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

Burak Yıldırım
1901042609

1. PROBLEM SOLUTION APPROACH

   Part1:

   I created HashMapIterator interface and implemented it in MapIterator class.

   Part2:

   I implemented Hashtable with chaining and open addressing techniques. Chaining techniques are using LinkedList and TreeSet to store convergent elements and open addressing technique is using Coalesced hashing.

2. TEST CASES

   Part1:

   I put 5 elements in the hash map and firstly with zero parameter constructer I iterated the keys using next() and prev() methods, then with parameterized constructor I iterated the keys using next and prev() methods.

   Part2:

   Small = 10, Medium = 100, Large = 1000

   For small/medium/large number of elements, I repeated the following process:

   I added small/medium/large number of elements to the hashtables and compared the time passed for each hashtable.

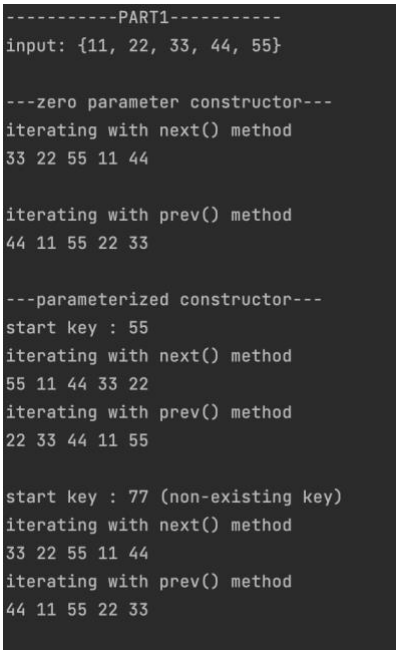
   I accessed half of the elements in the hashtables and compared the time passed for each hashtable.

   I accessed non-existing element in the hashtables and compared the time passed for each hashtable.

   I removed half of the elements in the hashtables and compared the time passed for each hashtable.

   I removed non-existing elements in the hashtables and compared the time passed for each hashtable.

3. RUNNING AND RESULTS



```
----------PART2----------
-----small-sized data-----
adding(upper bound is 5000)
number of data added: 10
HashtableLinkedList performance: 475070 nanoseconds
HashtableTreeSet performance: 720247 nanoseconds
HashtableCoalesced performance: 62262 nanoseconds

accessing existing elements(first 5 elements)
HashtableLinkedList
accessed elements: 2535 4938 2410 4363 1215
performance: 33013 nanoseconds

HashtableTreeSet
accessed elements: 2535 4938 2410 4363 1215
performance: 39046 nanoseconds

HashtableCoalesced
accessed elements: 2535 4938 2410 4363 1215
performance: 13360 nanoseconds
```

```
accessing non-existing elements(5001 to 5006)
HashtableLinkedList
accessed elements: null null null null null
performance: 23998 nanoseconds

HashtableTreeSet
accessed elements: null null null null null
performance: 11806 nanoseconds

HashtableCoalesced
accessed elements: null null null null null
performance: 6821 nanoseconds
```

```
removing an existing element(first 5 elements)
HashtableLinkedList
removed elements: 2535 4938 2410 4363 1215
performance: 30149 nanoseconds

HashtableTreeSet
removed elements: 2535 4938 2410 4363 1215
performance: 99335 nanoseconds

HashtableCoalesced
removed elements: 2535 4938 2410 4363 1215
performance: 28246 nanoseconds
```

```
removing non-existing elements(5001 to 5006)
HashtableLinkedList
removed elements: null null null null null
performance: 8501 nanoseconds

HashtableTreeSet
removed elements: null null null null null
performance: 8384 nanoseconds

HashtableCoalesced
removed elements: null null null null null
performance: 7223 nanoseconds
```

```
-----medium-sized data-----
adding(upper bound is 5000)
number of data added: 100
HashtableLinkedList performance: 236328 nanoseconds
HashtableTreeSet performance: 357823 nanoseconds
HashtableCoalesced performance: 1136659 nanoseconds

accessing existing elements(first 50 elements)
HashtableLinkedList performance: 94750 nanoseconds

HashtableTreeSet performance: 90471 nanoseconds

HashtableCoalesced performance: 28563 nanoseconds

accessing non-existing elements(5001 to 5051)
HashtableLinkedList performance: 74818 nanoseconds

HashtableTreeSet performance: 55705 nanoseconds

HashtableCoalesced performance: 20852 nanoseconds
```

```
removing existing elements(first 50 elements)
HashtableLinkedList performance: 93150 nanoseconds

HashtableTreeSet performance: 145052 nanoseconds

HashtableCoalesced performance: 54680 nanoseconds

removing non-existing elements(5001 to 5051)
HashtableLinkedList performance: 31526 nanoseconds

HashtableTreeSet performance: 30841 nanoseconds

HashtableCoalesced performance: 31663 nanoseconds
```

```
-----large-sized data-----
adding(upper bound is 5000)
number of data added: 1000
HashtableLinkedList performance: 2319159 nanoseconds
HashtableTreeSet performance: 3822743 nanoseconds
HashtableCoalesced performance: 576469 nanoseconds

accessing existing elements(first 500 elements)
HashtableLinkedList performance: 184834 nanoseconds

HashtableTreeSet performance: 305049 nanoseconds

HashtableCoalesced performance: 73226 nanoseconds

accessing non-existing elements(5001 + 5501)
HashtableLinkedList performance: 225754 nanoseconds

HashtableTreeSet performance: 240210 nanoseconds

HashtableCoalesced performance: 67712 nanoseconds
```

```
removing existing elements(first 500 elements)
HashtableLinkedList performance: 343404 nanoseconds

HashtableTreeSet performance: 598059 nanoseconds

HashtableCoalesced performance: 359257 nanoseconds

removing non-existing elements(5001 to 5501)
HashtableLinkedList performance: 112408 nanoseconds

HashtableTreeSet performance: 91321 nanoseconds

HashtableCoalesced performance: 143398 nanoseconds
```