GIT Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework 7 Report

Student Name
Samet NALBANT
Student Number
1801042614

1. SYSTEM REQUIREMENTS

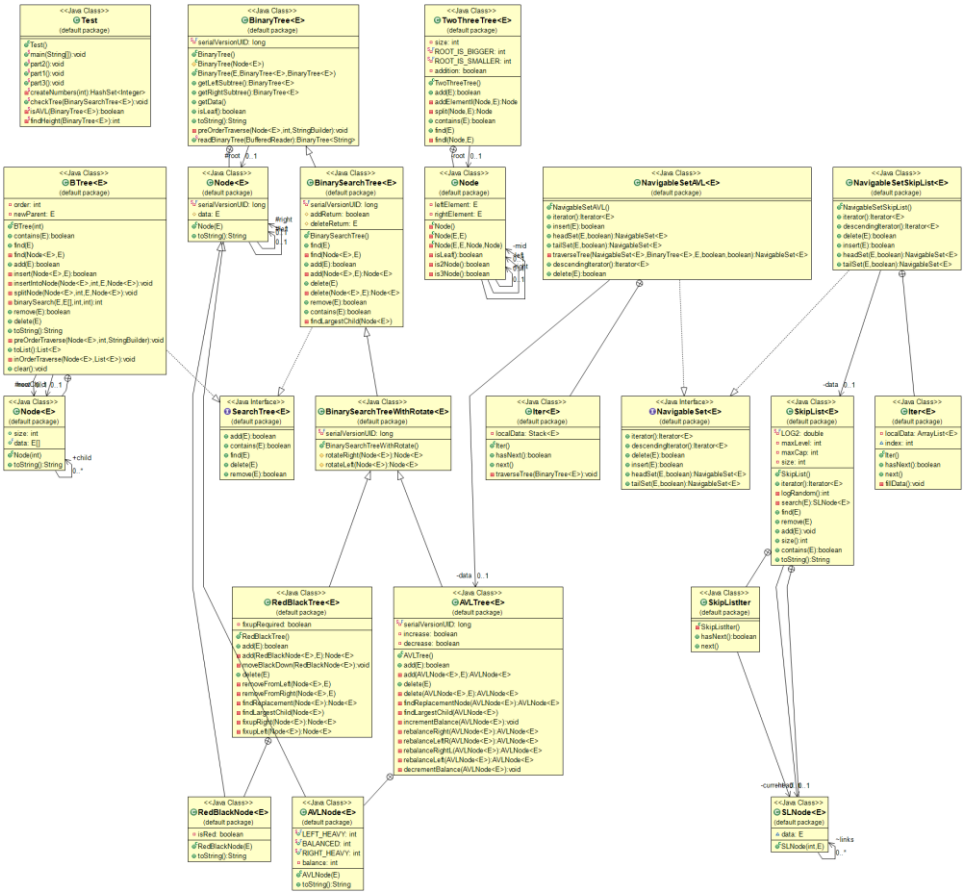
A) Functional Requirements

- NavigableSetAVL should insert value.
- NavigableSetAVL should have iterator.
- NavigableSetAVL should have headset and tailSet functions.
- NavigableSetSkipList should insert and delete value.
- NavigableSetSkipList should have descending iterator.
- Function to determine Binary Search Tree is an AVL Tree or not should implemented.

B) Non Functional Requirements

Hardware should be able to run at least JAVA SE13.

2. CLASS DIAGRAM



3. PROBLEM SOLUTION APPROACH

Navigable Set interface is created and the necessary functions which are described in homework are added. AVL Tree and Skip List implementation are taken from book. NavigableSetAVL class which has data field AVL Tree is created. Insert function implemented. To implement a Iterator, Tree is traversed recursively and the data itself is pushed to stack. So, Iterator's functions implemented. To implement headSet and tailSet functions all tree traversed recursively and the data added to temporary Navigable Set. After the all tree traversed Navigable Set which is created temporary is returned. NavigableSetSkipList class which has data field Skip List is created. Insert and delete functions and descending iterator are implemented. To determine whether Binary Search Tree is an AVL tree or not, recursive to find a height of tree function is created and this helper function used to determine whether Tree is an AVL or not. For the part3 necessary classes are implemented from book and driver function is implemented proper way

