GIT Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework 4 Report

Coşkun Hasan ŞALTU
1801042631

1. SYSTEM REQUIREMENTS

Functional Requirments

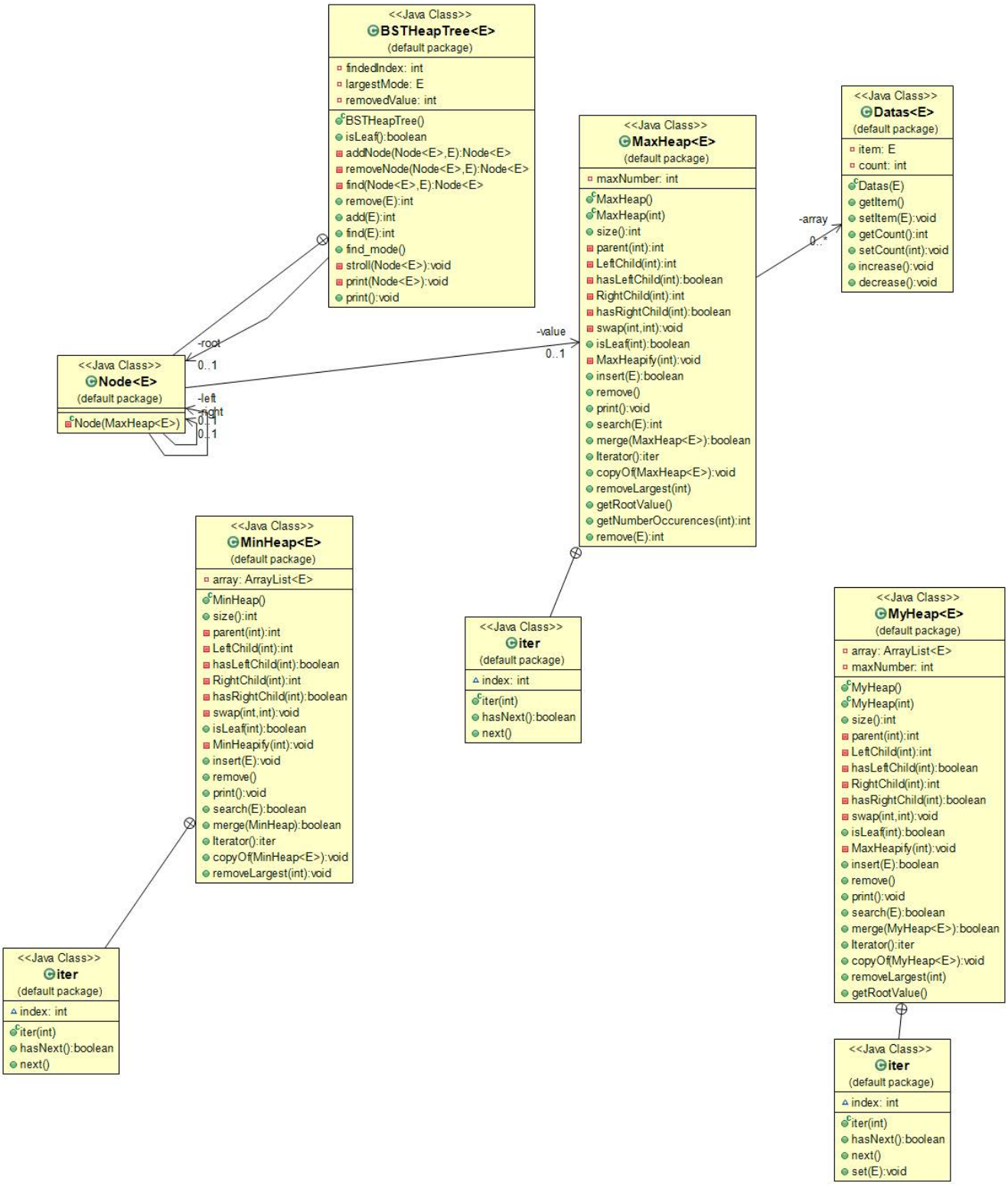
int add (E item) – returns the number of occurrences of the item after insertion
int remove (E item) – returns the number of occurrences of the item after removal
int find (E) – returns the number of occurrences of the item in the BSTHeapTree
find_mode () - returns the element of the largest mode.
boolean search(E item) - Search for an element
boolean merge(MyHeap<E> other) - Merge with another heap
E removeLargest(int index) - Removing ith largest element from the Heap
iter Iterator() - Extend the Iterator class by adding a method to set the value (value passed as
parameter) of the last element returned by the next methods.

Unfunctional Requirments

Java language

2. USE CASE AND CLASS DIAGRAMS

2.1 Class Diagram



3. PROBLEM SOLUTION APPROACH

One myheap class has been created. The myheap class arranges the elements in a binary tree structure, ordering them in descending order. When you add any element, it will always be the largest element at the top of the tree. When you want to delete an element, it deletes it from the top of the tree. We can combine two myheap with each other.

A maxheap class has been created. The maxheap class is very similar to the myheap class. Myheap with maxheap

The difference is that it also keeps the frequency of the data in the maxheap. A BSTHeap class has been created. The BTSHeap class is very similar to the binary tree search class. However, the BTSHeap class keeps one maxheap data in each element.

4. TEST CASES

1. Insert the 3000 numbers that are randomly generated in the range 0-5000 into the BSTHeapTree. Store these numbers in an array as well. Sort the numbers to find the number occurrences of all the numbers.
2. Search for 100 numbers in the array and 10 numbers not in the array and make sure that the number of occurrences is correct.
3. Find the mode of the BSTHeapTree. Check whether the mode value is correct.
4. Remove 100 numbers in the array and 10 numbers not in the array and make sure that the number of occurrences after removal is correct.

