cse15l-lab-report5

Debugging Scenario

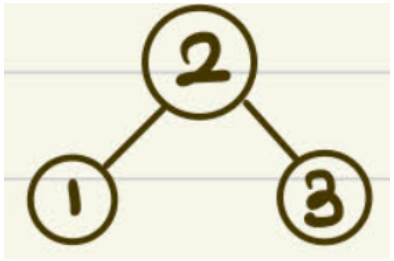
Student:

Hello, I am going to implement a binary search tree using array in java. I also tried to write a the tree structure and inserting node function in `ArrayTree.java` file and create a test.file called `ArrayTreeTests.java` to test if it work in correctly.
According to the definition of binary tree, we can use the following method to define a binary tree in an array:

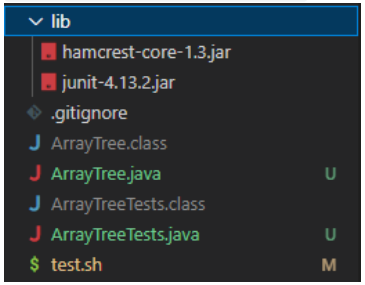
```
if root = array[n];
root.left = array[2 * n + 1];
root.right = array[2 * n + 2];
```

The main method takes an array as a parameter and returns a search binary tree formed from this array.
For example, if we input an array `[2, 3, 1]` as parameter, it will insert the node with `[2, 3, 1]` one by one in a correct place of the tree.
So when we get a binary search tree, the result will return `[2, 1, 3]`. If the node value equals to `0`, which means the node is null.

```
ArrayTree newTree = ArrayTree([2, 3, 1])
newTree.printTree() // it will return a array [2, 1, 3]
```



`File and Directory Structure`:



`ArrayTree.java` : contains the tree structor and functions

ArrayTree.java > ArrayTree

```java
import java.util.Arrays;

// a binary tree containing node value, left and right child
class ArrayTree {
    static int root = 0;
    static int[] tree = {};

    public ArrayTree(int[] array) {
        tree = new int[array.length];
        for (int n : array) {
            insertNode(n, root);
        }
    }

    public void insertNode(int key, int node) {
        if (tree[node] == 0) {
            tree[node] = key;
            return;
        } else if (key <= tree[node]) { // insert to its left subtree
            node = (node * 2) + 1;
            insertNode(key, node);
        } else { // insert to its right subtree
            node = (node * 2) + 2;
            insertNode(key, node);
        }
    }

    // print Tree without the rest 0
    public int[] printTree() {
        if (tree.length == 0)
            return tree;
        int n = 0;
        for (int i = tree.length - 1; i > 0; i--) {
            if (tree[i] != 0) {
                n = i;
                break;
            }
        }
        return Arrays.copyOfRange(tree, 0, n + 1);
    }
}
```

`insertNode(int key, int node)` : comapred with the value with `tree[node]`, and insert a new node with value `key`.
`printTree()` : If the tree incomplete, we delete the extra `0` from the end of the array to the last not `0` number. For example:

```
if the tree is [2, 1, 0]
printTree() will return [2, 1]
```

`ArrayTreeTests.java` : contains some test functions to test if the tree is correct.

ArrayTreeTests.java > ...

```java
import static org.junit.Assert.*;
import org.junit.*;

public class ArrayTreeTests {

    @Test(timeout = 500)
    public void testArrayToBST() {
        int[] treeNodes = { 2, 3, 1 };
        ArrayTree newTree = new ArrayTree(treeNodes);
        assertArrayEquals(new int[] { 2, 1, 3 }, newTree.printTree());
    }

    @Test(timeout = 500)
    public void testArrayToBST2() {
        int[] treeNodes = {};
        ArrayTree newTree = new ArrayTree(treeNodes);
        assertArrayEquals(new int[] {}, newTree.printTree());
    }

    @Test(timeout = 500)
    public void testArrayToBST3() {
        int[] treeNodes = { 4, 5, 6, 2, 3, 1 };
        ArrayTree newTree = new ArrayTree(treeNodes);
        assertArrayEquals(new int[] { 4, 2, 5, 1, 3, 0, 6 }, newTree.printTree());
    }
}
```

`test.sh` : a script file to run the test file.

test.sh

```
javac -cp ".;lib/hamcrest-core-1.3.jar;lib/junit-4.13.2.jar" *.java
java -cp ".;lib/hamcrest-core-1.3.jar;lib/junit-4.13.2.jar" org.junit.runner.JUnitCore ArrayTreeTests
```

