GIT Department of Computer Engineering

CSE 222/505 - Spring 2021
Homework 7 # Report

Okan Torun

1801042662

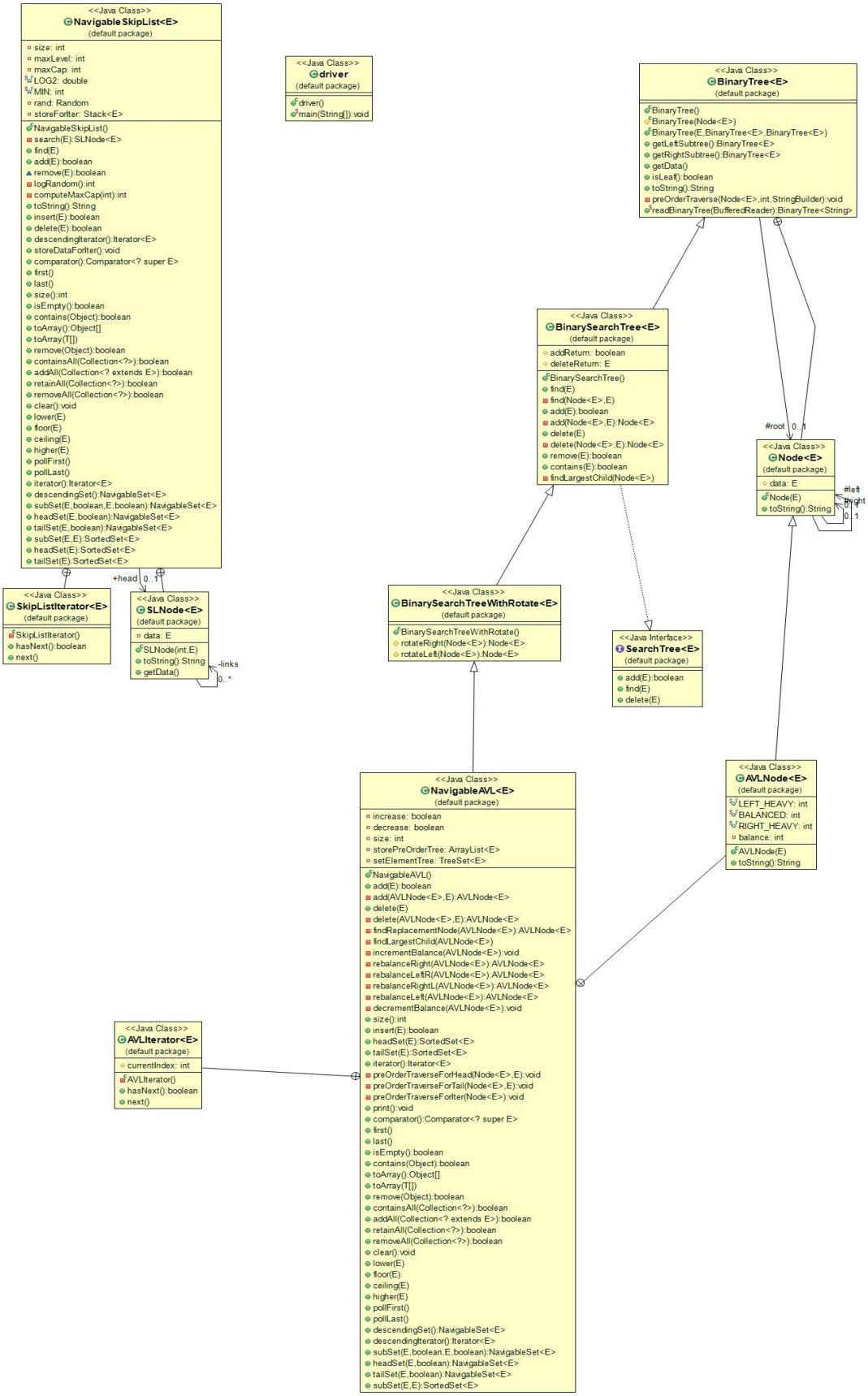
PART-1

1)SYSTEM REQUIREMENTS

There are two different classes that implement the NavigableSet interface. NavigableAVL and NavigableSkipList. The desired features in NavigableAVL are delete, insert and descandingIterator. The Delete method performs the deletion of elements from the SkipList. Likewise, the insert method performs the process of adding elements to the Skip List. The descandingIterator method iterates starting from the end of the SkipList.

