

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

**Coşkun Hasan ŞALTU**  
**1801042631**

## 1. SYSTEM REQUIREMENTS

### Functional Requirments

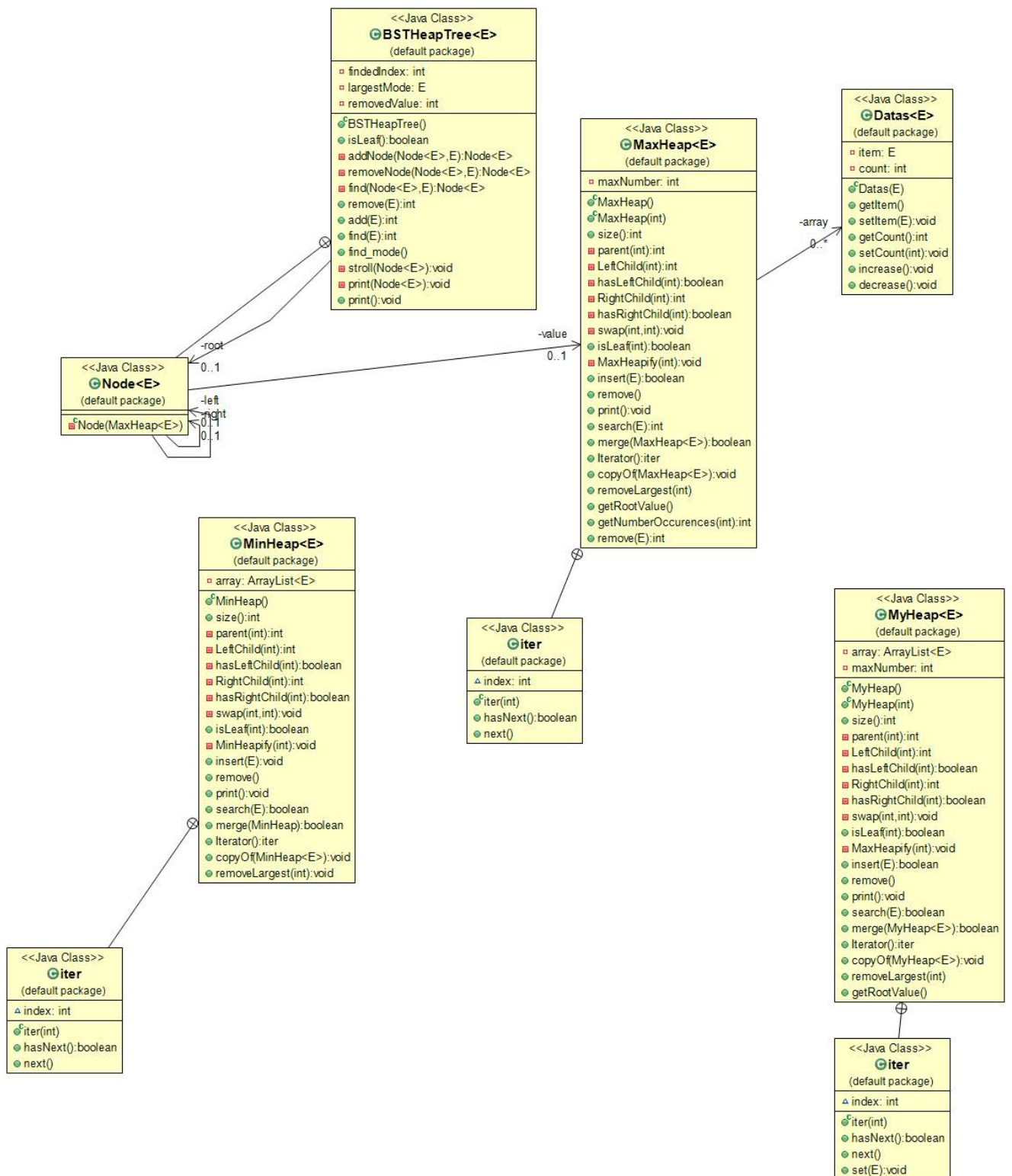
int add (E item) – returns the number of occurrences of the item after insertion  
int remove (E item) – returns the number of occurrences of the item after removal  
int find (E) – returns the number of occurrences of the item in the BSTHeapTree  
find\_mode () - returns the element of the largest mode.  
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 BSTHeap class is very similar to the binary tree search class. However, the BSTHeap 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 BSTHeap
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.

```
2. test BSTHeap
2
it is true.1 is found in tree.1 times
it is true.2 is found in tree.1 times
it is true.6 is found in tree.1 times
it is true.7 is found in tree.2 times
it is true.7 is found in tree.2 times
it is true.9 is found in tree.1 times
it is true.10 is found in tree.1 times
it is true.14 is found in tree.3 times
it is true.14 is found in tree.3 times
it is true.14 is found in tree.3 times
it is true.15 is found in tree.1 times
it is true.18 is found in tree.1 times
it is true.20 is found in tree.1 times
it is true.22 is found in tree.1 times
it is true.25 is found in tree.1 times
it is true.26 is found in tree.1 times
it is true.30 is found in tree.1 times
it is true.31 is found in tree.3 times
it is true.31 is found in tree.3 times
it is true.31 is found in tree.3 times
it is true.32 is found in tree.1 times
it is true.34 is found in tree.1 times
it is true.36 is found in tree.1 times
it is true.37 is found in tree.1 times
it is true.38 is found in tree.2 times
it is true.38 is found in tree.2 times
it is true.43 is found in tree.1 times
it is true.46 is found in tree.1 times
it is true.47 is found in tree.1 times
it is true.48 is found in tree.1 times
it is true.49 is found in tree.1 times
it is true.52 is found in tree.1 times
it is true.53 is found in tree.1 times
it is true.54 is found in tree.1 times
it is true.55 is found in tree.2 times
it is true.55 is found in tree.2 times
it is true.59 is found in tree.1 times
it is true.62 is found in tree.1 times
it is true.63 is found in tree.1 times
it is true.64 is found in tree.1 times
it is true.65 is found in tree.1 times
it is true.66 is found in tree.1 times
it is true.68 is found in tree.1 times
it is true.70 is found in tree.2 times
it is true.70 is found in tree.2 times
```