Issues

```
$ bash test.sh
JUnit version 4.13.2
...E
Time: 0.013
There was 1 failure:
1) testArrayToBST3(ArrayTreeTests)
java.lang.ArrayIndexOutOfBoundsException: Index 6 out of bounds for length 6
        at ArrayTree.insertNode(ArrayTree.java:16)
        at ArrayTree.insertNode(ArrayTree.java:24)
        at ArrayTree.insertNode(ArrayTree.java:24)
        at ArrayTree.<init>(ArrayTree.java:11)
        at ArrayTreeTests.testArrayToBST3(ArrayTreeTests.java:23)

FAILURES!!!
Tests run: 3,  Failures: 1
```

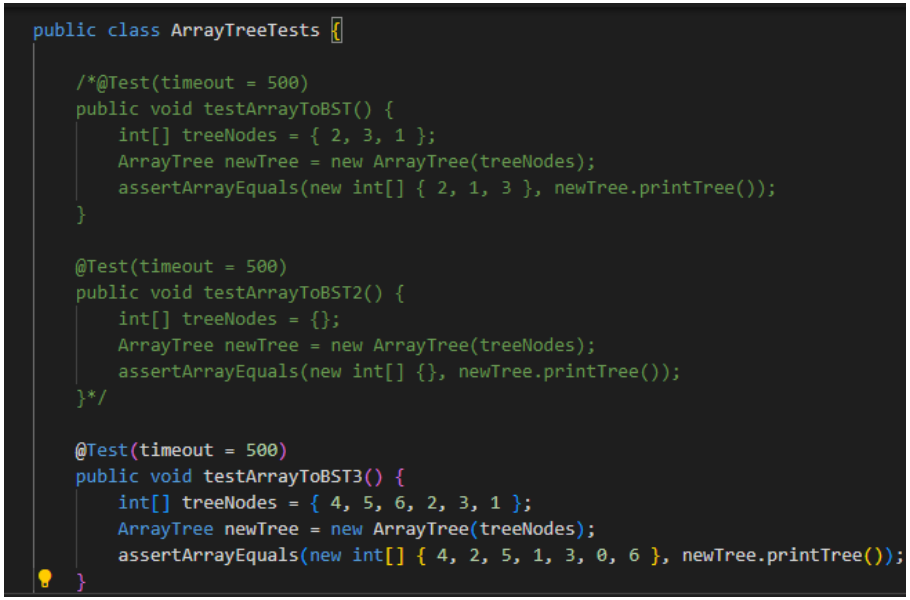
symptom

It successfully pass the test 1 and 2, which means the return is correct when I input a small size array and empty array. The test3 is fail. According to the error message from terminal, I can know that the error occurs when inserting a new node to the tree in the process. The total size of the tree is 6, but the node is going to inserting to out of that bounds. I guess the bug is localed at the size of tree array, but I didn't figure it out because I thought the size of the array after forming the tree should be the same as before. I want to get some suggestion from TAs.
Thank you!

TAs:

Hi, it is true that the number of actual elements in the tree is equal to the input elements in a arrays. However, what if there are empty nodes in the tree? In a tree formed by List, we can use `null` to represent an empty node, but what about in a tree formed by an array? Furthermore, in order to save space resources, please think about how large the array size should be to store a tree.

Debug:



```java
public class ArrayTreeTests {

    /*@Test(timeout = 500)
    public void testArrayToBST() {
        int[] treeNodes = { 2, 3, 1 };
        ArrayTree newTree = new ArrayTree(treeNodes);
        assertArrayEquals(new int[] { 2, 1, 3 }, newTree.printTree());
    }

    @Test(timeout = 500)
    public void testArrayToBST2() {
        int[] treeNodes = {};
        ArrayTree newTree = new ArrayTree(treeNodes);
        assertArrayEquals(new int[] {}, newTree.printTree());
    }*/

    @Test(timeout = 500)
    public void testArrayToBST3() {
        int[] treeNodes = { 4, 5, 6, 2, 3, 1 };
        ArrayTree newTree = new ArrayTree(treeNodes);
        assertArrayEquals(new int[] { 4, 2, 5, 1, 3, 0, 6 }, newTree.printTree());
    }
```