4. TEST CASES

Skip List is created and added 10 values from 1 to 10.

```
NavigableSetSkipList<Integer> data1 = new NavigableSetSkipList<>();

System.out.println("Integers inserting to skip list");
for(int i=0;i<10;i++) {
    data1.insert(i+1);
}
```

Value which exist in skip list is deleted.

```
System.out.println("Existing value is trying to deleting");
System.out.println(data1.delete(1));
```

Value which is not exist in skip list is deleted.

```
System.out.println("Non existing value is trying to deleting");
System.out.println(data1.delete(1));
```

Descending iterator is created and values are displayed.

```
System.out.println("Descending iterator is created and all values are iterating");
iter = data1.descendingIterator();
while(iter.hasNext()) {
    System.out.println(iter.next());
}
```

Iterator is accessed non existing value.

```
System.out.println("Iterator trying to access non existing value");
while(iter.hasNext()) {
    System.out.println(iter.next());
}
try {
      iter.next();
    }
    catch(Exception e) {
        System.out.println("Exception is catched!");
    }
```

AVL Tree is created and values are inserted from 10 to 20.

```
NavigableSetAVL<Integer> data2 =  new NavigableSetAVL<>();

System.out.println("Integers inserting to avl tree");
for(int i=10;i<20;i++) {
    data2.insert(i+1);
}
```

Iterator is created and values are displayed.

```
System.out.println("Iterator is created and all values are iterating");
iter = data2.iterator();
while(iter.hasNext()) {
    System.out.println(iter.next());
}
```

Iterator is accessed non existing value.

```
System.out.println("Iterator trying to access non existing value");
while(iter.hasNext()) {
    System.out.println(iter.next());
}
try {
      iter.next();
    }
    catch(Exception e) {
        System.out.println("Exception is catched!");
    }
```

Head Set is created according to 16 and inclusive. Values are displayed using iterator.

```
System.out.println("Head set function calling and the values will shown according to value 16 and inclusive");
temp = (NavigableSetAVL<Integer>) data2.headSet(16, true);
iter = temp.iterator();
while(iter.hasNext()) {
    System.out.println(iter.next());
}
```

Tail Set is created according to 18 and inclusive. Values are displayed using iterator.

```
System.out.println("tail set function calling and the values will shown according to value 18 and inclusive");
temp = (NavigableSetAVL<Integer>) data2.tailSet(18, true);
iter = temp.iterator();
while(iter.hasNext()) {
    System.out.println(iter.next());
}
```

Binary Search Tree is created and values added to tree and checked the tree is AVL Tree or not.

```
binarySearchTree.add(120);
binarySearchTree.add(47);
binarySearchTree.add(40);
binarySearchTree.add(25);
binarySearchTree.add(36);

checkTree(binarySearchTree);
```

AVL Tree is created and values added to tree and checked the tree is AVL Tree or not.

```
avlTree.add(120);
avlTree.add(47);
avlTree.add(40);
avlTree.add(25);
avlTree.add(36);

checkTree(avlTree);
```

Binary Search Tree is created and random unique accordingly 10000, 20000, 40000 and 80000 number and the time measured to add extra 100 random number for 10 times and the average time is calculated.

```
System.out.println("---Tests for binary search tree---");
for(int i=0;i<10;i++) {
    searchTree = new BinarySearchTree<>();
    numbers = createNumbers(10000);
    for(int j : numbers) {
        searchTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        searchTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to binary search tree which has 10000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");
```

```
total =0;
for(int i=0;i<10;i++) {
    searchTree = new BinarySearchTree<>();
    numbers = createNumbers(20000);
    for(int j : numbers) {
        searchTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        searchTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to binary search tree which has 20000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");
```

```
for(int i=0;i<10;i++) {
    searchTree = new BinarySearchTree<>();
    numbers = createNumbers(40000);
    for(int j : numbers) {
        searchTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        searchTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to binary search tree which has 40000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");
```

```
for(int i=0;i<10;i++) {
    searchTree = new BinarySearchTree<>();
    numbers = createNumbers(80000);
    for(int j : numbers) {
        searchTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        searchTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to binary search tree which has 80000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n\n\n");
```

