



**T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY of ENGINEERING
DEPARTMENT of COMPUTER ENGINEERING**

SELF BALANCED TREES

CSE 222 DATA STRUCTURES AND ALGORITHMS HOMEWORK 7 REPORT

STUDENT

**Şeyda Nur DEMİR
12 10 44 042**

LECTURER

Prof. Dr. Fatih Erdoğan SEVİLGİN

TEACHING ASSISTANT

**Başak KARAKAŞ
Mehmet Burak KOCA**

KOCAELİ, 2021

1.DESCRPTION

This homework has 3 parts.

Part 1

Provide two partial implementations of NavigableSet interface:

- Implement following methods of NavigableSet interface using skip-list
 - insert
 - delete
 - descendingIterator
- Implement following methods of NavigableSet interface using AVL tree
 - insert
 - iterator
 - headSet
 - tailSet

Part 2

Write method that takes a BinarySearchTree and checks whether the tree is

- an AVL tree
- a Red-Black tree (Suppose BinarySearchTree has a method isRed which returns a boolean value which indicates whether the root node is a red node or not).

Part 3

Compare insertion performance of the following data structures :

- Binary search tree implementation in the book
- Red-Black tree implementation in the book
- 2-3 tree implementation in the book
- B-tree implementation in the book
- Skip list implementation in the book

For each data structure ;

- Construct an instance of each data structure by inserting a collection of randomly generated (non-repeating) numbers. Perform this operation 10 times for 10.000, 20.000, 40.000 and 80.000 random numbers (10 times for each). So, you will have 10 instances for each data structure and each different size (i.e., 200 instances in total).
- Compare the experimental run-time performance of the insertion operation for the data structures above as follows:
 - Insert 100 extra random numbers into the structures you built and measure the running time
 - Calculate the average running time for each data structure and problem size (i.e., number of elements in data structure)
 - Draw a graph for running-time vs problem size.
 - Compare the running times and their increase rate.

Restrictions

- Can be only one main class in project
- Don't use any other third part library

General Rules

- For any question firstly use course news forum in Moodle, and then the contact TA.
- You can submit assignment one day late and will be evaluated over sixty percent (%60).

Technical Rules

- You must write a driver function that demonstrates all possible actions in your homework. For example, if you are asked to implement an array list and perform an iterative search on the list then. The driver function should run when the code file is executed.
- Implement clean code standards in your code ;
 - Classes, methods and variables names must be meaningful and related with the functionality.
 - Your functions and classes must be simple, general, reusable and focus on one topic.
 - Use standard java code name conventions.

Report Rules

- Add all javadoc documentations for classes, methods, variables ...etc. All explanation must be meaningful and understandable.
- You should submit your homework code, Javadoc and report to Moodle in a "studentid_hw7.tar.gz" file.
- Use the given homework format including selected parts from the table below:
 - Problem solutions approach
 - Test cases
 - Running command and results

Grading

- No OOP design: -100
- No interface: -95
- No method overriding: -95
- No error handling: -50
- No inheritance: -95
- No polymorphism: -95
- No javadoc documentation: -50
- No report: -90
- Disobey restrictions: -100
- Cheating: -200
- Your solution is evaluated over 100 as your performance.

2.REPORT

I detailed here what I did in my homework.

I detailed Part 1, Part 2, Part 3 separately.