The desired features in NavigableAVL are insert, iterator , headSet and tailSet. The Insert method performs the operation of adding elements to the AVL tree. The Iterator method provides pre-order traversal in the tree. The headSet method returns the list of elements smaller than the desired element in the tree. The tailSet method returns a list of elements larger than the desired element in the tree.

2) CLASS DIAGRAMS



3)PROBLEM SOLUTION APPROACH

The insert, delete and descendingIterator methods are requested for the NavigableSkipList. I called the add method of the SkipList to apply the insert method. I called the remove method of the SkipList for the delete method. For the descendingIterator, I first put all the elements in the SkipList into the stack structure and met the feature of the descendingIterator with the Last in First out principle.

For NavigableAVL, insert, iterator, tailSet and headSet properties are required. For the Insert method, I called the add function of the AVL tree. For the tailSet method, I created a TreeSet structure and collected the larger elements in the TreeSet structure and returned them.

For the headSet method, I listed the elements smaller than the sent element and returned them in the TreeSet. For the Iterator method, I put the elements in the AVL tree as pre-order in the Arraylist structure I created. I iterator pre-order over ArrayList.

4)TEST CASES

-I initialize the navigableAVL and navigableSET classes.

```
NavigableAVL<Integer> avlNavigable = new NavigableAVL<Integer>();
NavigableSkipList<Integer> skipListNavigable=new NavigableSkipList<>();
```

-Inserting to AVL navigable

```
avlNavigable.insert(8);
```

-print AVL navigable

```
avlNavigable.print();
```

-Initialize avlNavigable Iterator

```
Iterator<Integer> iter=avlNavigable.iterator();
```

-Itereator next operation

```
System.out.println(iter.next());
```

-Iterator hasNext operation

```
System.out.println("Has Next: "+iter.hasNext());
```

-I print the elements in the headSet list by traversing with forEach.

```
for(Integer temp:avlNavigable.headSet(7))
    System.out.println(temp);
```

-I print the elements in the tailSet list by traversing with forEach.

```
for(Integer temp:avlNavigable.tailSet(7))
    System.out.println(temp);
```

-Inserting to SkipList navigable

```
skipListNavigable.insert(8);
```

-Print all of elements SkipList

```
System.out.println(skipListNavigable.toString());
```

-Deleting element from SkipList

```
skipListNavigable.delete(8);
```

-Initialize descendingIterator for SkipList

```
Iterator<Integer>iterSkipList=skipListNavigable.descendingIterator();
```

-Itereator next operation

```
System.out.println(iterSkipList.next());
```

-Itereator hasNext operation

```
System.out.println("Has Next: "+iterSkipList.hasNext());
```

5)RUNNİNG AND RESULTS

```
---------------------------------Navigable AVL---------------------------------
-------------Adding elements to AVL Tree and Print--------------
-1: 8
  1: 3
    0: 1
      null
      null
    0: 6
      0: 4
        null
        null
      0: 7
        null
        null
  0: 13
    0: 10
      null
      null
    0: 14
      null
      null
```

```
-------------Pre-order Traverse with Iterator-------------
8
Has Next: true
3
1
6
4
7
13
10
14
Has Next: false
------------Test HeadSet for 7-------------
1
3
4
6
------------Test TailSet for 7--------------
7
8
10
13
14
```

```
--------------------------------Navigable SkipList--------------------------------


------------Adding elements to SkipList and Print-------------

Head: 4 --> 1 |1| --> 3 |1| --> 4 |2| --> 6 |2| --> 7 |3| --> 8 |1| --> 10 |1| --> 13 |4| --> 14 |1|


------------Removing 8 from SkipList and Print-------------

Head: 4 --> 1 |1| --> 3 |1| --> 4 |2| --> 6 |2| --> 7 |3| --> 10 |1| --> 13 |4| --> 14 |1|
```

