GIT Department of Computer Engineering

CSE 222/505 – Spring 2021

Homework 5 Report

Ahmet Tuğkan Ayhan

1901042692

1. Problem Solution Approach

PART 1

All hashtable classes I have writen in this homework implements KWHashMap interface which available in the book. For the MapIterator class, I first implemented KWHashMap interface with HashtableOpen class. Inside of this class I created an inner class with name MapIterator. This also implements Java's original Iterator interface. I also added prev() method additionaly.

hasNext -> It returns true if number of iterations that have been made is lower than the size of the hashtable class.

next -> It behaves like a circular array. and until it finds a non-null element it will iterate. If it finds a non-null element, it will increase number of iteration by 1 and iterate index. (also assigns lastItemReturned)

prev -> Everything is same with next method. But instead of iterating forward this one moves backward. It will decrease currentIteration by 1 if it finds a non-null element. (After element at index 0 it goes last index.)

MapIterator -> uses find(K key) method to find index of given key. After that, changes currentIndex into given index if table[index] is not null.

PART 2

For the first class(HashtableChain), I took methods from the course book. For the second class(HashtableTree), I changed some methods and the entry array which holds key & value parities. In chain technique entries were stored with linkedlist and in treeset technique entries were stroed with treeset data structure. For the last part, I wrote everything from scratch, entries were stored with Nodes(which is an inner class that contains Entry class and next Entry index).

find(K key) -> It returns index of the given key in the Node array. First gets initial index by using key's hashcode and dividing it into table.length. After that if initial index doesn't contain given key, then iterates by using quadratic probing. After it finds the given key, returns it's index in the Node array. (-1 if element doesn't exist)

findParent(K key) -> It returns index of previous item of given key. For example if 3, 13 added into the hashtable, 3 points into index 4 where 13 inserted. And this method returns 3 when given parameter is 13 because 3 points into 13. Method itself first get's index of given key by using find() method. Then starting from initial index(for 3 and 13, it is index 3) it looks for next value. If the key that next index points into is the same with given key, then it returns currNode's index as parent.

get -> First it get's the index of given key by using find method. If given key is not the key, then looks for next positions until current node is null.

put -> Get's initial index of given key by using it hashcode and dividing into table.length. Then, assigns this index into current node and goes for next value until it finds empty space. Also holds a parent node to point into currnode if they are not same.

remove -> first gets index of the key and parent index of the key. If it is the first element with that remaining, and points into no element, then just assign it to the DELETED node. If key doesn't exist(index is -1) give a warning message and return null. If key exist and it has a next value, then assign next node data into curr node. If key doesn't have a next value and has a parent index, remove current node and make sure parent index now points into nothing(-1).

size -> returns numKeys data field which holds total number of keys added to the hashtable

isEmpty -> returns true when numKeys is lower than 1.

2. Test Cases

PART 1

- Initialize HashtableOpen object and insert 6 elements.
- Initialize MapIterator object with no parameter and search through every element going forward and backwards
- Initialize MapIterator object with one parameter and search through every element going forward and backwards

PART 2

- Create 3 arrays with small-medium-large size, insert random values into them(5,100,2000).
- Initialize 3 Hashtable object and insert small random array into them, calculate time collapsed between insertion.
- Try to acces 2 existing element and calculate time collapsed between get operation.
- Try to access 1 non-existing element and calculate time collapsed between get operation. Also check return value.
- Remove 2 elements and show time collapsed between remove operation. Also show sizes before removing and after removing
- Initialize 3 Hashtable object and insert medium random array into them, calculate time collapsed between insertion.
- Try to acces 10 existing element and calculate time collapsed between get operation.
- Try to access 1 non-existing element and calculate time collapsed between get operation. Also check return value.

- Remove 10 elements and show time collapsed between remove operation. Also show sizes before removing and after removing
- Initialize 3 Hashtable object and insert medium random array into them, calculate time collapsed between insertion.
- Try to acces 100 existing element and calculate time collapsed between get operation.
- Try to access 1 non-existing element and calculate time collapsed between get operation. Also check return value.
- Remove 100 elements and show time collapsed between remove operation.