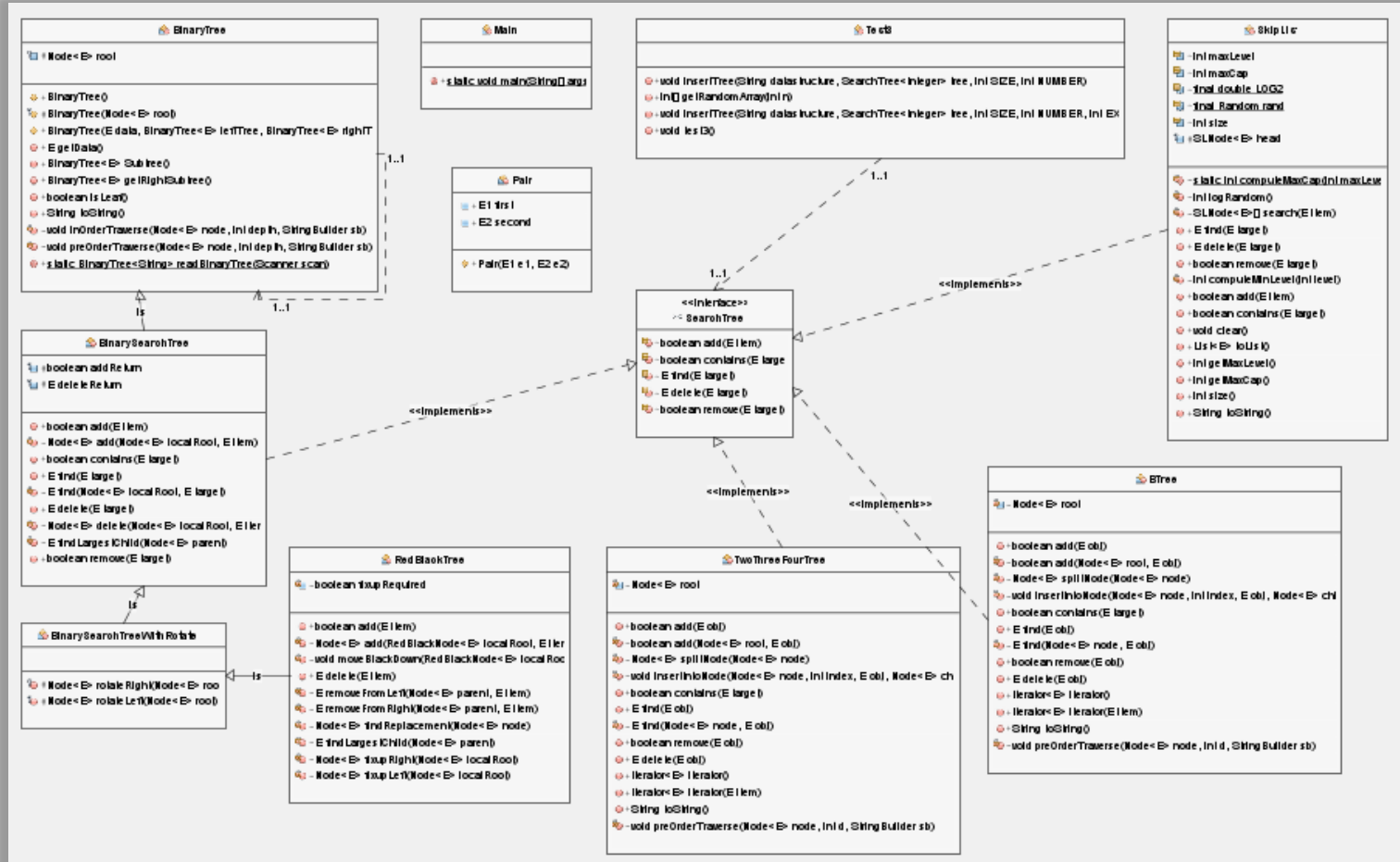
Part 3 : 2.1.Problem Solutions Approach

Note : I googled the way you said in the report to problem solving approach, but I could not any useful article, what I found was either paid or long. After all I found a useful post from medium. I prepared this part according to this post. I hope I got it right.

- **Clearly understanding and/or defining the problem :**
I understood and defined the problem clearly.
 - We should **implement 5 self balanced tree**.
 - And **compare their running time performances**.
- **Breaking down large problems into smaller problems :**
I broke down large problem **implementation and running time comparisons**, to small problems **“data structures implementations”**, **“writing main method and test of the methods”**, get them together and print results.
 - We have a large problem, implementations and comparisons.
 - I have now small problems, **“Binary Search Tree implementation”**, **“Red Black Tree implementation”**, **“Two Three Tree implementation”**, **“B Tree implementation”**, **“Skip List implementation”**, **“Main Class and main method”**, **“Test Class and test methods”**, **“calculating running times and drawing graph”** modules
- **Solving the problem at an abstract level first :**
I solved the problem at an abstract level first.
 - I thought a lot about the problem.
 - I scribbled something about this subject in the ledger.
 - Something started to take shape in my head.
 - I used my knowledge of data structures, and it is.
- **Using notes and pseudo-code :**
I noted what I thought and wrote permanently pseudo-codes mixed with java codes.
 - If I understand or find something new, I noted.
 - I wrote pseudo-codes mixed with java codes, not clear.
- **Running code early and often :**
I wrote real codes and run them often.
 - I turned my pseudo-codes into real java codes.
 - I coded them in ide and run them often.