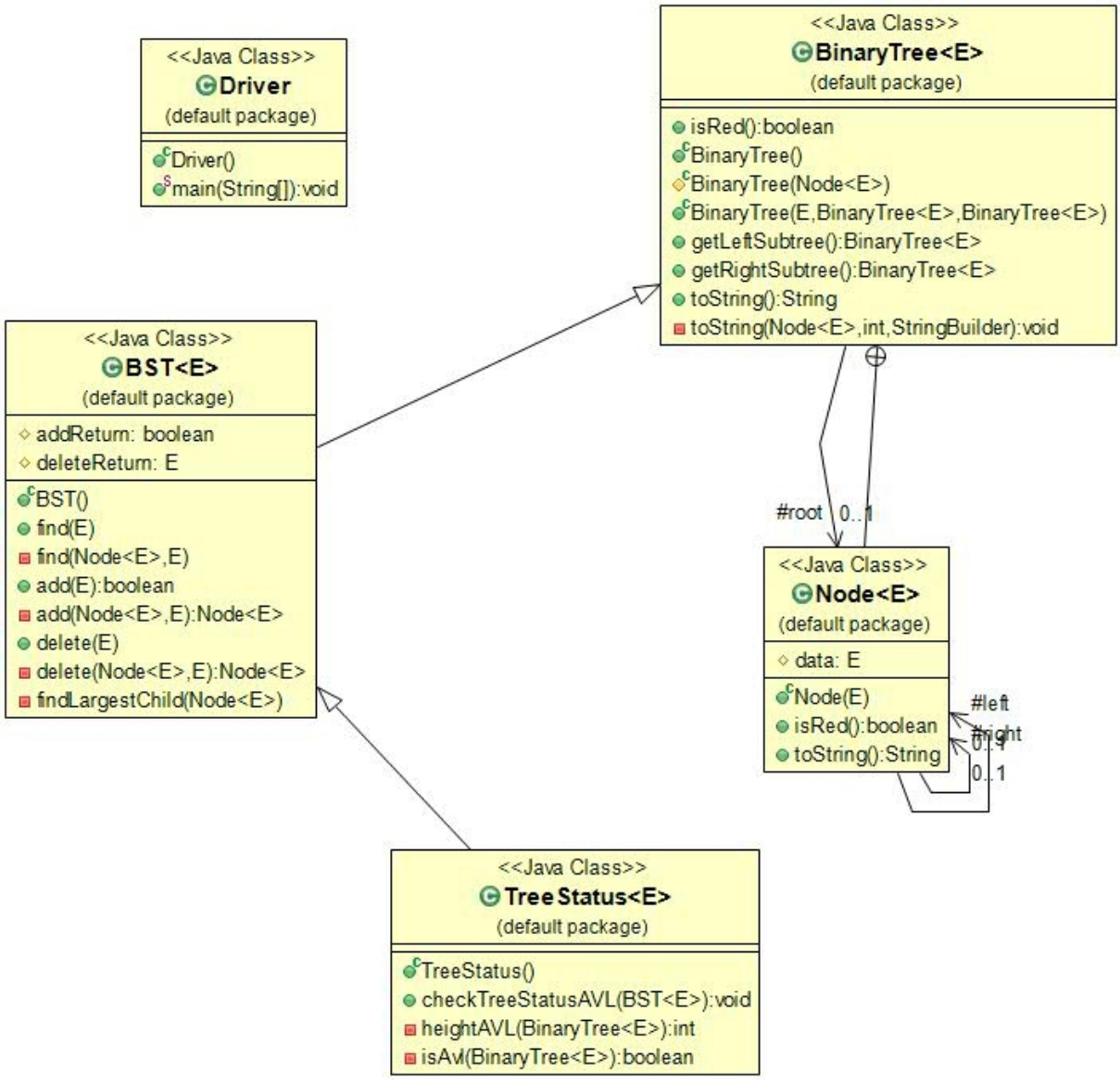
```
-------------Descending Iterator-------------
14
13
Has Next: true
10
7
6
4
3
1
Has Next: false
```