Red Black Tree is created and random unique accordingly 10000, 20000, 40000 and 80000 number and the time measured to add extra 100 random number for 10 times and the average time is calculated.

```
total =0;
for(int i=0;i<10;i++) {
    redBlackTree = new RedBlackTree<>();
    numbers = createNumbers(10000);
    for(int j : numbers) {
        redBlackTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        redBlackTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to red black tree which has 10000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");
```

```
total =0;
for(int i=0;i<10;i++) {
    redBlackTree = new RedBlackTree<>();
    numbers = createNumbers(20000);
    for(int j : numbers) {
        redBlackTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        redBlackTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to red black tree which has 20000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");
```

```
total =0;
for(int i=0;i<10;i++) {
    redBlackTree = new RedBlackTree<>();
    numbers = createNumbers(40000);
    for(int j : numbers) {
        redBlackTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        redBlackTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to red black tree which has 40000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");
```

```
total =0;
for(int i=0;i<10;i++) {
    redBlackTree = new RedBlackTree<>();
    numbers = createNumbers(80000);
    for(int j : numbers) {
        redBlackTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        redBlackTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to red black tree which has 80000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n\n\n");
```

2-3 Tree is created and random unique accordingly 10000, 20000, 40000 and 80000 number and the time measured to add extra 100 random number for 10 times and the average time is calculated.

```
total =0;
for(int i=0;i<10;i++) {
    twoThreeTree = new TwoThreeTree<>();
    numbers = createNumbers(10000);
    for(int j : numbers) {
        twoThreeTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        twoThreeTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to 2-3 tree tree which has 10000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");
```

```
total =0;
for(int i=0;i<10;i++) {
    twoThreeTree = new TwoThreeTree<>();
    numbers = createNumbers(20000);
    for(int j : numbers) {
        twoThreeTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        twoThreeTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to 2-3 tree tree which has 20000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");
```

```
total =0;
for(int i=0;i<10;i++) {
    twoThreeTree = new TwoThreeTree<>();
    numbers = createNumbers(40000);
    for(int j : numbers) {
        twoThreeTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        twoThreeTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to 2-3 tree tree which has 40000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");
```

```
total =0;
for(int i=0;i<10;i++) {
    twoThreeTree = new TwoThreeTree<>();
    numbers = createNumbers(80000);
    for(int j : numbers) {
        twoThreeTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        twoThreeTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values to 2-3 tree tree which has 80000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n\n\n");
```

Skip List is created and random unique accordingly 10000, 20000, 40000 and 80000 number and the time measured to add extra 100 random number for 10 times and the average time is calculated.

```
total =0;
for(int i=0;i<10;i++) {
    skipList = new SkipList<>();
    numbers = createNumbers(10000);
    for(int j : numbers) {
        skipList.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        skipList.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values skip list which has 10000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");

 total =0;
 for(int i=0;i<10;i++) {
     skipList = new SkipList<>();
     numbers = createNumbers(20000);
     for(int j : numbers) {
         skipList.add(j);
     }
     numbers = createNumbers(100);
     start = System.nanoTime();
     for(int j : numbers) {
         skipList.add(j);
     }
     end = System.nanoTime();
     total += end-start;
 }
 System.out.print("Avarage time passed to add 100 values skip list which has 20000 values for 10 times: ");
 System.out.print(total/10+ " nano seconds\n");


System.out.print(total/10+ " nano seconds\n");
total =0;
for(int i=0;i<10;i++) {
    skipList = new SkipList<>();
    numbers = createNumbers(40000);
    for(int j : numbers) {
        skipList.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        skipList.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values skip list which has 40000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");

System.out.print(total/10+ " nano seconds\n");
total =0;
for(int i=0;i<10;i++) {
    skipList = new SkipList<>();
    numbers = createNumbers(80000);
    for(int j : numbers) {
        skipList.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        skipList.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values skip list which has 80000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");
```

