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

Barış Ayyıldız
1901042252

1. SYSTEM REQUIREMENTS

File structure :

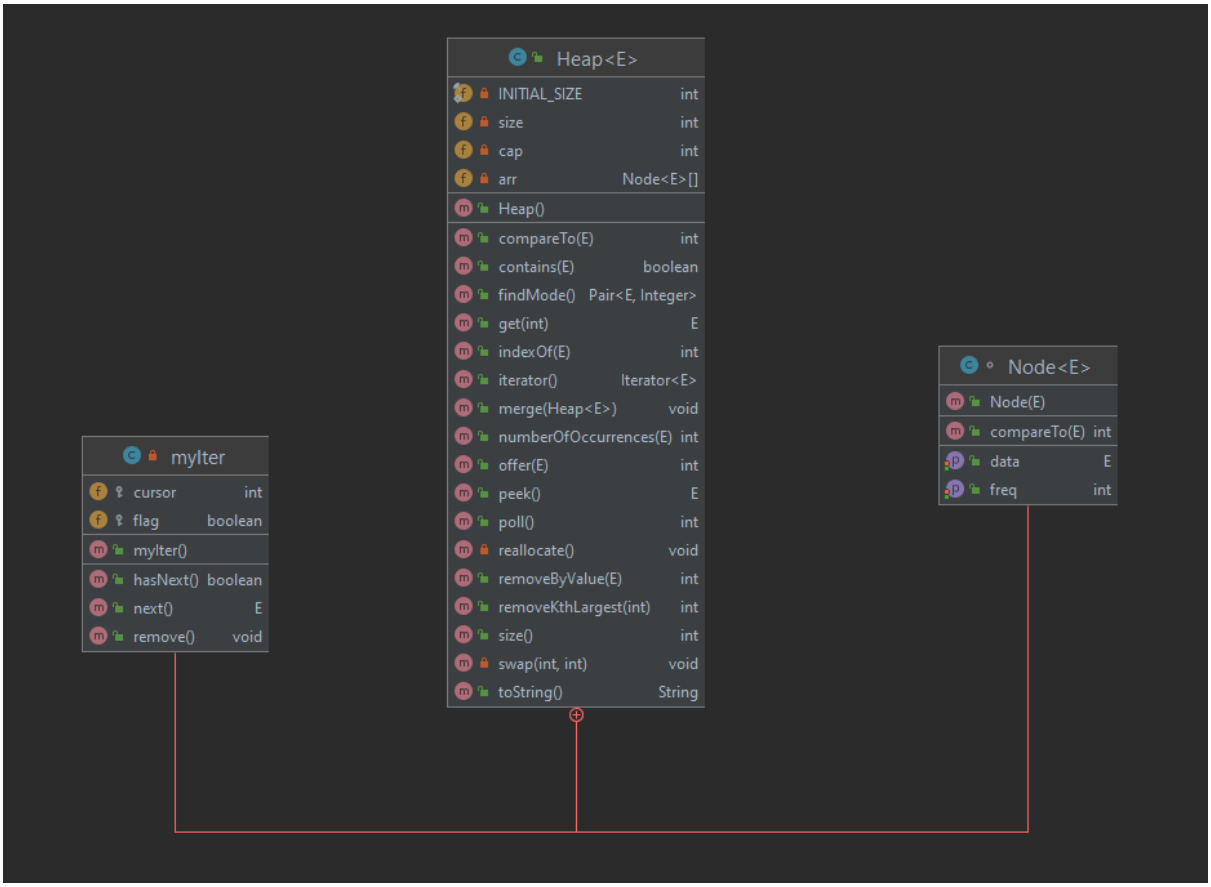
1. com
    a. lib
        i. Pair
    b. structures
        i. Heap
        ii. BinarySearchTree

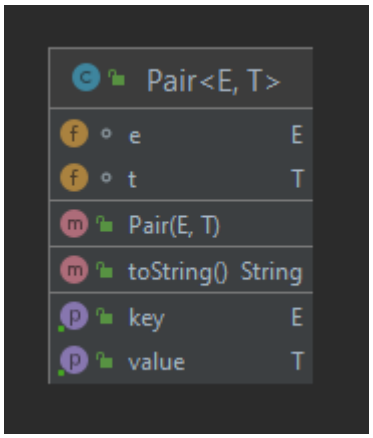
User should Heap and BinarySearchTree data structures use Pair class. Pair class hold a key value pair. And it is used to calculate mode of a heap. BinarySearchTree also use Heap data structure. So it is better if the the imports all the classes at once. Such as