Part 3 : 2.2.Uml Diagram

You can also see uml diagram in png format in "Report" directory.



Part 3 : 2.3.Test Cases

Testing Requirements

Test Case	Pres / Posts	Done
Construct Data Structures	<ul style="list-style-type: none">• Should have implementations of needed data structures	Done
Insert 10000 Elements Insert 20000 Elements Insert 40000 Elements Insert 80000 Elements	<ul style="list-style-type: none">• Should have constructed instances of data structures• Should have constructed instances as array with given size	Done
Insert Extra 100 Elements	<ul style="list-style-type: none">• Should have constructed instances of data structures with given sizes• Should have an array with 100 random numbers to insert to the data structures	Done

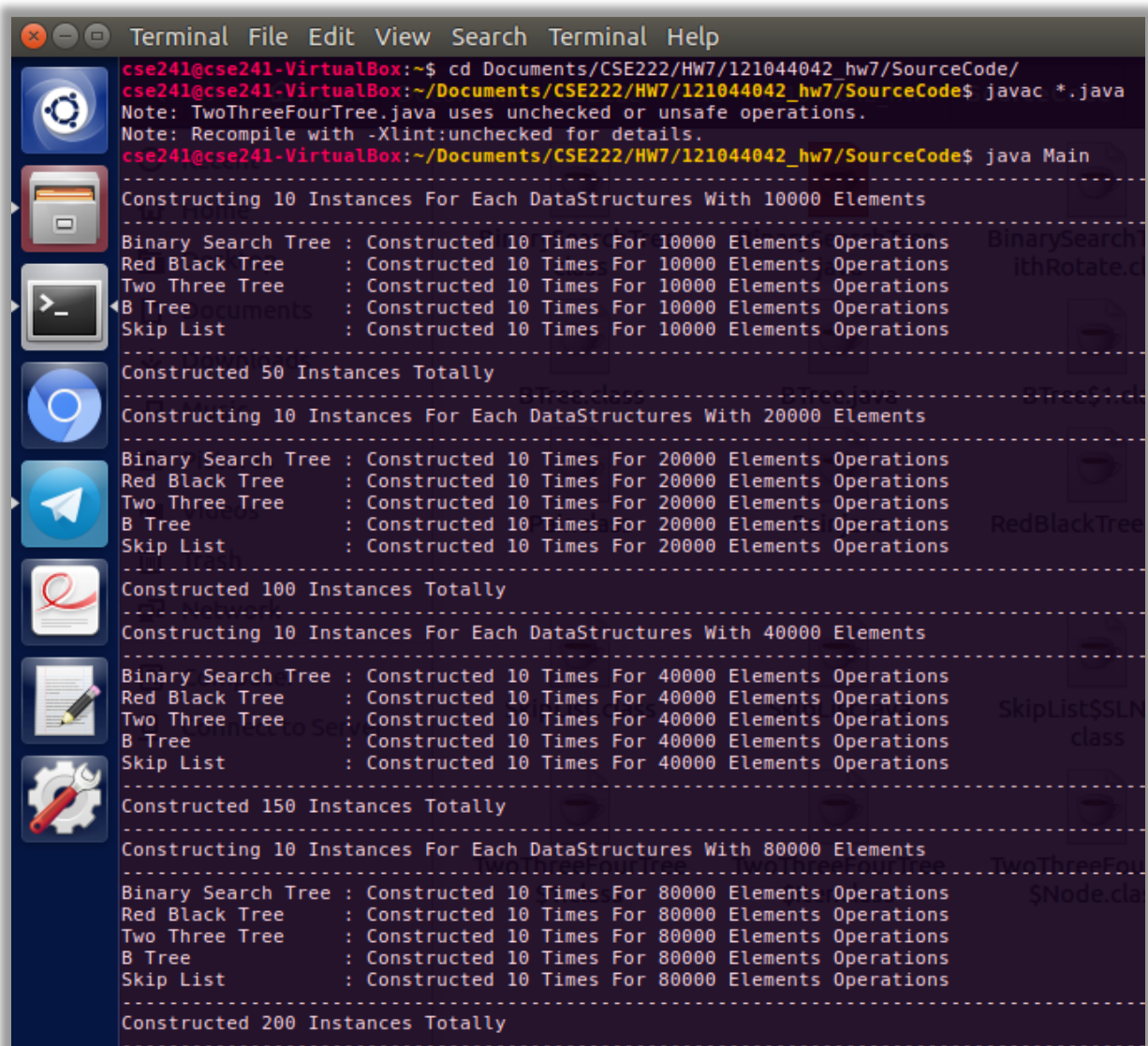
Part 3 : 2.4. Running Commands and Results