PART 2

1)SYSTEM REQUIREMENTS

The functions I wrote to understand whether the binarySearchTree sent as a parameter is AVL are heightAVL vs isAVL.We learn whether the tree is an AVL by calling the checkTreeStatusAvl method with the BinarySearchTree you want to check.

2)Class Diagrams



3)PROBLEM SOLUTION APPROACH

While checking whether the Binary Search Tree is an AVL tree or not, the branches need to be navigated one by one and compared with their absolute values. In this, I used the isHeight method and made step-by-step comparisons with the method that returns the length of the subTree with more depth. Finally, if the comparing difference is greater than 1, it is false. Returns. Otherwise, it continues until the null node.

4)TEST CASES

-I initialize the BST data structure and my control class

```
BST<Integer>myAVL=new BST<Integer>();

TreeStatus<Integer>status=new TreeStatus<>();
```

-I am adding element to BST structure

```
myAVL.add(15);
```

-I list elements in BST

```
System.out.println(myAVL.toString());
```

-Checking if tree is AVL

```
status.checkTreeStatusAVL(myAVL);
```

5)RUNNİNG AND RESULTS

```
15
 10
  5
   null
   null
  null
 20
  null
  30
   null
   null

Tree is AVL
```

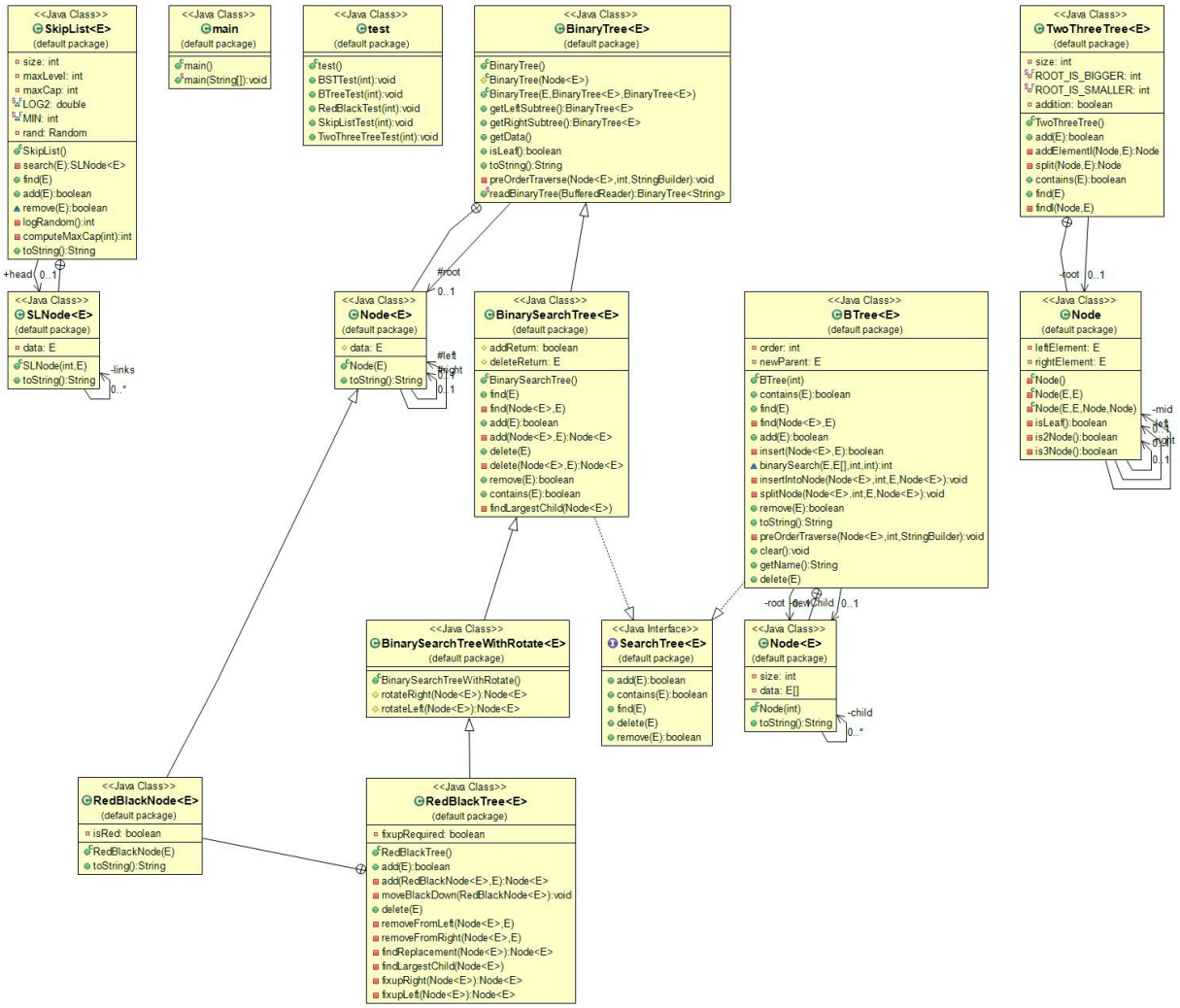
```
15
 10
  5
   null
   null
  null
 20
  null
  30
   null
   40
    null
    null

Tree is not AVL
```

