CSE 222 HOMEWORK 5 REPORT

Anıl Mert ULAŞAN
171044004

Problem Solution Approach

Hash Table Linked List

I took the implementation from the book. I wrote the missing parts by looking to the algorithm.

Hash Table Tree Set

It is really similar to linked list table. I made my entry class comparable to be operated by TreeSet. My remove operation doesn't work on Tree Set Hash Table.

Hash Table Coalesced

This approach is very similar to open addressing. To hold the next node, i added my entry class an Integer next. It is null until a collision happens and new key tries to be added to same index. When new entry added, this entry holds its index on the table.

When removing an object, I check if the entry going to be removed is anybody's next value. If it is, i replace next value with deleted item, then make my table index of next entry null.

TEST CASES

PART 1

After adding elements to the hashMap;

1) Test Mapiterator(K key) constructor. Try giving a non existing key as parameter. See if it will throw an exception.

2) Test Mapiterator(K key) constructir. Try giving an existing key as parameter. See if it starts from that key.

3) Test hasNext() method. Try to iterate over keys more than the number of keys. See hasNext() methods stops when there is no non-iterated element left.

4) Test next() method. Try to iterate over keys more than the number of keys. See if it starts from index 0 when no non-iterated elements left.

5) Test prev() method. Try to iterate over keys more than the number of keys. See if it goes to the last key when it comes to index -1.

6) Test Mapiterator() constructor. Try to see where it starts to iterating.

PART 2

I tested all three type of tables with small, medium, large values. I only printed values to screen when testing small tables because printing medium and large tables would be unnecessary.

a) Small Table Test(Printed)

1) Insert some entries to the table and print.
2) Remove some entries form table and print, see if it breaks the table.
3) Insert some entries that has same key that i inserted before, print. See if could same key can be inserted again, if not, see if the value of the same key is changed.

The rest of the operations are made for calculating the time it took to do this operations and compare all 9 results. They could'nt be printed.

The tests that mentioned above are made in all of the tests. The other operations that not printed;

4)Try to access non-existing keys
5) Access existing keys
6) Try to remove non-existing keys

b) Time Calculation Test

These all 6 tests are made in all of the tables(Small-Medium-Large). Every type has been tested with those table sizes. Every test returns the time it took. In the end of driver code, all tests results(times) are printed to the screen.

Running Command&Results

PART 1

```
----------MapIterator Test----------


----------Adding some elements to the hashmap(int-int)----------

{0=100, 1=99, 2=98, 3=97, 4=96, 5=95, 6=94, 7=93, 8=92, 9=91, 10=90, 11=89, 12=88, 13=87, 14=86, 15=85, 16=84, 17=83, 18=82, 19=81, 20=80, 21=79, 22=78, 23=77, 24=76}
----------25 key&values are added----------


----------Create the iterator with non-existent key '-1'----------
Exception occured: Key is not found in the hashmap...
```

```
--------------Create the iterator with an existent key '5'---------------
--------------Iterate with next() method 30 times using hasNext()---------------
key->5
key->6
key->7
key->8
key->9
key->10
key->11
key->12
key->13
key->14
key->15
key->16
key->17
key->18
key->19
key->20
key->21
key->22
key->23
key->24
key->0
key->1
key->2
key->3
key->4
```

```
--------------Iterate with next() method 30 times---------------
key->0
key->1
key->2
key->3
key->4
key->5
key->6
key->7
key->8
key->9
key->10
key->11
key->12
key->13
key->14
key->15
key->16
key->17
key->18
key->19
key->20
key->21
key->22
key->23
key->24
key->0
key->1
key->2
```

```
--------------Iterate with prev() method 30 times---------------
key->5
key->4
key->3
key->2
key->1
key->0
key->24
key->23
key->22
key->21
key->20
key->19
key->18
key->17
key->16
key->15
key->14
key->13
key->12
key->11
key->10
key->9
key->8
key->7
key->6
key->5
key->4
key->3
key->2
key->1
```

```
--------------Create the iterator with no parameter---------------
--------------Iterate with next() method 30 times---------------
key->22
key->23
key->24
key->0
key->1
key->2
key->3
key->4
key->5
key->6
key->7
key->8
key->9
key->10
key->11
key->12
key->13
key->14
key->15
key->16
key->17
key->18
key->19
key->20
key->21
key->0
key->1
key->2
key->3
key->4
```

PART 2

```
----------------Inserting some entries that has existent keys to the table--------------------
Key-> 0, Value-> 0
Key-> 2, Value-> 5
Key-> 4, Value-> 10
Key-> 6, Value-> 9
Key-> 8, Value-> 12
Key-> 10, Value-> 15
Key-> 12, Value-> 30
Key-> 14, Value-> 35
Key-> 22, Value-> 55
Key-> 24, Value-> 60
Key-> 26, Value-> 65
Key-> 28, Value-> 70
Key-> 30, Value-> 75
Key-> 32, Value-> 80
Key-> 34, Value-> 85
Key-> 36, Value-> 90
Key-> 38, Value-> 95
```

Resim yazısını buraya girin.

```
--------------Test small Table for ChainLL-----------------



----------------Inserting 20 entries to the table--------------------
Key-> 0, Value-> 0
Key-> 2, Value-> 5
Key-> 4, Value-> 10
Key-> 6, Value-> 15
Key-> 8, Value-> 20
Key-> 10, Value-> 25
Key-> 12, Value-> 30
Key-> 14, Value-> 35
Key-> 16, Value-> 40
Key-> 18, Value-> 45
Key-> 20, Value-> 50
Key-> 22, Value-> 55
Key-> 24, Value-> 60
Key-> 26, Value-> 65
Key-> 28, Value-> 70
Key-> 30, Value-> 75
Key-> 32, Value-> 80
Key-> 34, Value-> 85
Key-> 36, Value-> 90
Key-> 38, Value-> 95
```

```
----------------Removing some entries from the table--------------------
Key-> 0, Value-> 0
Key-> 2, Value-> 5
Key-> 4, Value-> 10
Key-> 6, Value-> 15
Key-> 8, Value-> 20
Key-> 10, Value-> 25
Key-> 12, Value-> 30
Key-> 14, Value-> 35
Key-> 22, Value-> 55
Key-> 24, Value-> 60
Key-> 26, Value-> 65
Key-> 28, Value-> 70
Key-> 30, Value-> 75
Key-> 32, Value-> 80
Key-> 34, Value-> 85
Key-> 36, Value-> 90
Key-> 38, Value-> 95
```

Resim yazısını buraya girin.