Compile and Run Commands, Testing Steps

Usage of my program :

1. Compile program with “**javac *.java**” command
2. Run program with “**java Main**” command

Simply Follow This Screenshot



```
Terminal File Edit View Search Terminal Help
cse241@cse241-VirtualBox:~$ cd Documents/CSE222/HW7/121044042_hw7/SourceCode/
cse241@cse241-VirtualBox:~/Documents/CSE222/HW7/121044042_hw7/SourceCode$ javac *.java
Note: TwoThreeFourTree.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
cse241@cse241-VirtualBox:~/Documents/CSE222/HW7/121044042_hw7/SourceCode$ java Main

Constructing 10 Instances For Each DataStructures With 10000 Elements
-----
Binary Search Tree : Constructed 10 Times For 10000 Elements Operations
Red Black Tree : Constructed 10 Times For 10000 Elements Operations
Two Three Tree : Constructed 10 Times For 10000 Elements Operations
B Tree : Constructed 10 Times For 10000 Elements Operations
Skip List : Constructed 10 Times For 10000 Elements Operations
-----
Constructed 50 Instances Totally

Constructing 10 Instances For Each DataStructures With 20000 Elements
-----
Binary Search Tree : Constructed 10 Times For 20000 Elements Operations
Red Black Tree : Constructed 10 Times For 20000 Elements Operations
Two Three Tree : Constructed 10 Times For 20000 Elements Operations
B Tree : Constructed 10 Times For 20000 Elements Operations
Skip List : Constructed 10 Times For 20000 Elements Operations
-----
Constructed 100 Instances Totally

Constructing 10 Instances For Each DataStructures With 40000 Elements
-----
Binary Search Tree : Constructed 10 Times For 40000 Elements Operations
Red Black Tree : Constructed 10 Times For 40000 Elements Operations
Two Three Tree : Constructed 10 Times For 40000 Elements Operations
B Tree : Constructed 10 Times For 40000 Elements Operations
Skip List : Constructed 10 Times For 40000 Elements Operations
-----
Constructed 150 Instances Totally

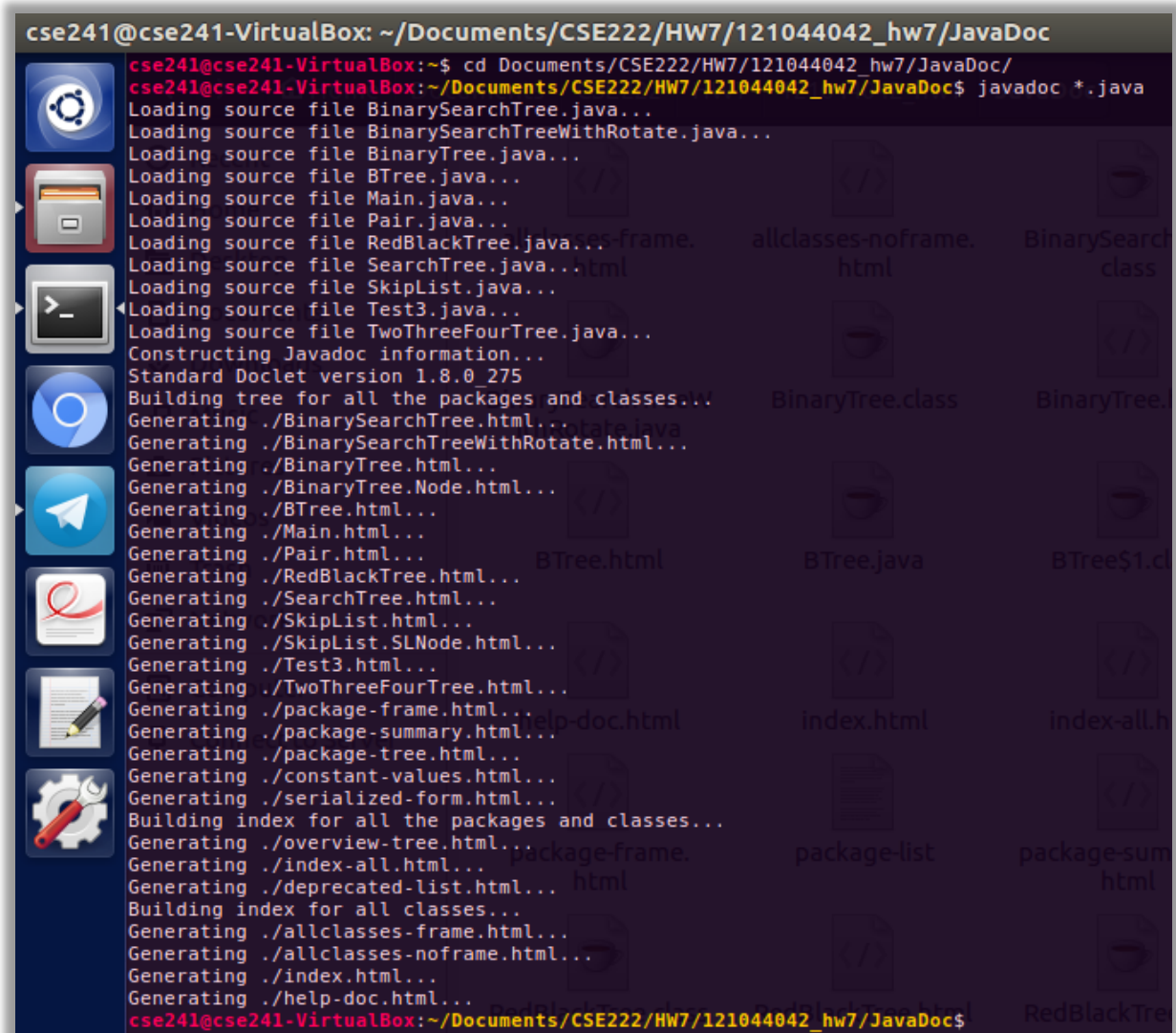
Constructing 10 Instances For Each DataStructures With 80000 Elements
-----
Binary Search Tree : Constructed 10 Times For 80000 Elements Operations
Red Black Tree : Constructed 10 Times For 80000 Elements Operations
Two Three Tree : Constructed 10 Times For 80000 Elements Operations
B Tree : Constructed 10 Times For 80000 Elements Operations
Skip List : Constructed 10 Times For 80000 Elements Operations
-----
Constructed 200 Instances Totally
```