B Tree is created and random unique accordingly 10000, 20000, 40000 and 80000 number and the time measured to add extra 100 random number for 10 times and the average time is calculated.

```
total =0;
for(int i=0;i<10;i++) {
    bTree = new BTree<>(5);
    numbers = createNumbers(10000);
    for(int j : numbers) {
        bTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        bTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values b tree  which has 10000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");


 total =0;
 for(int i=0;i<10;i++) {
     bTree = new BTree<>(5);
     numbers = createNumbers(20000);
     for(int j : numbers) {
         bTree.add(j);
     }
     numbers = createNumbers(100);
     start = System.nanoTime();
     for(int j : numbers) {
         bTree.add(j);
     }
     end = System.nanoTime();
     total += end-start;
 }
 System.out.print("Avarage time passed to add 100 values b tree which has 20000 values for 10 times: ");
 System.out.print(total/10+ " nano seconds\n");

total =0;
for(int i=0;i<10;i++) {
    bTree = new BTree<>(5);
    numbers = createNumbers(40000);
    for(int j : numbers) {
        bTree.add(j);
    }
    numbers = createNumbers(100);
    start = System.nanoTime();
    for(int j : numbers) {
        bTree.add(j);
    }
    end = System.nanoTime();
    total += end-start;
}
System.out.print("Avarage time passed to add 100 values b tree which has 40000 values for 10 times: ");
System.out.print(total/10+ " nano seconds\n");


 total =0;
 for(int i=0;i<10;i++) {
     bTree = new BTree<>(5);
     numbers = createNumbers(80000);
     for(int j : numbers) {
         bTree.add(j);
     }
     numbers = createNumbers(100);
     start = System.nanoTime();
     for(int j : numbers) {
         bTree.add(j);
     }
     end = System.nanoTime();
     total += end-start;
 }
 System.out.print("Avarage time passed to add 100 values b tree which has 80000 values for 10 times: ");
 System.out.print(total/10+ " nano seconds\n\n\n");
```