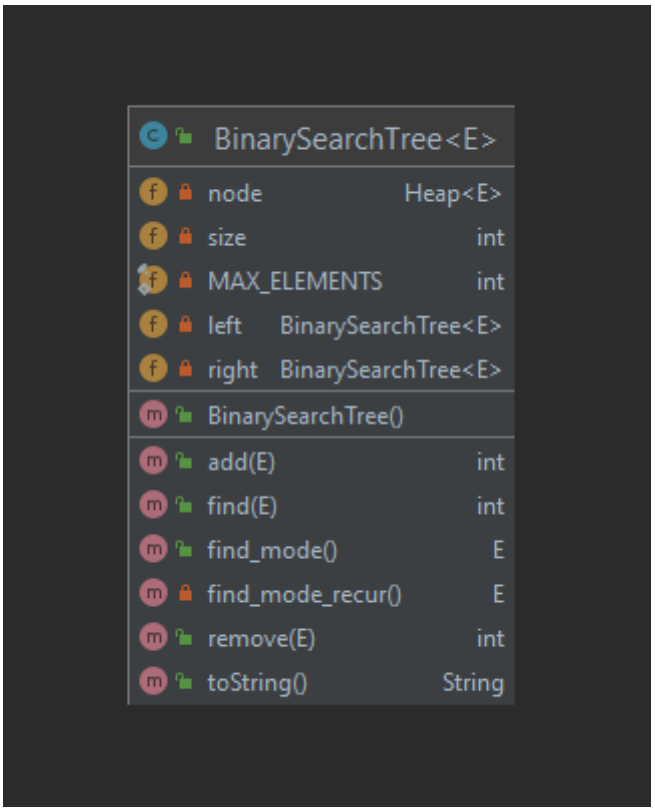
import com.lib.*;

import com.structures.*;

2. CLASS DIAGRAMS







3. PROBLEM SOLUTION APPROACH

In the homework we were asked to implement generic Heap and Binary Search Tree data structures. For the Heap implementation I used simple java arrays. Initial size set to 8. And the capacity is duplicated when the size reaches capacity.

In part-1, we had to be able to insert the same values multiple times and in part-2 we were supposed to only increase the number of occurrences. **That's why there are two constructors for Heap. One is no parameter constructor, other one takes a boolean value. If the boolean value is true user not able to insert same value twice(for part-2),** it only increases the frequency. But it is false by default.

Every item in the array is a Node<E> class object. Node class contains a generic(E) data and the number of occurrences of that data. Node<E> is an inner class.

I implemented **contains(E)** for searching a value in the heap, **merge(Heap<E>)** to merge two heaps, **removeKthLargest(int)** to remove kth largest element in the heap. I was not able to implement set method for the iterator, I had some problems with the generic types. And I also implemented a couple more useful methods

In binary search tree, every node is a Heap<E> class object. And the maximum size a node can have is 7. There are also left and right pointers for every node.

add(E) method calls it's heap's offer method, remove(E) method call's it's heap's remove method, find(E) methods looks given parameter int the whole BST

4. TEST CASES

TEST ID	Scenerio	Test Data	Expected Result	Actual Results	Pass/Fail
001	size() method of Heap class called	e1 : [] e2 : [7,3,5]	0 3	As expected	Pass
002	poll() method of Heap class called	e : [7,3,5]	[5,3]	As expected	Pass
003	peek() method of Heap class called	e:[5,3]	5	As Expected	Pass
004	contains() method of Heap class called	input : 5 input : 12 e: [5,3]	true false	As Expected	Pass
005	merge() method of Heap class called	e1:[5,3] e2 : [11,9,4,2,-6]	[11,9,5,3,4,2,-6]	As Expected	Pass
006	removeKthLargest() method of Heap class called	input : 1 e:[11,9,5,3,4,2,-6]	[9,5,3,4,2,-6]	As Expected	Pass
007	removeByValue() method of Heap class called	input : 5 e:[9,3,5,2,-6]	[9,3,-6,2]	As Expected	Pass
008	indexOf() method of Heap class called	input : 2 e:[9,3,5,2,-6]	3	As Expected	Pass
009	numberOfOccurrences() method of Heap class called	inputs : 5,8 e:[[12,1], [8,4], [11,1],[2,3],[5,1],[4,1]]	1,4	As Expected	Pass
010	findMode() method of Heap class called	e:[[12,1], [8,4], [11,1],[2,3],[5,1],[4,1]]	8	As Expected	Pass
011	Heap's iterator's next() method called	e:[73,34,1,2,7]	73	As Expected	Pass
012	Heap's iterator's remove() method called after calling next method twice	e:[73,34,1,2,7]	[73,7,1,2]	As Expected	Pass
013	BST, add() method called	input : 9,3,12,12 e : []	[[12,2],[9,1],[3,1]]	As Expected	Pass
014	BST, remove() method called	input : 3 e : [[12,2],[9,1],[3,1]]	[[12,2],[9,1]]	As Expected	Pass
015	BST, find() method called	input : 12 e : [[12,2],[9,1]]	2	As Expected	Pass
016	BST, find_mode() method called	e : [[12,2],[9,1]]	12	As Expected	Pass