Javadoc Command

Usage of my program's javadoc code :

3. Compile program with "javadoc *.java" command

Simply Follow This Screenshot



```
cse241@cse241-VirtualBox: ~/Documents/CSE222/HW7/121044042_hw7/JavaDoc
cse241@cse241-VirtualBox:~$ cd Documents/CSE222/HW7/121044042_hw7/JavaDoc/
cse241@cse241-VirtualBox:~/Documents/CSE222/HW7/121044042_hw7/JavaDoc$ javadoc *.java
Loading source file BinarySearchTree.java...
Loading source file BinarySearchTreeWithRotate.java...
Loading source file BinaryTree.java...
Loading source file BTree.java...
Loading source file Main.java...
Loading source file Pair.java...
Loading source file RedBlackTree.java...
Loading source file SearchTree.java...
Loading source file SkipList.java...
Loading source file Test3.java...
Loading source file TwoThreeFourTree.java...
Constructing Javadoc information...
Standard Doclet version 1.8.0_275
Building tree for all the packages and classes...
Generating ./BinarySearchTree.html...
Generating ./BinarySearchTreeWithRotate.html...
Generating ./BinaryTree.html...
Generating ./BinaryTree.Node.html...
Generating ./BTree.html...
Generating ./Main.html...
Generating ./Pair.html...
Generating ./RedBlackTree.html...
Generating ./SearchTree.html...
Generating ./SkipList.html...
Generating ./SkipList.SLNode.html...
Generating ./Test3.html...
Generating ./TwoThreeFourTree.html...
Generating ./package-frame.html...
Generating ./package-summary.html...
Generating ./package-tree.html...
Generating ./constant-values.html...
Generating ./serialized-form.html...
Building index for all the packages and classes...
Generating ./overview-tree.html...
Generating ./index-all.html...
Generating ./deprecated-list.html...
Building index for all classes...
Generating ./allclasses-frame.html...
Generating ./allclasses-noframe.html...
Generating ./index.html...
Generating ./help-doc.html...
```