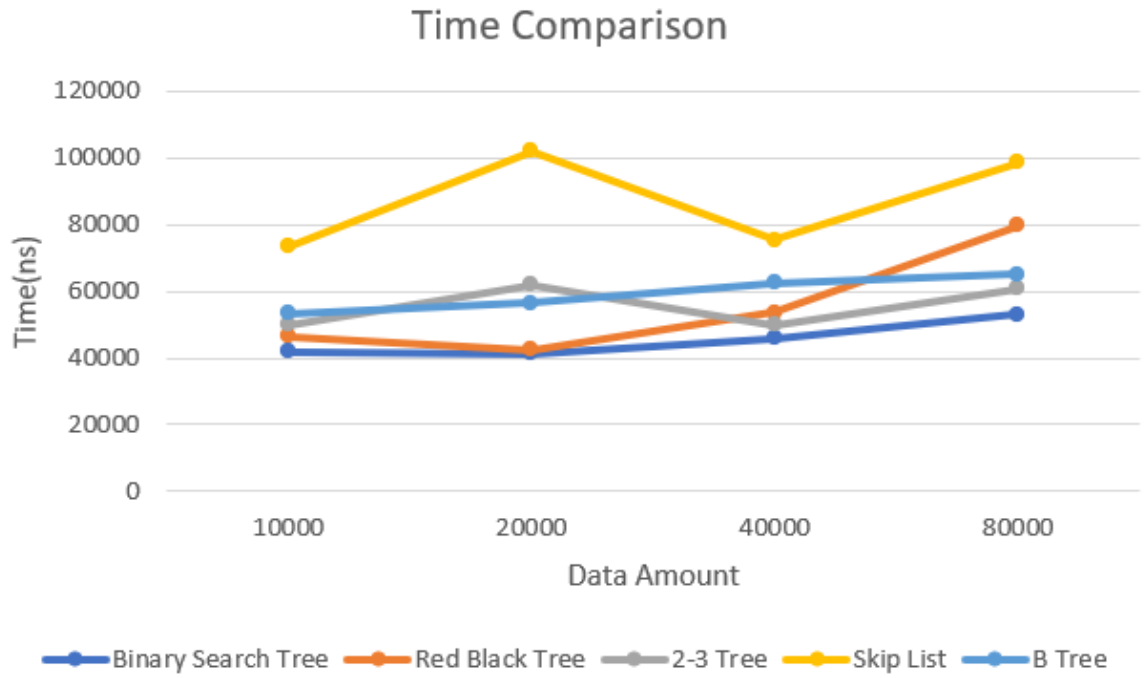
5. RUNNING AND RESULTS

```
--- Tests for part1 ---
Integers inserting to skip list
Existing value is trying to deleting
true
Non existing value is trying to deleting
false
Descending iterator is created and all values are iterating
10
9
8
7
6
5
4
3
2
Iterator trying to access non existing value
Exception is catched!
---------

Integers inserting to avl tree
Iterator is created and all values are iterating
20
19
17
15
16
18
13
11
12
14
Iterator trying to access non existing value
Exception is catched!
Head set function calling and the values will shown according to value 16 and inclusive
15
16
13
11
12
14
tail set function calling and the values will shown according to value 18 and inclusive
20
18
19
```

```
--- Test for part2 ---
Tree is AVL false
Tree is AVL true
```

```
---Tests for binary search tree---
Avarage time passed to add 100 values to binary search tree which has 10000 values for 10 times: 41760 nano seconds
Avarage time passed to add 100 values to binary search tree which has 20000 values for 10 times: 41210 nano seconds
Avarage time passed to add 100 values to binary search tree which has 40000 values for 10 times: 45740 nano seconds
Avarage time passed to add 100 values to binary search tree which has 80000 values for 10 times: 53170 nano seconds


---Tests for red black tree---
Avarage time passed to add 100 values to red black tree which has 10000 values for 10 times: 46310 nano seconds
Avarage time passed to add 100 values to red black tree which has 20000 values for 10 times: 42480 nano seconds
Avarage time passed to add 100 values to red black tree which has 40000 values for 10 times: 53620 nano seconds
Avarage time passed to add 100 values to red black tree which has 80000 values for 10 times: 79550 nano seconds


---Tests for 2-3 tree---
Avarage time passed to add 100 values to 2-3 tree tree which has 10000 values for 10 times: 49900 nano seconds
Avarage time passed to add 100 values to 2-3 tree tree which has 20000 values for 10 times: 62010 nano seconds
Avarage time passed to add 100 values to 2-3 tree tree which has 40000 values for 10 times: 49920 nano seconds
Avarage time passed to add 100 values to 2-3 tree tree which has 80000 values for 10 times: 60690 nano seconds


---Tests for skip list---
Avarage time passed to add 100 values skip list which has 10000 values for 10 times: 73480 nano seconds
Avarage time passed to add 100 values skip list which has 20000 values for 10 times: 102010 nano seconds
Avarage time passed to add 100 values skip list which has 40000 values for 10 times: 75740 nano seconds
Avarage time passed to add 100 values skip list which has 80000 values for 10 times: 98530 nano seconds


---Tests for btree list---
Avarage time passed to add 100 values b tree  which has 10000 values for 10 times: 53310 nano seconds
Avarage time passed to add 100 values b tree which has 20000 values for 10 times: 56490 nano seconds
Avarage time passed to add 100 values b tree which has 40000 values for 10 times: 62590 nano seconds
Avarage time passed to add 100 values b tree which has 80000 values for 10 times: 65040 nano seconds
```


Time Comparison
120000
100000
80000
60000
40000
20000
0
Time(ns)
Binary Search Tree
Red Black Tree
2-3 Tree
Skip List
B Tree
Data Types
10000
20000
40000
80000


Time Comparison
120000
100000
80000
60000
40000
20000
0
Time(ns)
10000
20000
40000
80000
Data Amount
Binary Search Tree
Red Black Tree
2-3 Tree
Skip List
B Tree