5. RUNNING AND RESULTS

```
// ==================== HEAP METHODS ==================== //
Heap, offer method :
Before :
After : 7 | 3 | 5 |

Heap, size method :
Size : 3

Heap, poll method :
Before : 7 | 3 | 5 |
After : 5 | 3 |

Heap, peek method :
Element in the root : 5

Heap, contains method :
Current heap : 5 | 3 |
Is it contains 5 : true
Is it contains 12 : false

Heap, merge method :
heap-1 : 5 | 3 |
heap-2 : 11 | 9 | 4 | 2 | -6 |
After merging : 11 | 9 | 5 | 3 | 4 | 2 | -6 |

Heap, removeKthLargest method :
After removing 1st greatest number : 9 | 4 | 5 | 3 | -6 | 2 |
After removing 3rd greatest number : 9 | 3 | 5 | 2 | -6 |
Trying to remove -2nd greatest element... : 9 | 3 | 5 | 2 | -6 |

Heap, removeByValue method :
Before : 9 | 3 | 5 | 2 | -6 |
After removing the value 5 : 9 | 3 | -6 | 2 |

Heap, indexOf method :
Current heap : 9 | 3 | -6 | 2 |
Index of 2 : 3
Index of 55 : -1, so it does not exists
```

```
Heap, numberOfOccurrences :
Current heap : 12,1 | 8,4 | 11,1 | 2,3 | 5,1 | 4,1 |
Frequency of 5 : 1
Frequency of 8 : 4

Heap, findMode :
Current heap : 12,1 | 8,4 | 11,1 | 2,3 | 5,1 | 4,1 |
Mode of the heap : 8, it is repated for 4 times

// ==================== HEAP ITERATOR METHODS ==================== //
Current heap : 73 | 34 | 1 | 2 | 7 |
iter.next : 73
iter.next : 34
iter.next : 1
iter.next : 2
iter.next : 7
Iterator reseted...
Removing second element
Heap : 73 | 7 | 1 | 2 |

// ==================== BINARY SEARCH TREE METHODS ==================== //
BST, add method :
Before :
After : 9,1 | 12,2 | 6,1 | 7,1 | 3,3 | -4,4 | 1,1 | 5,2 | 55,2 | 23,1 |


        INORDER TRAVERSAL IS USED!


After removing 9, there are  : 0, 9 left  -> 12,2 | 6,1 | 7,1 | 3,3 | -4,4 | 1,1 | 5,2 | 55,2 | 23,1 |
After removing 55, there are  : 1, 55 left  -> 12,2 | 6,1 | 7,1 | 3,3 | -4,4 | 1,1 | 5,2 | 55,1 | 23,1 |

BST, find method :
Current : 12,2 | 6,1 | 7,1 | 3,3 | -4,4 | 1,1 | 5,2 | 55,1 | 23,1 |
Number of occurences of 12 : 2
Number of occurences of -4 : 4
Number of occurences of 100 : 0

BST, find_mode method :
Current : 12,2 | 6,1 | 7,1 | 3,3 | -4,4 | 1,1 | 5,2 | 55,1 | 23,1 |
Mode : -4
```

```
// ================= TEST CASES FOR PART 2 ================= //

Number of occurences of 2096    in BST is 2, in array : 2
Number of occurences of 796     in BST is 1, in array : 1
Number of occurences of 414     in BST is 0, in array : 0
Number of occurences of 1748    in BST is 0, in array : 0
Number of occurences of 1243    in BST is 2, in array : 2
Number of occurences of 2793    in BST is 1, in array : 1
Number of occurences of 2646    in BST is 0, in array : 0
Number of occurences of 2343    in BST is 0, in array : 0
Number of occurences of 1564    in BST is 0, in array : 0
Number of occurences of 744     in BST is 0, in array : 0
Number of occurences of 1485    in BST is 0, in array : 0
Number of occurences of 2461    in BST is 1, in array : 1
Number of occurences of 2219    in BST is 0, in array : 0
Number of occurences of 1087    in BST is 1, in array : 1
Number of occurences of 2008    in BST is 0, in array : 0
Number of occurences of 1765    in BST is 1, in array : 1
Number of occurences of 958     in BST is 0, in array : 0
Number of occurences of 2964    in BST is 1, in array : 1
Number of occurences of 1734    in BST is 1, in array : 1
Number of occurences of 702     in BST is 0, in array : 0
Number of occurences of 1228    in BST is 0, in array : 0
Number of occurences of 2728    in BST is 1, in array : 1
Number of occurences of 2797    in BST is 0, in array : 0
Number of occurences of 156     in BST is 0, in array : 0
Number of occurences of 1420    in BST is 0, in array : 0
Number of occurences of 2838    in BST is 0, in array : 0
Number of occurences of 2138    in BST is 2, in array : 2
Number of occurences of 1253    in BST is 0, in array : 0
Number of occurences of 899     in BST is 3, in array : 3
Number of occurences of 2231    in BST is 0, in array : 0
Number of occurences of 2264    in BST is 0, in array : 0
Number of occurences of 2801    in BST is 0, in array : 0
Number of occurences of 2796    in BST is 2, in array : 2
Number of occurences of 2165    in BST is 1, in array : 1
Number of occurences of 2274    in BST is 1, in array : 1
Number of occurences of 580     in BST is 2, in array : 2
Number of occurences of 1753    in BST is 1, in array : 1
Number of occurences of 1754    in BST is 0, in array : 0
Number of occurences of 2787    in BST is 0, in array : 0
Number of occurences of 370     in BST is 0, in array : 0
Number of occurences of 1152    in BST is 2, in array : 2
Number of occurences of 36      in BST is 2, in array : 2
```

```
Number of occurences of 2802    in BST is 1, in array : 1
Number of occurences of 2605    in BST is 0, in array : 0
Number of occurences of 1306    in BST is 0, in array : 0
Number of occurences of 2095    in BST is 1, in array : 1
Number of occurences of 1900    in BST is 1, in array : 1
Number of occurences of 2566    in BST is 1, in array : 1
Number of occurences of 2853    in BST is 1, in array : 1
Number of occurences of 792     in BST is 0, in array : 0
Number of occurences of 657     in BST is 0, in array : 0
Number of occurences of 1470    in BST is 1, in array : 1
Number of occurences of 616     in BST is 1, in array : 1
Number of occurences of 1842    in BST is 1, in array : 1
Number of occurences of 2897    in BST is 0, in array : 0
Number of occurences of 599     in BST is 0, in array : 0
Number of occurences of 863     in BST is 0, in array : 0
Number of occurences of 2463    in BST is 0, in array : 0
Number of occurences of 783     in BST is 0, in array : 0