Commands and Results with Screenshots

Test Results : Construct Data Structures

```
-----
Constructing 10 Instances For Each DataStructures With 10000 Elements
-----
Binary Search Tree : Constructed 10 Times For 10000 Elements Operations
Red Black Tree      : Constructed 10 Times For 10000 Elements Operations
Two Three Tree      : Constructed 10 Times For 10000 Elements Operations
B Tree              : Constructed 10 Times For 10000 Elements Operations
Skip List           : Constructed 10 Times For 10000 Elements Operations
-----
Constructed 50 Instances Totally
-----
Constructing 10 Instances For Each DataStructures With 20000 Elements
-----
Binary Search Tree : Constructed 10 Times For 20000 Elements Operations
Red Black Tree      : Constructed 10 Times For 20000 Elements Operations
Two Three Tree      : Constructed 10 Times For 20000 Elements Operations
B Tree              : Constructed 10 Times For 20000 Elements Operations
Skip List           : Constructed 10 Times For 20000 Elements Operations
-----
Constructed 100 Instances Totally
-----
Constructing 10 Instances For Each DataStructures With 40000 Elements
-----
Binary Search Tree : Constructed 10 Times For 40000 Elements Operations
Red Black Tree      : Constructed 10 Times For 40000 Elements Operations
Two Three Tree      : Constructed 10 Times For 40000 Elements Operations
B Tree              : Constructed 10 Times For 40000 Elements Operations
Skip List           : Constructed 10 Times For 40000 Elements Operations
-----
Constructed 150 Instances Totally
-----
Constructing 10 Instances For Each DataStructures With 80000 Elements
-----
Binary Search Tree : Constructed 10 Times For 80000 Elements Operations
Red Black Tree      : Constructed 10 Times For 80000 Elements Operations
Two Three Tree      : Constructed 10 Times For 80000 Elements Operations
B Tree              : Constructed 10 Times For 80000 Elements Operations
Skip List           : Constructed 10 Times For 80000 Elements Operations
-----
Constructed 200 Instances Totally
-----
```

Test Results : Insert Elements

Inserting 10000 Random Elements To The 10 Instances of Each DataStructures With 10000 Elements

Binary Search Tree : Inserted 10000 Elements To The Tree
Red Black Tree : Inserted 10000 Elements To The Tree
Two Three Tree : Inserted 10000 Elements To The Tree
B Tree : Inserted 10000 Elements To The Tree
Skip List : Inserted 10000 Elements To The Tree

Inserted 10000 Random Elements To The 10 Instances of DataStructures With 10000 Elements

Inserting 20000 Random Elements To The 10 Instances of DataStructures With 20000 Elements

Binary Search Tree : Inserted 20000 Elements To The Tree
Red Black Tree : Inserted 20000 Elements To The Tree
Two Three Tree : Inserted 20000 Elements To The Tree
B Tree : Inserted 20000 Elements To The Tree
Skip List : Inserted 20000 Elements To The Tree

Inserted 20000 Random Elements To The 10 Instances of DataStructures With 20000 Elements

Inserting 40000 Random Elements To The 10 Instances of DataStructures With 40000 Elements

Binary Search Tree : Inserted 40000 Elements To The Tree
Red Black Tree : Inserted 40000 Elements To The Tree
Two Three Tree : Inserted 40000 Elements To The Tree
B Tree : Inserted 40000 Elements To The Tree
Skip List : Inserted 40000 Elements To The Tree

Inserted 40000 Random Elements To The 10 Instances of DataStructures With 40000 Elements

Inserting 80000 Random Elements To The 10 Instances of DataStructures With 80000 Elements