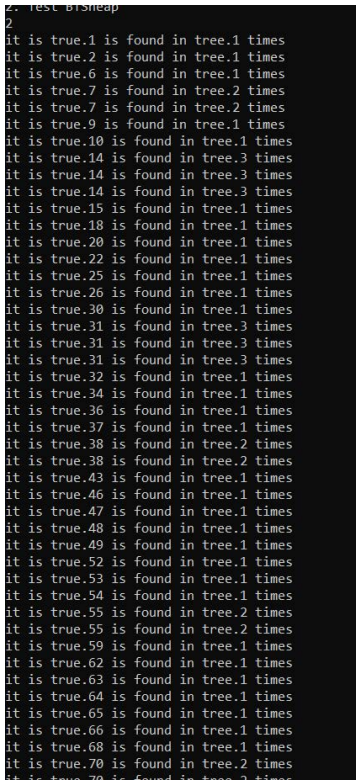
5.RUNNING AND RESULTS

Heap Test

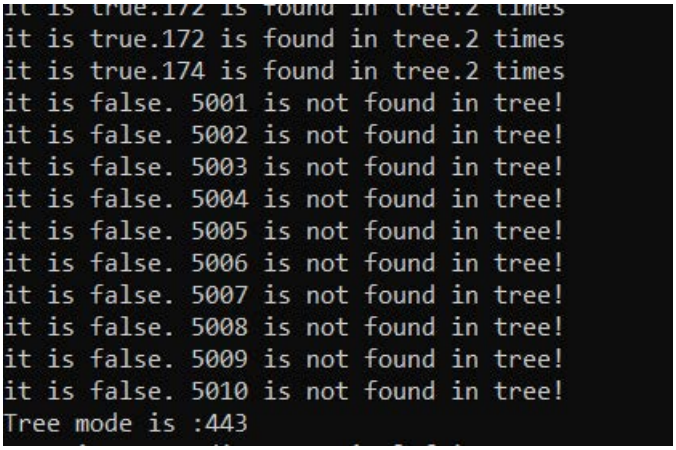
```
2. Test BTSheap
1
Heap 1:
 PARENT : 3 LEFT CHILD : 1 RIGHT CHILD :2
Heap 2:
 PARENT : 7 LEFT CHILD : 6 RIGHT CHILD :5
 PARENT : 6 LEFT CHILD : 4
Heap 2 after merge:
 PARENT : 7 LEFT CHILD : 6 RIGHT CHILD :5
 PARENT : 6 LEFT CHILD : 4 RIGHT CHILD :3
 PARENT : 5 LEFT CHILD : 1 RIGHT CHILD :2
Remove Heap 2 first element :7
 PARENT : 6 LEFT CHILD : 4 RIGHT CHILD :5
 PARENT : 4 LEFT CHILD : 2 RIGHT CHILD :3
 PARENT : 5 LEFT CHILD : 1
Remove 1. largest element6
5
4
3
2
```

BSTHeap Test

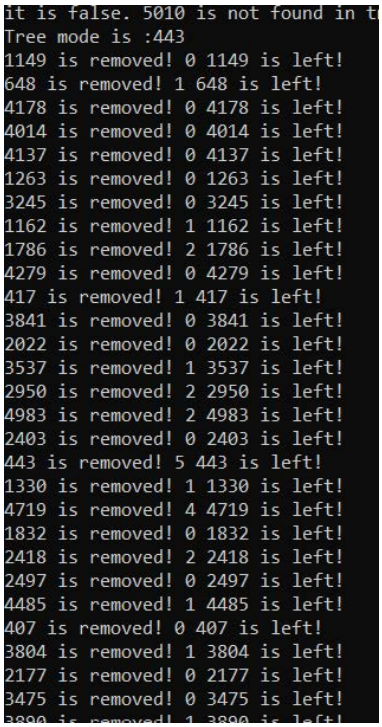
Search in tree.Number of occurrences.



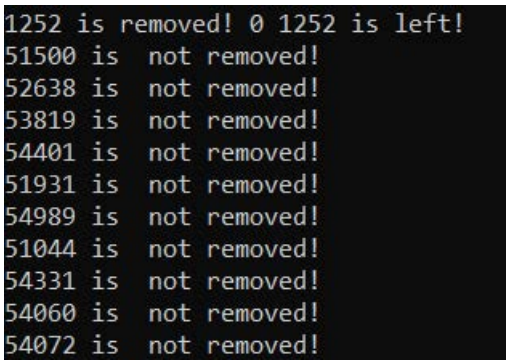
Search not elements of tree.



Remove elements in tree.



Remove tree which is not in tree.



6. Time Complexity Analyze

Search

```
public boolean search(E item) {
    for(int i=0;i<array.size();i++)
        if(array.get(i).equals(item))
            return true;
    return false;

}
```

Merge

```
public boolean merge(MyHeap<E> other) {

    Iterator<E> iter = other.Iterator();

    while(iter.hasNext() != false) {
        this.insert(iter.next());
    }
    return true;
}
```

Remove i. element

```
public E removeLargest(int index) {

    E returnedValue = null;

    MyHeap<E> other = new MyHeap<E>();
    E temp = (E) this.remove();
    int count=0;
    while(temp != null) {
        if(count != index-1) {
            other.insert(temp);
        }
        else {
            returnedValue = temp;
        }
        temp = (E) this.remove();
        count++;
    }

    this.copyOf(other);
    return returnedValue;

}
```