Number of occurences of 2452    in BST is 2, in array : 2
Number of occurences of 564     in BST is 1, in array : 1
Number of occurences of 2659    in BST is 3, in array : 3
Number of occurences of 2171    in BST is 0, in array : 0
Number of occurences of 905     in BST is 1, in array : 1
Number of occurences of 2501    in BST is 2, in array : 2
Number of occurences of 2896    in BST is 0, in array : 0
Number of occurences of 2254    in BST is 0, in array : 0
Number of occurences of 2529    in BST is 0, in array : 0
Number of occurences of 2686    in BST is 3, in array : 3
Number of occurences of 634     in BST is 1, in array : 1
Number of occurences of 2998    in BST is 1, in array : 1
Number of occurences of 2810    in BST is 2, in array : 2
Number of occurences of 2251    in BST is 0, in array : 0
Number of occurences of 599     in BST is 0, in array : 0
Number of occurences of 1578    in BST is 2, in array : 2
Number of occurences of 903     in BST is 0, in array : 0
Number of occurences of 2639    in BST is 0, in array : 0
Number of occurences of 413     in BST is 2, in array : 2
Number of occurences of 1407    in BST is 0, in array : 0
Number of occurences of 1882    in BST is 0, in array : 0
Number of occurences of 2155    in BST is 0, in array : 0
Number of occurences of 2415    in BST is 0, in array : 0
Number of occurences of 1632    in BST is 1, in array : 1
Number of occurences of 1587    in BST is 1, in array : 1
Number of occurences of 2590    in BST is 1, in array : 1
Number of occurences of 1205    in BST is 0, in array : 0
```

```
Number of occurences of 492     in BST is 0, in array : 0
Number of occurences of 2970    in BST is 2, in array : 2
Number of occurences of 154     in BST is 0, in array : 0
Number of occurences of 1040    in BST is 1, in array : 1
Number of occurences of 849     in BST is 1, in array : 1
Number of occurences of 1157    in BST is 0, in array : 0
Number of occurences of 1471    in BST is 0, in array : 0
Number of occurences of 7       in BST is 0, in array : 0
Number of occurences of 687     in BST is 0, in array : 0
Number of occurences of 1038    in BST is 0, in array : 0
Number of occurences of 817     in BST is 0, in array : 0
Number of occurences of 1408    in BST is 1, in array : 1
Number of occurences of 2600    in BST is 1, in array : 1
Number of occurences of 1612    in BST is 1, in array : 1



Number of occurences of 6000    in BST is 0, in array : 0
Number of occurences of 6010    in BST is 0, in array : 0
Number of occurences of 6020    in BST is 0, in array : 0
Number of occurences of 6030    in BST is 0, in array : 0
Number of occurences of 6040    in BST is 0, in array : 0
Number of occurences of 6050    in BST is 0, in array : 0
Number of occurences of 6060    in BST is 0, in array : 0
Number of occurences of 6070    in BST is 0, in array : 0
Number of occurences of 6080    in BST is 0, in array : 0
Number of occurences of 6090    in BST is 0, in array : 0

// ================== REMOVAL ================== //

After removal, number of occurences of 205      in BST is 0, in array : 0
After removal, number of occurences of 869      in BST is 1, in array : 1
After removal, number of occurences of 1139     in BST is 0, in array : 0
After removal, number of occurences of 2920     in BST is 0, in array : 0
After removal, number of occurences of 65       in BST is 0, in array : 0
After removal, number of occurences of 2084     in BST is 0, in array : 0
After removal, number of occurences of 2605     in BST is 0, in array : 0
After removal, number of occurences of 2500     in BST is 0, in array : 0
After removal, number of occurences of 1737     in BST is 0, in array : 0
After removal, number of occurences of 72       in BST is 0, in array : 0
After removal, number of occurences of 1061     in BST is 0, in array : 0
After removal, number of occurences of 2880     in BST is 0, in array : 0
After removal, number of occurences of 2649     in BST is 0, in array : 0
After removal, number of occurences of 1888     in BST is 0, in array : 0
```

```
After removal, number of occurences of 1624    in BST is 1, in array : 1
After removal, number of occurences of 1995    in BST is 0, in array : 0
After removal, number of occurences of 1695    in BST is 0, in array : 0
After removal, number of occurences of 971     in BST is 0, in array : 0
After removal, number of occurences of 2465    in BST is 0, in array : 0
After removal, number of occurences of 864     in BST is 0, in array : 0
After removal, number of occurences of 2139    in BST is 0, in array : 0
After removal, number of occurences of 1102    in BST is 0, in array : 0
After removal, number of occurences of 1364    in BST is 0, in array : 0
After removal, number of occurences of 626     in BST is 1, in array : 1
After removal, number of occurences of 400     in BST is 0, in array : 0
After removal, number of occurences of 2886    in BST is 0, in array : 0
After removal, number of occurences of 1626    in BST is 1, in array : 1
After removal, number of occurences of 2333    in BST is 0, in array : 0
After removal, number of occurences of 1431    in BST is 0, in array : 0
After removal, number of occurences of 828     in BST is 1, in array : 1
After removal, number of occurences of 1403    in BST is 2, in array : 2
After removal, number of occurences of 2053    in BST is 0, in array : 0
After removal, number of occurences of 474     in BST is 0, in array : 0
After removal, number of occurences of 2274    in BST is 0, in array : 0
After removal, number of occurences of 2080    in BST is 0, in array : 0