Search not elements of tree.

```
it is true.172 is found in tree.2 times
it is true.172 is found in tree.2 times
it is true.174 is found in tree.2 times
it is false. 5001 is not found in tree!
it is false. 5002 is not found in tree!
it is false. 5003 is not found in tree!
it is false. 5004 is not found in tree!
it is false. 5005 is not found in tree!
it is false. 5006 is not found in tree!
it is false. 5007 is not found in tree!
it is false. 5008 is not found in tree!
it is false. 5009 is not found in tree!
it is false. 5010 is not found in tree!
Tree mode is :443
```

Remove elements in tree.

```
it is false. 5010 is not found in t
Tree mode is :443
1149 is removed! 0 1149 is left!
648 is removed! 1 648 is left!
4178 is removed! 0 4178 is left!
4014 is removed! 0 4014 is left!
4137 is removed! 0 4137 is left!
1263 is removed! 0 1263 is left!
3245 is removed! 0 3245 is left!
1162 is removed! 1 1162 is left!
1786 is removed! 2 1786 is left!
4279 is removed! 0 4279 is left!
417 is removed! 1 417 is left!
3841 is removed! 0 3841 is left!
2022 is removed! 0 2022 is left!
3537 is removed! 1 3537 is left!
2950 is removed! 2 2950 is left!
4983 is removed! 2 4983 is left!
2403 is removed! 0 2403 is left!
443 is removed! 5 443 is left!
1330 is removed! 1 1330 is left!
4719 is removed! 4 4719 is left!
1832 is removed! 0 1832 is left!
2418 is removed! 2 2418 is left!
2497 is removed! 0 2497 is left!
4485 is removed! 1 4485 is left!
407 is removed! 0 407 is left!
3804 is removed! 1 3804 is left!
2177 is removed! 0 2177 is left!
3475 is removed! 0 3475 is left!
3880 is removed! 1 3880 is left!
```

Remove tree which is not in tree.

```
1252 is removed! 0 1252 is left!
51500 is not removed!
52638 is not removed!
53819 is not removed!
54401 is not removed!
51931 is not removed!
54989 is not removed!
51044 is not removed!
54331 is not removed!
54060 is not removed!
54072 is not removed!
```

## 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;  
}
```

$\rightarrow O(n)$   
 $\rightarrow O(1)$   
 $T(n) = O(n)$   
 $T(n) = O(n)$

Merge

```
public boolean merge(MyHeap<E> other) {  
    Iterator<E> iter = other.iterator();  
    while(iter.hasNext() != false) {  
        this.insert(iter.next());  
    }  
    return true;  
}
```

$\rightarrow O(1)$   
 $\rightarrow O(n)$   
 $\rightarrow O(\log(n))$   
 $T(n) = O(n \cdot \log(n))$

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;  
}
```

$\rightarrow O(\log(n))$   
 $\rightarrow O(n)$   
 $\rightarrow O(\log(n))$   
 $\rightarrow O(\log(n))$   
 $T(n) = O(n \cdot \log(n))$