PART 3

1)SYSTEM REQUIREMENTS

I made the implementationos of Binary Serach Tree, Red-Black tree, 2-3 tree, B-tree and Skip-List. And I measured their efficiency by adding different sizes in the test class.

2) CLASS DIAGRAMS



3)PROBLEM SOLUTION APPROACH

I made the necessary additions to the Binary Serach Tree, Red-Black Tree, 2-3 trees, B tree and Jump List at the desired numbers and intervals. I paid attention to the uniqueness of the random numbers I added so that it would work better. And after I added the numbers of the desired size, I added an extra 100 numbers and measured the elapsed times. Finally, I calculated the averages and plotted them.

4)TEST CASES

-I created randomGenerateArray and put a set of unique numbers in it. Then I created the necessary data structure and placed these numbers on it until it reached the desired size. Then I added another 100 numbers and calculated the elapsed
time.

```
public void BSTTest(int size) {
    Random rand = new Random();
    long startTime = 0, endTime = 0, time = 0;
    BinarySearchTree<Integer> BST=new BinarySearchTree<>();
    int[] randomGenerateNumbers = rand.ints(0, 2000000).distinct().limit(size+100).toArray();
    long avgTime=0;
    System.out.println("-------Adding "+100+" randomly numbers to "+size+" BST--------");
    for(int i=0;i<10;i++) {
        BST=new BinarySearchTree();
        for(int j=0;j<size;j++){
            BST.add(randomGenerateNumbers[j]);
        }
        startTime = System.nanoTime();
        for(int k=size;k<size+100;k++) {
            BST.add(randomGenerateNumbers[k]);
        }
        endTime = System.nanoTime();
        time = endTime - startTime;
        avgTime+=time;
       // System.out.println("Test"+(i+1)+": "+time);
    }
    System.out.println("Avg Time: "+avgTime/10);
    avgTime=0;

}
```

