GTU Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework #5 Report

Part-2

Ömer Faruk Bitikçioğlu
161044010

1. PROBLEM SOLUTION APPROACH

- *For the first part,* I completely copied the HashtableChain class from the book and refined some code of it (the remove method etc.)

- *For the second part* I just changed the LinkedList parts with the TreeSet. Then I changed the K value to extend Comparable interface. Because the implementation of TreeSet uses the comparison when inserting new elements to the tree. I also overridden the equals method of Object class for this purpose.

- *For the third part* I copied the HashtableOpen from the book and changed its linear probing to quadratic probing first. I added nextIndex field to the Entry class that holds the next colliding entry.

  I separated the find method with find and isPresent methods. isPresent() method checks if the given key is present in the HashMap or not. It uses the next index references in a loop. If a key is available in the table, then it must be in the index or the next indices. If not, then the key is not present.

  Find method first checks that if the given key is present in the table or not. If not, then it looks for an appropriate null place by using quadratic probing. While doing so, it bounds the colliding entries together with the nextIndex references.

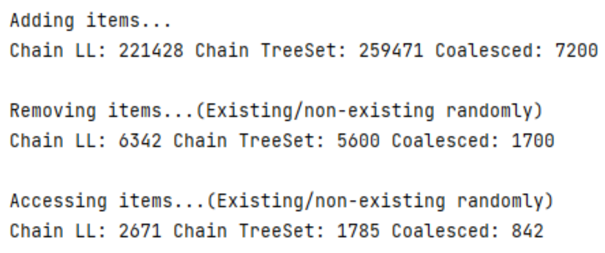
  Get() method uses the isPresent method from now on, since the entry may not be present and in a such scenario nextIndex references crashes. So, I used isPresent method to avoid this problem.

  Remove method first checks if the given key is present in the table, then it removes from the table. If the removed entry has next elements, it shifts the following entries to back in a loop. Since we no longer use a field like DELETED, we do not count the deleted items and we do not check them when rehashing.

2. TEST CASES

1. Test with 7 entries and boundary of 100
2. Test with 7 entries and boundary of 10000
3. Test with 17 entries and boundary of 100
4. Test with 17 entries and boundary of 10000
5. Test with 29 entries and boundary of 100
6. Test with 29 entries and boundary of 10000

3. RUNNING AND RESULTS

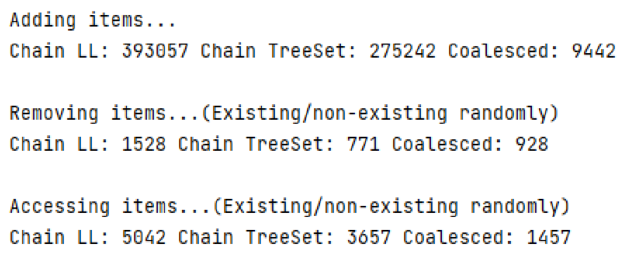


```
Adding items...
Chain LL: 221428 Chain TreeSet: 259471 Coalesced: 7200

Removing items...(Existing/non-existing randomly)
Chain LL: 6342 Chain TreeSet: 5600 Coalesced: 1700

Accessing items...(Existing/non-existing randomly)
Chain LL: 2671 Chain TreeSet: 1785 Coalesced: 842
```

1.

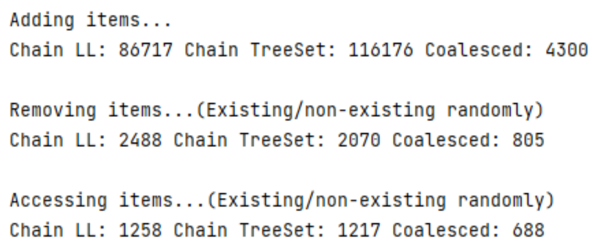


```
Adding items...
Chain LL: 393057 Chain TreeSet: 275242 Coalesced: 9442

Removing items...(Existing/non-existing randomly)
Chain LL: 1528 Chain TreeSet: 771 Coalesced: 928

Accessing items...(Existing/non-existing randomly)
Chain LL: 5042 Chain TreeSet: 3657 Coalesced: 1457
```

2.

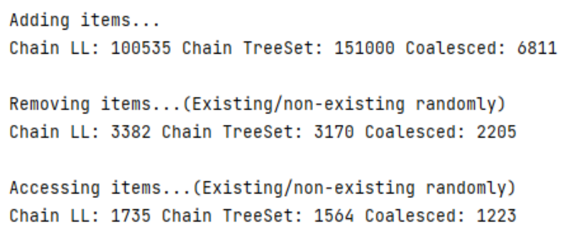


```
Adding items...
Chain LL: 86717 Chain TreeSet: 116176 Coalesced: 4300

Removing items...(Existing/non-existing randomly)
Chain LL: 2488 Chain TreeSet: 2070 Coalesced: 805

Accessing items...(Existing/non-existing randomly)
Chain LL: 1258 Chain TreeSet: 1217 Coalesced: 688
```

3.

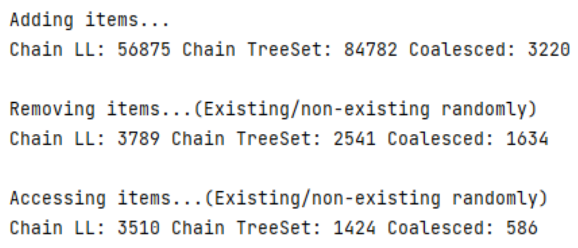


```
Adding items...
Chain LL: 100535 Chain TreeSet: 151000 Coalesced: 6811

Removing items...(Existing/non-existing randomly)
Chain LL: 3382 Chain TreeSet: 3170 Coalesced: 2205

Accessing items...(Existing/non-existing randomly)
Chain LL: 1735 Chain TreeSet: 1564 Coalesced: 1223
```

4.

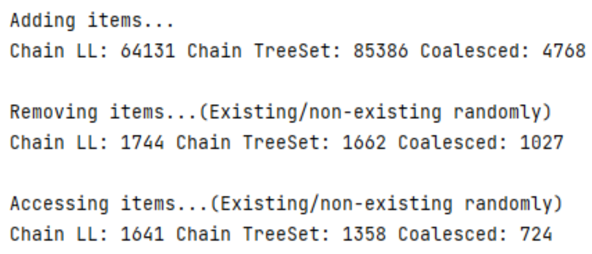


```
Adding items...
Chain LL: 56875 Chain TreeSet: 84782 Coalesced: 3220

Removing items...(Existing/non-existing randomly)
Chain LL: 3789 Chain TreeSet: 2541 Coalesced: 1634

Accessing items...(Existing/non-existing randomly)
Chain LL: 3510 Chain TreeSet: 1424 Coalesced: 586
```

5.



```
Adding items...
Chain LL: 64131 Chain TreeSet: 85386 Coalesced: 4768

Removing items...(Existing/non-existing randomly)
Chain LL: 1744 Chain TreeSet: 1662 Coalesced: 1027

Accessing items...(Existing/non-existing randomly)
Chain LL: 1641 Chain TreeSet: 1358 Coalesced: 724
```

6.