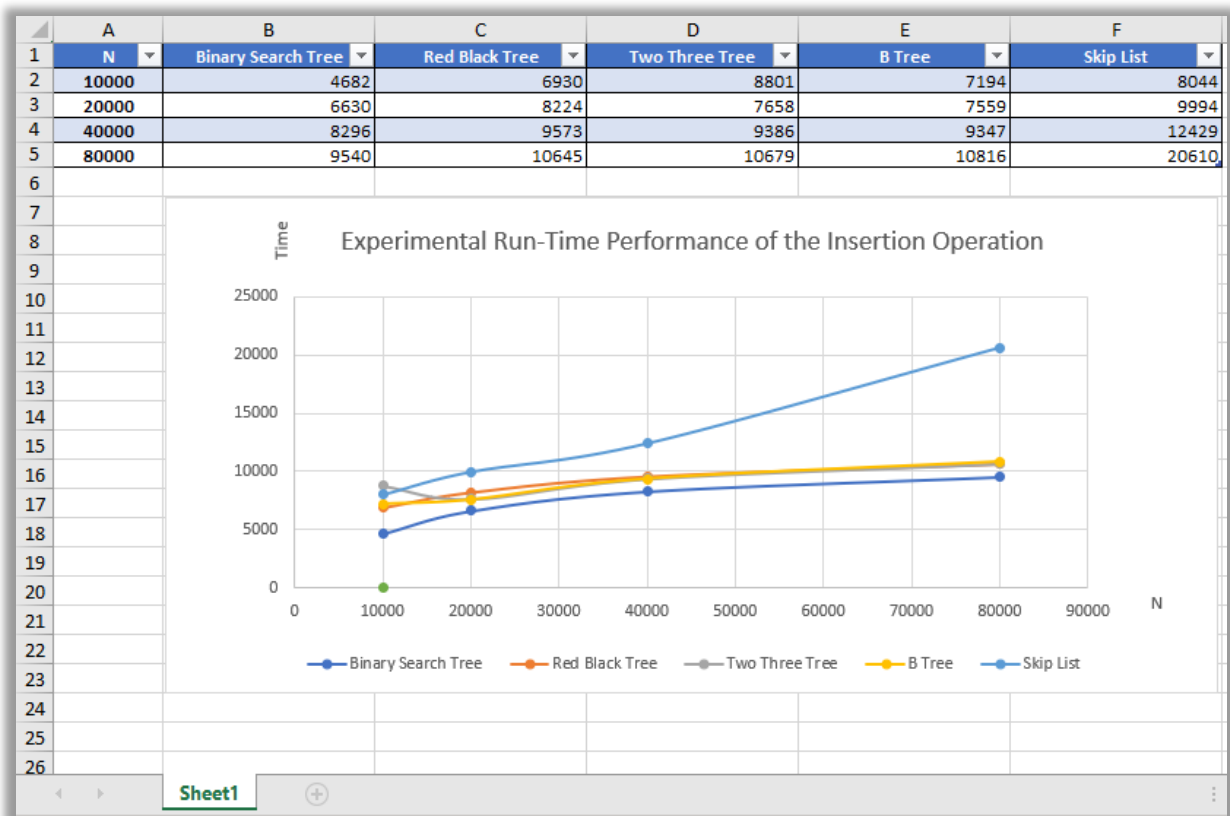
Binary Search Tree : Inserted 80000 Elements To The Tree
Red Black Tree : Inserted 80000 Elements To The Tree
Two Three Tree : Inserted 80000 Elements To The Tree
B Tree : Inserted 80000 Elements To The Tree
Skip List : Inserted 80000 Elements To The Tree

Inserted 80000 Random Elements To The 10 Instances of DataStructures With 80000 Elements

Test Results : Inserting Extra 100 Elements and Calculating Running Times

[illegible]

Test Results : Graph of Total and Average Running Times



END OF THE REPORT

LAST UPDATE

Jun 12, 2021 Saturday 10:00

STUDENT

**Şeyda Nur DEMİR
12 10 44 042**

LECTURER

Prof. Dr. Fatih Erdoğan SEVİLGİN

TEACHING ASSISTANT

**Başak KARAKAŞ
Mehmet Burak KOCA**

KOCAELİ, 2021