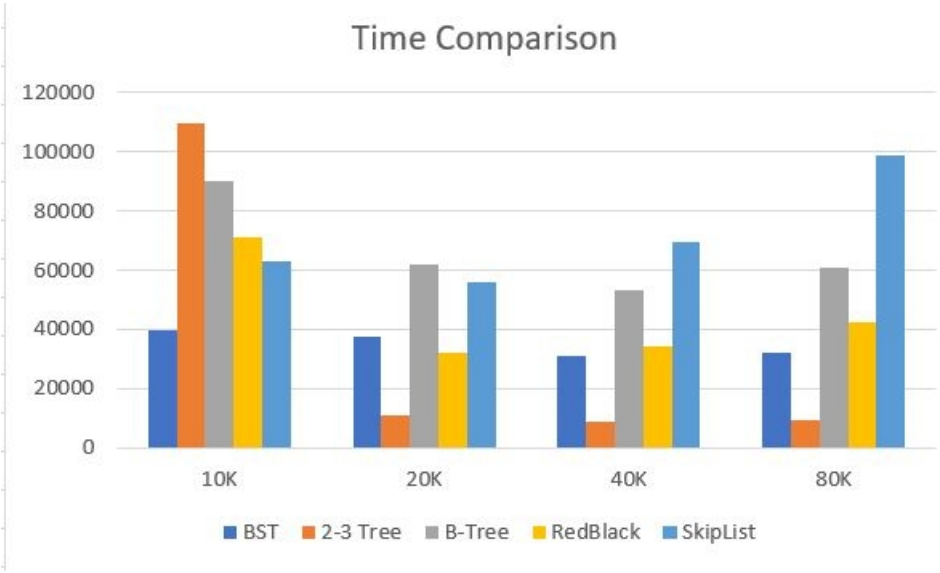
```java
public void BTreeTest(int size) {
    Random rand = new Random();
    long startTime = 0, endTime = 0, time = 0;
    BTree<Integer> BTreeObj=new BTree<>(3);
    int[] randomGenerateNumbers = rand.ints(0, 2000000).distinct().limit(size+100).toArray();
    long avgTime=0;
    System.out.println("-------Adding "+100+" randomly numbers to "+size+" BTree--------");
    for(int i=0;i<10;i++) {
        BTreeObj=new BTree<>(3);
        for(int j=0;j<size;j++){
            BTreeObj.add(randomGenerateNumbers[j]);
        }
        startTime = System.nanoTime();
        for(int k=size;k<size+100;k++) {
            BTreeObj.add(randomGenerateNumbers[k]);
        }
        endTime = System.nanoTime();
        time = endTime - startTime;
        avgTime+=time;
        //System.out.println("Test"+(i+1)+": "+time);
    }
    System.out.println("Avg Time: "+avgTime/10);
    avgTime=0;

}
```

```java
public void RedBlackTest(int size) {
    Random rand = new Random();
    long startTime = 0, endTime = 0, time = 0;
    RedBlackTree<Integer> RBtree=new RedBlackTree<>();
    int[] randomGenerateNumbers = rand.ints(0, 2000000).distinct().limit(size+100).toArray();
    long avgTime=0;
    System.out.println("-------Adding "+100+" randomly numbers to "+size+" RedBlackTree--------");
    for(int i=0;i<10;i++) {
        RBtree=new RedBlackTree<>();
        for(int j=0;j<size;j++){
            RBtree.add(randomGenerateNumbers[j]);
        }
        startTime = System.nanoTime();
        for(int k=size;k<size+100;k++) {
            RBtree.add(randomGenerateNumbers[k]);
        }
        endTime = System.nanoTime();
        time = endTime - startTime;
        avgTime+=time;
        //System.out.println("Test"+(i+1)+": "+time);
    }
    System.out.println("Avg Time: "+avgTime/10);
    avgTime=0;

}
```

```java
public void SkipListTest(int size) {
    Random rand = new Random();
    long startTime = 0, endTime = 0, time = 0;
    SkipList<Integer> skipList=new SkipList<>();
    int[] randomGenerateNumbers = rand.ints(0, 2000000).distinct().limit(size+100).toArray();
    long avgTime=0;
    System.out.println("-------Adding "+100+" randomly numbers to "+size+" SkipList--------");
    for(int i=0;i<10;i++) {
        skipList=new SkipList<>();
        for(int j=0;j<size;j++){
            skipList.add(randomGenerateNumbers[j]);
        }
        startTime = System.nanoTime();
        for(int k=size;k<size+100;k++) {
            skipList.add(randomGenerateNumbers[k]);
        }
        endTime = System.nanoTime();
        time = endTime - startTime;
        avgTime+=time;
      //  System.out.println("Test"+(i+1)+": "+time);
    }
    System.out.println("Avg Time: "+avgTime/10);
    avgTime=0;

}
```

```java
public void TwoThreeTreeTest(int size) {
    Random rand = new Random();
    long startTime = 0, endTime = 0, time = 0;
    TwoThreeTree<Integer> twoThreeTree=new TwoThreeTree<>();
    int[] randomGenerateNumbers = rand.ints(0, 2000000).distinct().limit(size+100).toArray();
    long avgTime=0;
    System.out.println("-------Adding "+100+" randomly numbers to "+size+" TwoThreeTree--------");
    for(int i=0;i<10;i++) {
        twoThreeTree=new TwoThreeTree<>();
        for(int j=0;j<size;j++){
            twoThreeTree.add(randomGenerateNumbers[i]);
        }
        startTime = System.nanoTime();
        for(int k=size;k<size+100;k++) {
            twoThreeTree.add(randomGenerateNumbers[k]);
        }
        endTime = System.nanoTime();
        time = endTime - startTime;
        avgTime+=time;
        //System.out.println("Test"+(i+1)+": "+time);
    }
    System.out.println("Avg Time: "+avgTime/10);
    avgTime=0;

}
```

