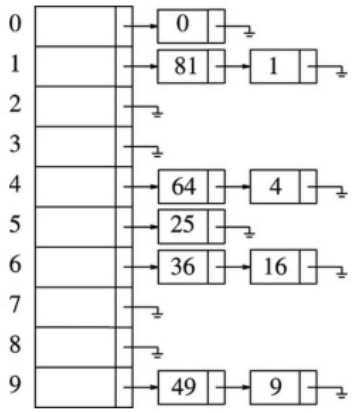
Homework 4

Mehmet Çağrı Aksoy ID: 12105037560

January 6, 2019

Abstract

In this project, required to realize a hash table using chaining as seen in figure. This hash structure will be used to trace a text file. The program will get a path of a text file written in English. The program will trace each word and keep track the positions of them using the hash tables and their related links. After the table is organized, the program ask a user a word and this word is searched through the hash table. If it is found in the text, the program inform user it's first place in the text.



Given table

1 Environment

In this project, Netbeans IDE 8.2 is used, in Ubuntu 18.04 x64 operation system. OverLeaf Online Latex editor is used to design and write this report.

2 Introduction and Algorithm

The algorithm is based on hashtable data structure and linkedlist implementation. Due to requirements of homework, some functions and interface is used. In algorithm side; The algorithm first requests a file path and size of buckets in hash table from the user. The file is scanned from the received file path with the "Scanner" function and the reading process is started. After, those values are sent to BuildHash() function, that split words and erase

some punctuations. Then, it adds them to hash table, during my work I have used two different hash table, one of them is used for save elements to text file and other one is used for search and sort purposes. One of hash table is standard hashtable function which calls with;

```
Hashtable dictionary = new Hashtable();
```

On the other hand, other hash table which is in HashtableLList is created by me and it uses linked list structure to avoid collusions.

```
HashTableLList hll = new HashTableLList(size);
```

Since a hash function gets us a small number for a key which is a big integer or string, there is possibility that two keys result in same value. The situation where a newly inserted key maps to an already occupied slot in hash table is called collision and must be handled using some collision handling technique.

Why am I implemented own hash table with Separate Chaining?

If we want to create hash table in java, we have some ways to do that. Chaining, Separate Chaining and open addressing are ways to create hash table and obtain collisions. In this project, I have used Seperate chaining because easy to use, The idea is to make each cell of hash table point to a linked list of records that have same hash function value. In chaining, hash table never fills up, we can always add more elements to chain, less sensitive to the hash function or load factors, It is mostly used when it is unknown how many and how frequently keys may be inserted or deleted etc.

Inside of my cHash() class; I have used some important functions. As can be seen on below, convertInt functions convert string to long to my string. In FindHash function, it uses output of ConvertInt function to create suitable location for string inside of hash table. InsertHash function is inserting elements into the hashtable. According to the request of the assignment, "inserthash" function involves the other two functions.

```
@Override
   public long ConvertInt(String mystring) {
       return (long) mystring.hashCode();
   }
   @Override
   public int FindHash(long myvalue) {
      myvalue %=  size_table;
      if (myvalue < 0)
           myvalue += size_table;
      return (int)myvalue;
   }
   @Override
   public void InsertHash(String mystring) {
       HashTableLList hll = new HashTableLList(size);
       int tmp = FindHash(ConvertInt(mystring));
               hll.add(mystring, tmp);
   }
```

ShowMax () function is used, to find the most repetitive word in the list. This function compares the values of the words in the hash table. I've defined a table[] array with linkedlist for this. This holds all values and data. Also, this makes it easy to search linearly with a for loop with a complexity like O(n). In the linear search, the ShowMax() function returns the most repetitive word to the screen. Codeblock of ShowMax function can be seen on below:

```
for (int i = 0; i < size ; i++){
            SLLNode start = table[i];
            while(start.next != null){
                if(start.val > start.next.val){
                    max=start.data;
                }else{
                    max=start.next.data;
                }
                start = start.next;
            }
```

Also, the linkedlist design and inserting element to hash table can be seen below. To add elements to a table, we first need to know the hash value that it represents. After learning this, we can add the element to the table and perform the listing tasks.Another task that Homework 4 wants is to save the hash table as a text document. The algorithm that I wrote fulfills this task and saves it in the "output.txt" document. This document can be found in the directory to which the project is linked.Display function that save all whole table as text file:

```
    public void Display(String Outputfile) {
        try (PrintStream out = new PrintStream(new FileOutputStream(Outputfile))){

            for(Object key: dictionary.keySet()){

                out.println(key + ": " + dictionary.get(key));
                //System.out.println(key + ": " + dictionary.get(key));
            }
        }catch (Exception e){
        }

public void insert(String key, int val){
        int pos = myhash(key);

        nptr = new SLLNode(key,val);
        if (table[pos] == null)
            table[pos] = nptr;
        else{
            nptr.next = table[pos];
            table[pos] = nptr;
            }
```

3 Conclusions

As much as creating an algorithm, transferring the algorithm to the code is at least as important as the algorithm itself. To integrate the written algorithm into the code block, java properties such as linkedlist, and hash table are used. Code block consists of 4 different files. One is "interface" and the others are classes whose is named "HW4, cHash, HashTableLList". The most important point in designing the algorithm is complexity. Two complexity types exist, time and space. During my work, I wrote the program taking into account the time complexity. For, seperate chaining, single linked list structure is used so, its complexity is O(n) for access and search operations. Other operations, such as, insertion and deletion have O(n) complexity. Also, in the worst case, hashtables have O (n) complexity for all operations. As you see, to add some elements and wants to to avoid collision, we have O(2n) which is approximately O(n) complexity. Due to tasks for assignment, program need to find max repeated element that is in hash table. So, it's complexity is much more than other operations. Finding the biggest value in hashtable, cost me O (n^2) complexity.

4 References

http://www.algolist.net/Data$_s$*tructures*/$Hash_t$*able*/*Chaining*

http://courses.csail.mit.edu/6.006/fall09/lecture$_n$*otes*/*lecture*05.*pdf*

*h*ttps://www.geeksforgeeks.org/implementing-our-own-hash-table-with-separate-chaining-in-java/

https://www.cs.cmu.edu/ adamchik/15-121/lectures/Hashing/hashing.html