After removal, number of occurences of 999     in BST is 0, in array : 0
After removal, number of occurences of 2154    in BST is 0, in array : 0
After removal, number of occurences of 846     in BST is 0, in array : 0
After removal, number of occurences of 2693    in BST is 0, in array : 0
After removal, number of occurences of 405     in BST is 0, in array : 0
After removal, number of occurences of 977     in BST is 0, in array : 0
After removal, number of occurences of 1784    in BST is 0, in array : 0
After removal, number of occurences of 2321    in BST is 0, in array : 0
After removal, number of occurences of 1633    in BST is 0, in array : 0
After removal, number of occurences of 1971    in BST is 1, in array : 1
After removal, number of occurences of 891     in BST is 0, in array : 0
After removal, number of occurences of 411     in BST is 0, in array : 0
After removal, number of occurences of 2917    in BST is 0, in array : 0
After removal, number of occurences of 898     in BST is 0, in array : 0
After removal, number of occurences of 922     in BST is 0, in array : 0
After removal, number of occurences of 766     in BST is 2, in array : 2
After removal, number of occurences of 1900    in BST is 0, in array : 0
After removal, number of occurences of 197     in BST is 0, in array : 0
After removal, number of occurences of 1197    in BST is 0, in array : 0
After removal, number of occurences of 2303    in BST is 0, in array : 0
After removal, number of occurences of 1025    in BST is 0, in array : 0
After removal, number of occurences of 282     in BST is 1, in array : 1
After removal, number of occurences of 2220    in BST is 0, in array : 0
```

```
After removal, number of occurences of 265     in BST is 0, in array : 0
After removal, number of occurences of 2789    in BST is 0, in array : 0
After removal, number of occurences of 211     in BST is 0, in array : 0
After removal, number of occurences of 1422    in BST is 0, in array : 0
After removal, number of occurences of 2605    in BST is 0, in array : 0
After removal, number of occurences of 957     in BST is 1, in array : 1
After removal, number of occurences of 794     in BST is 0, in array : 0
After removal, number of occurences of 2052    in BST is 0, in array : 0
After removal, number of occurences of 2202    in BST is 0, in array : 0
After removal, number of occurences of 83      in BST is 0, in array : 0
After removal, number of occurences of 2530    in BST is 0, in array : 0
After removal, number of occurences of 2879    in BST is 0, in array : 0
After removal, number of occurences of 2644    in BST is 0, in array : 0
After removal, number of occurences of 45      in BST is 0, in array : 0
After removal, number of occurences of 2358    in BST is 1, in array : 1
After removal, number of occurences of 806     in BST is 0, in array : 0
After removal, number of occurences of 2709    in BST is 0, in array : 0
After removal, number of occurences of 2413    in BST is 0, in array : 0
After removal, number of occurences of 2317    in BST is 0, in array : 0
After removal, number of occurences of 2118    in BST is 0, in array : 0
After removal, number of occurences of 1378    in BST is 0, in array : 0
After removal, number of occurences of 1751    in BST is 0, in array : 0
After removal, number of occurences of 2112    in BST is 0, in array : 0
After removal, number of occurences of 1574    in BST is 2, in array : 2
After removal, number of occurences of 935     in BST is 0, in array : 0
After removal, number of occurences of 607     in BST is 0, in array : 0
After removal, number of occurences of 288     in BST is 0, in array : 0
After removal, number of occurences of 490     in BST is 0, in array : 0
After removal, number of occurences of 36      in BST is 1, in array : 1
After removal, number of occurences of 2237    in BST is 0, in array : 0
After removal, number of occurences of 1809    in BST is 1, in array : 1
After removal, number of occurences of 1686    in BST is 0, in array : 0
After removal, number of occurences of 632     in BST is 0, in array : 0
After removal, number of occurences of 1043    in BST is 0, in array : 0
After removal, number of occurences of 1776    in BST is 0, in array : 0
After removal, number of occurences of 1519    in BST is 0, in array : 0
After removal, number of occurences of 589     in BST is 0, in array : 0
After removal, number of occurences of 1448    in BST is 0, in array : 0
After removal, number of occurences of 1689    in BST is 0, in array : 0
After removal, number of occurences of 351     in BST is 0, in array : 0
After removal, number of occurences of 636     in BST is 0, in array : 0
After removal, number of occurences of 1895    in BST is 0, in array : 0
```

```
After removal, number of occurences of 6000     in BST is 0, in array : 0
After removal, number of occurences of 6010     in BST is 0, in array : 0
After removal, number of occurences of 6020     in BST is 0, in array : 0
After removal, number of occurences of 6030     in BST is 0, in array : 0
After removal, number of occurences of 6040     in BST is 0, in array : 0
After removal, number of occurences of 6050     in BST is 0, in array : 0
After removal, number of occurences of 6060     in BST is 0, in array : 0
After removal, number of occurences of 6070     in BST is 0, in array : 0
After removal, number of occurences of 6080     in BST is 0, in array : 0
After removal, number of occurences of 6090     in BST is 0, in array : 0
Mode of BST : 2250
Mode of array : 2250

If there is an error finding the modes, the values should have the same frequency...
```