5)RUNNİNG AND RESULTS




```
----------------------For 10K------------------------
-------Adding 100 randomly numbers to 10000 BST--------
Avg Time: 39770
-------Adding 100 randomly numbers to 10000 BTree--------
Avg Time: 90190
-------Adding 100 randomly numbers to 10000 RedBlackTree--------
Avg Time: 71320
-------Adding 100 randomly numbers to 10000 SkipList--------
Avg Time: 63190
-------Adding 100 randomly numbers to 10000 TwoThreeTree--------
Avg Time: 109560
```

```
----------------------For 20K------------------------
-------Adding 100 randomly numbers to 20000 BST--------
Avg Time: 37620
-------Adding 100 randomly numbers to 20000 BTree--------
Avg Time: 61730
-------Adding 100 randomly numbers to 20000 RedBlackTree--------
Avg Time: 32060
-------Adding 100 randomly numbers to 20000 SkipList--------
Avg Time: 55690
-------Adding 100 randomly numbers to 20000 TwoThreeTree--------
Avg Time: 11150
```

```
----------------------For 40K------------------------
-------Adding 100 randomly numbers to 40000 BST--------
Avg Time: 31150
-------Adding 100 randomly numbers to 40000 BTree--------
Avg Time: 53320
-------Adding 100 randomly numbers to 40000 RedBlackTree--------
Avg Time: 34240
-------Adding 100 randomly numbers to 40000 SkipList--------
Avg Time: 69730
-------Adding 100 randomly numbers to 40000 TwoThreeTree--------
Avg Time: 9000



----------------------For 80K------------------------
-------Adding 100 randomly numbers to 80000 BST--------
Avg Time: 32320
-------Adding 100 randomly numbers to 80000 BTree--------
Avg Time: 61000
-------Adding 100 randomly numbers to 80000 RedBlackTree--------
Avg Time: 42310
-------Adding 100 randomly numbers to 80000 SkipList--------
Avg Time: 98840
-------Adding 100 randomly numbers to 80000 TwoThreeTree--------
Avg Time: 9260
```