3. Running Command and Results

```
tgknyhn@tgknyhn-VirtualBox:~/Desktop/Homework5$ make
javac src/*.java
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
java -cp ./src Run

************
** PART 1 **
************

Adding elements...

With no parameter constructor
-----------------------------
Elements (next method) : abd, fransa, japonya, almanya, türkiye, ingiltere
Elements (prev method) : ingiltere, türkiye, almanya, japonya, fransa, abd

With one parameter constructor(I choosed "japonya")
---------------------------------------------------
Elements (next method) : japonya, almanya, türkiye, ingiltere, abd, fransa
Elements (prev method) : fransa, abd, ingiltere, türkiye, almanya, japonya


************
** PART 2 **
************

Inserting 5 elements into hashTables
------------------------------------

Inserting...

-- HashtableChain --

1) Key:   810   Value: 800
2) Key:   924   Value: 914
3) Key:   756   Value: 746
4) Key:   593   Value: 583
5) Key:   605   Value: 595

-- HashtableTree --

1) Key:   810   Value: 800
2) Key:   924   Value: 914
3) Key:   756   Value: 746
4) Key:   593   Value: 583
5) Key:   605   Value: 595

-- HashtableCoalesced --
```

```
-- HashtableCoalesced --

        Index    Key    Next    Value
        -----   -----   ----    -----
          0      810      -      800
          1       -       -       -
          2       -       -       -
          3      593      -      583
          4      924      -      914
          5      605      -      595
          6      756      -      746
          7       -       -       -
          8       -       -       -
          9       -       -       -


Time collapsed while inserting 5 elements into HashtableChain object     : 2316604 nanosecond
Time collapsed while inserting 5 elements into HashtableTree object      : 1594688 nanosecond
Time collapsed while inserting 5 elements into HashtableCoalesced object : 22080 nanosecond

Getting 2 elements which exist in hashTables
---------------------------------------------

Getting...

HashtableChain      -> Key: 593 Value: 583
HashtableChain      -> Key: 810 Value: 800

HashtableTree       -> Key: 593 Value: 583
HashtableTree       -> Key: 810 Value: 800

HashtableCoalesced -> Key: 593 Value: 583
HashtableCoalesced -> Key: 810 Value: 800

Time collapsed while getting 2 elements that exist in HashtableChain object     : 34682633 nanosecond
Time collapsed while getting 2 elements that exist in HashtableTree object      : 10564176 nanosecond
Time collapsed while getting 2 elements that exist in HashtableCoalesced object : 3250757 nanosecond

Getting one element which not exist in hashTables
-------------------------------------------------

Getting...

HashtableChain      -> Key: 1099 Value: null

HashtableTree       -> Key: 1099 Value: null

HashtableCoalesced -> Key: 1099 Value: null

Time collapsed while getting one element which not exist in HashtableChain object     : 1230978 nanosecond
Time collapsed while getting one element which not exist in HashtableTree object      : 3360727 nanosecond
Time collapsed while getting one element which not exist in HashtableCoalesced object : 1656125 nanosecond

Removing 2 element that exist in hashTables
-------------------------------------------

Removing... (593, 810)

(HashtableChain)     Size before removing : 5 Size after removing : 3
(HashtableTree)      Size before removing : 5 Size after removing : 3
(HashtableCoalesced) Size before removing : 5 Size after removing : 3

Time collapsed while removing 2 elements that exist in HashtableChain object     : 48499 nanosecond
Time collapsed while removing 2 elements that exist in HashtableTree object      : 55662 nanosecond
Time collapsed while removing 2 elements that exist in HashtableCoalesced object : 12025 nanosecond

Inserting 100 elements into hashTables
--------------------------------------

Inserting...

Time collapsed while inserting 100 elements into HashtableChain object     : 596782 nanosecond
Time collapsed while inserting 100 elements into HashtableTree object       : 1095683 nanosecond
Time collapsed while inserting 100 elements into HashtableCoalesced object : 1009346 nanosecond

Getting 10 elements which exist in hashTables
---------------------------------------------

Getting...


Time collapsed while getting 2 elements that exist in HashtableChain object     : 39674 nanosecond
Time collapsed while getting 2 elements that exist in HashtableTree object      : 35438 nanosecond
Time collapsed while getting 2 elements that exist in HashtableCoalesced object : 25907 nanosecond

Getting one element which not exist in hashTables
-------------------------------------------------

Getting...

HashtableChain      -> Key: 1099 Value: null

HashtableTree       -> Key: 1099 Value: null

HashtableCoalesced -> Key: 1099 Value: null
```

```
Time collapsed while getting one element which not exist in HashtableChain object     : 409476 nanosecond
Time collapsed while getting one element which not exist in HashtableTree object      : 487484 nanosecond
Time collapsed while getting one element which not exist in HashtableCoalesced object : 481981 nanosecond

Removing 10 element that exist in hashTables
-----------------------------------------

Removing...

(HashtableChain)     Size before removing : 97 Size after removing : 87
(HashtableTree)      Size before removing : 97 Size after removing : 87
(HashtableCoalesced) Size before removing : 97 Size after removing : 87

Time collapsed while removing 10 elements that exist in HashtableChain object     : 61799 nanosecond
Time collapsed while removing 10 elements that exist in HashtableTree object      : 97399 nanosecond
Time collapsed while removing 10 elements that exist in HashtableCoalesced object : 90763 nanosecond

Inserting 2000 elements into hashTables
-----------------------------------

Inserting...

Time collapsed while inserting 2000 elements into HashtableChain object     : 11055379 nanosecond
Time collapsed while inserting 2000 elements into HashtableTree object       : 11374892 nanosecond
Time collapsed while inserting 2000 elements into HashtableCoalesced object : 9749244 nanosecond

Getting 100 elements which exist in hashTables
------------*----------------------------------

Getting...


Time collapsed while getting 100 elements that exist in HashtableChain object     : 164096 nanosecond
Time collapsed while getting 100 elements that exist in HashtableTree object      : 123741 nanosecond
Time collapsed while getting 100 elements that exist in HashtableCoalesced object : 102327 nanosecond

Getting one element which not exist in hashTables
-------------------------------------------------
```

```
Getting one element which not exist in hashTables
-------------------------------------------------

Getting...

HashtableChain     -> Key: 1099 Value: null

HashtableTree      -> Key: 1099 Value: null

HashtableCoalesced -> Key: 1099 Value: null

Time collapsed while getting one element which not exist in HashtableChain object     : 380521 nanosecond
Time collapsed while getting one element which not exist in HashtableTree object      : 334845 nanosecond
Time collapsed while getting one element which not exist in HashtableCoalesced object : 359680 nanosecond

Removing 500 element that exist in hashTables
-----------------------------------------------

Removing...

Time collapsed while removing 100 elements that exist in HashtableChain object     : 462568 nanosecond
Time collapsed while removing 100 elements that exist in HashtableTree object      : 430020 nanosecond
Time collapsed while removing 100 elements that exist in HashtableCoalesced object : 667719 nanosecond

Removing one element that doesn't exist in hashTables
-----------------------------------------------------
The item you are trying to remove doesn't exist in the hashmap
tgknyhn@tgknyhn-VirtualBox:~/Desktop/Homework5$
```