PART-3

```java
@SuppressWarnings("unchecked")
public Heap(boolean noDuplicate)
{
  this.cap = INITIAL_SIZE;
  this.size = 0;
  this.arr = new Node[this.cap];
  this.noDuplicate = noDuplicate;
}
```

Θ(1)

```java
public int offer(E e)
{
  if(this.size == this.cap) this.reallocate();

  if(this.noDuplicate){
    if(this.contains(e)){
      int index = this.indexOf(e);
      this.arr[index].setFreq(this.arr[index].getFreq()+1);
      return this.arr[index].getFreq();
    }
  }

  this.arr[this.size] = new Node<E>(e);

  int child = this.size;
  int parent = (child-1)/2;
  while(true)
  {
    if(this.arr[child].compareTo(this.arr[parent].getData()) > 0){
      this.swap(child, parent);
      child = parent;
      parent = (child-1)/2;
    }else{
      break;
    }

  }

  this.size++;
  return 1;
}
```

Θ(n) → Amortized

Θ(1)

Θ(n)

O(n)

Θ(1)

Θ(1)

O(n²)

$T_{worst} = O(n^2)$

Θ(1)

h: height of the tree (log n) → O(h) = O(log n)

Θ(1)

Θ(1)

Θ(1)

Θ(1)

Θ(1)

T_{best} (Amortized) = O(log n)

```java
@SuppressWarnings("unchecked")
public Heap()
{
  this.cap = INITIAL_SIZE;
  this.size = 0;
  this.arr = new Node[this.cap];
  this.noDuplicate = false;
}
```

Θ(1)

```java
public E peek()
{
  if(this.size() == 0)  return null;
  return this.arr[0].getData();
}
```

$\Theta(1)$

```java
public int size(){
  return this.size;
}
```

$\Theta(1)$

```java
public boolean contains(E e)
{
  for(int i=0; i<this.size; i++)
    if(this.arr[i].compareTo(e) == 0)
      return true;
  return false;
}
```

$O(n)$ · $O(1)$ → $\Theta(1)$ → $O(n)$

```java
public void merge(Heap<E> h1)
{
  for(int i=0; i<h1.size(); i++){
    this.offer(h1.arr[i].getData());
  }
}
```

$\Theta(n)$ → $T_{worst} = O(m^2)$

$T_{best} = O(\log m)$

$T_{worst} = O(nm^2)$

$= T_{best} = O(n \log m)$

```java
public int removeKthLargest(int k) throws Exception{

  if(k > this.size() || k <= 0)
    throw new Exception("Index is out of bound...");

  k = this.size() - k + 1;

  Heap<E> temp = new Heap<E>();

  for(int i=0; i<this.size(); i++){
    temp.offer(this.arr[i].getData());
    if(temp.size() > k){
      temp.poll();
    }
  }

  this.removeByValue(temp.peek());

  return temp.poll();

}
```

Θ(1)

Θ(n) O(n²) O(n³)

O(log n)

O(n) Θ(1) O(n)

O(log n)

= O(n³)

```java
public int removeByValue(E val){

  int index = this.indexOf(val);

  if(index < 0) return -1;

  if(this.noDuplicate){
    if(this.arr[index].getFreq() > 1){
      this.arr[index].setFreq(this.arr[index].getFreq() - 1);
      return this.arr[index].getFreq();
    }
  }

  this.arr[index] = this.arr[--this.size];


  int parent = index;
  int left, right;

  while(true)
  {
    left = 2*parent + 1;

    if(left >= this.size())
      break;

    right = left+1;

    int maxChild = left;
    if(right < this.size() && this.arr[right].compareTo(this.arr[left].getData()) > 0){
      maxChild = right;
    }

    if(this.arr[parent].compareTo(this.arr[maxChild].getData()) < 0){
      this.swap(parent, maxChild);
      parent = maxChild;
    }else{
      break;
    }

  }

  return 0;

}
```

O(n)

Θ(1)

Θ(1)

Θ(n)

O(log n)

Θ(1)

O(log n)

O(n)

```java
public int indexOf(E e){

  for(int i=0; i<this.size(); i++){ O(n)
    if(this.arr[i].compareTo(e) == 0) Θ(1)
      return i; Θ(1)
  }
  return -1; Θ(1)
}
```

O(n)

```java
public int numberOfOccurrences(E e){
  int total = 0; Θ(1)
  for(int i=0; i<this.size; i++){ Θ(n)
    if(this.arr[i].compareTo(e) == 0) Θ(1)
      total += this.arr[i].getFreq(); Θ(1)
  }
  return total; Θ(1)
}
```

Θ(n)

```java
@SuppressWarnings("unchecked")
private void reallocate()
{
  this.cap *= 2; Θ(1)        Θ(1)
  Node<E>[] temp = new Node[this.cap];

  for(int i=0; i<this.size; i++) Θ(n)
    temp[i] = this.arr[i]; Θ(1)
  this.arr = temp; Θ(1)
}
```

Θ(n)

```java
private void swap(int i1, int i2){
  Node<E> temp = this.arr[i1]; Θ(1)
  this.arr[i1] = this.arr[i2]; Θ(1)
  this.arr[i2] = temp; Θ(1)
}
```

Θ(1)

```java
@Override
public String toString()
{
  String str = ""; Θ(1)

  if(!this.noDuplicate){ Θ(1)
    for(int i=0; i<this.size; i++) Θ(n)
    {
      str += String.valueOf(this.arr[i].getData() + " | ") ; Θ(1)
    }
  }else{
    for(int i=0; i<this.size; i++)
    {
      str += String.valueOf(this.arr[i].getData() + "," + this.arr[i].getFreq() + " | ") ;
    }
  }

  return str; Θ(1)
}
```

Because Strings are immutable

Θ(n) Θ(n²) Θ(n²) Θ(n²)

```java
@Override
public int compareTo(E e){
  return this.arr[0].compareTo(e); Θ(1)
}
```

```java
public E get(int index) throws Exception{
  if(index < 0 || index >= this.size()) throw new Exception("Index is out of bounds...");
  return this.arr[index].getData();
}
```

Θ(1)

```java
public Pair<E,Integer> findMode(){
  E mode = null;
  int max = 0;

  for(int i=0; i<this.size; i++){
    if(this.arr[i].getFreq() > max){
      max = this.arr[i].getFreq();
      mode = this.arr[i].getData();
    }
  }

  return new Pair<E,Integer>(mode,max);

}
```

Θ(1) Θ(n) Θ(1) Θ(n)

```java
public Iterator<E> iterator(){
  return new myIter();
}
```

Θ(1)

```
public myIter(){
  this.cursor = 0;
  this.flag = false;
}
```
Θ(1)

```
public boolean hasNext(){
  return this.cursor != size();
}
```
Θ(1)

```
public E next(){
  if(this.cursor == size())
    throw new NoSuchElementException("No such element...");

  this.flag = true;
  return arr[this.cursor++].getData();
}
```
Θ(1) Θ(1) Θ(1) Θ(1)

```
public void remove(){
  if(!this.flag)
    throw new IllegalStateException("Next method needs to be called...");

  removeByValue(arr[this.cursor-1].getData());
  this.flag = false;
}
```
Θ(1) O(n) Θ(1) O(n)

```
public Node(E data){
  this.data = data;
  this.freq = 1;
}
```
Θ(1)

```
public E getData(){
  return this.data;
}
```
Θ(1)

```java
public int getFreq(){
  return this.freq;
}
```
Θ(1)

```java
public void setFreq(int freq){
  this.freq = freq;
}
```
Θ(1)

```java
public void setData(E data){
  this.data = data;
}
```
Θ(1)

```java
@Override
public int compareTo(E e){
  return this.data.compareTo(e);
}
```
Θ(1)

```java
public BinarySearchTree(){
  this.node = new Heap<E>(true);
  this.size = 0;
  this.left = null;
  this.right = null;
}
```
Θ(1)

```java
public int add(E e){

  int val;

  if( (this.node.size() >= MAX_ELEMENTS && this.node.contains(e)) || (this.node.size() < MAX_ELEMENTS) ){
    val = this.node.offer(e);
  }else{

    if(this.node.compareTo(e) < 0){
      if(this.right == null) this.right = new BinarySearchTree<E>();
      val = this.right.add(e);
    }else{
      if(this.left == null) this.left = new BinarySearchTree<E>();
      val = this.left.add(e);
    }

  }

  return val;

}
```

θ(1) θ(1) if the heap is full, size is constant (7) it does not depend on the size of BST

θ(1) θ(1) θ(1) O(logn)

= O(logn)

```java
public int remove(E item){

  int val = this.node.removeByValue(item);
  if(val >= 0)  return val;

  if(this.left != null){
    val = this.left.remove(item);
    if(val >= 0) return val;
  }

  if(this.right != null){
    val = this.right.remove(item);
    if(val >= 0) return val;
  }

  return val;

}
```

max number is constant (7)

θ(1)

O(logn)

Tworst = O(logn)

Tbest = θ(1)

```java
public int find(E item){

  int total = 0;
  total += this.node.numberOfOccurrences(item);
  if(this.left != null)
    total += this.left.find(item);
  if(this.right != null)
    total += this.right.find(item);
  return total;
}
```

It traverses all the tree

θ(n)

```java
public E find_mode(){
  Pair<E,Integer> pair = this.find_mode_recur();
  return pair.getKey();
}
```

$\Theta(n)$

```java
private Pair<E,Integer> find_mode_recur(){

  E mode = null;
  int max = 0;

  Pair<E,Integer> pair = new Pair<E,Integer>(mode, max);

  pair = this.node.findMode();
  if(pair.getValue() > max){
    max = pair.getValue();
    mode = pair.getKey();
  }

  if(this.left != null){
    pair = this.left.find_mode_recur();
    if(pair.getValue() > max){
      max = pair.getValue();
      mode = pair.getKey();
    }
  }

  if(this.right != null){
    pair = this.right.find_mode_recur();
    if(pair.getValue() > max){
      max = pair.getValue();
      mode = pair.getKey();
    }
  }

  return new Pair<E,Integer>(mode, max);

}
```

$\Theta(1)$

$\rightarrow \Theta(1)$, max size is constant

$\Theta(1)$

$\Theta(n)$

$= \Theta(n)$

$\Theta(1)$

```java
@Override
public String toString(){

  String str = "";
  if(this.left != null) str += this.left.toString();
  str += this.node.toString();
  if(this.right != null) str += this.right.toString();

  return str;
}
```

O(n)] Θ(n)

$= O(n^2)$

```java
public Pair(E e, T t){
  this.e = e;
  this.t = t;
}
```

Θ(1)

```java
public E getKey(){
  return this.e;
}
```

Θ(1)

```java
public T getValue(){
  return this.t;
}
```

Θ(1)

```java
@Override
public String toString(){
  return String.valueOf(this.e + " : " + this.t);
}
```

Θ(1)