- Comment out the first two tests, then I can focus on debugging the last test.

```
debug.sh
1   javac -g -cp ".;lib/hamcrest-core-1.3.jar;lib/junit-4.13.2.jar" *.java
2   jdb -classpath ".;lib/hamcrest-core-1.3.jar;lib/junit-4.13.2.jar" org.junit.runner.JUnitCore ArrayTreeTests
```

- Create a debug.sh, which help me run `jdb` debugging quickly.

```
79250@Alice MINGW64 ~/Desktop/CSE15/cse-15l-lab7 (main)
Initializing jdb ...
> stop at ArrayTree:16
Deferring breakpoint ArrayTree:16.
It will be set after the class is loaded.
> run
run org.junit.runner.JUnitCore ArrayTreeTests
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
>
VM Started: JUnit version 4.13.2
.Set deferred breakpoint ArrayTree:16

Breakpoint hit: "thread=Time-limited test", ArrayTree.insertNode(), line=16 bci=0
16            if (tree[node] == 0) {

Time-limited test[1] locals
Method arguments:
key = 4
node = 0
Local variables:
Time-limited test[1]
```

- `stop at ArrayTree:16`: create a breakpoint at line 16.
- run and stop at that line.

- `locals` : show the local variables to check the process of the function.

```
node = 0
Local variables:
Time-limited test[1] cont
>
Breakpoint hit: "thread=Time-limited test", ArrayTree.insertNode(), line=16 bci=0
16            if (tree[node] == 0) {

Time-limited test[1] locals
Method arguments:
key = 5
node = 0
Local variables:
Time-limited test[1] cont
>
Breakpoint hit: "thread=Time-limited test", ArrayTree.insertNode(), line=16 bci=0
16            if (tree[node] == 0) {

Time-limited test[1] locals
Method arguments:
key = 5
node = 2
Local variables:
Time-limited test[1] step
>
Step completed: "thread=Time-limited test", ArrayTree.insertNode(), line=17 bci=8
17                tree[node] = key;

Time-limited test[1] step
>
Step completed: "thread=Time-limited test", ArrayTree.insertNode(), line=18 bci=14
18                return;

Time-limited test[1] step
>
Step completed: "thread=Time-limited test", ArrayTree.insertNode(), line=26 bci=51
26        }

Time-limited test[1] locals
Method arguments:
key = 5
node = 2
Local variables:
Time-limited test[1] step
>
Step completed: "thread=Time-limited test", ArrayTree.<init>(), line=10 bci=40
10            for (int n : array) {

Time-limited test[1] step
>
Step completed: "thread=Time-limited test", ArrayTree.<init>(), line=11 bci=31
11                insertNode(n, root);

Time-limited test[1] locals
Method arguments:
array = instance of int[6] (id=1097)
Local variables:
n = 6
Time-limited test[1]
```

- using `cont` and `step` command to let the program stop at the place where the error occur.

```
Time-limited test[1] locals
Method arguments:
key = 6
node = 6
Local variables:
Time-limited test[1] print ArrayTree.tree
 ArrayTree.tree = instance of int[6] (id=1098)
Time-limited test[1] dump ArrayTree.tree
 ArrayTree.tree = {
4, 0, 5, 0, 0, 0
}
Time-limited test[1]
```

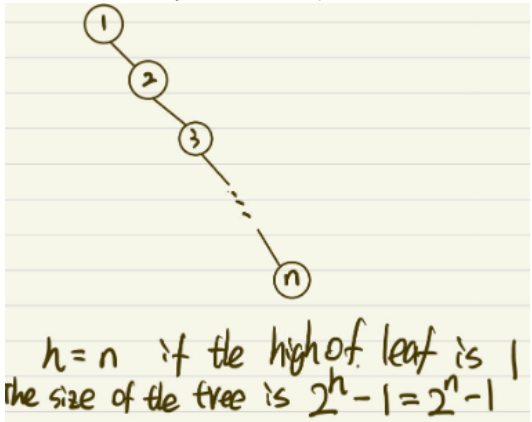
- `key = 6, node = 6` : the program is going to insert a value `6` to array[6].
- `print ArrayTree.tree` : we find that the size of the array is `6` ,which mean the maximum index is `array[5]` . There is no doubt that the program get excpetion.
- `dump ArrayTree.tree` : to show the content of the object tree. we find that there are many `0` in the array, which is the default value when creating the array.

```
9           tree = new int[array.length];
```

- Here's the bug code we need to fix.

Fixing the bug

- According to the jdb debugging, we already know that the reason of out of bounds is the number of elements in `ArrayTree` is more than that of the original array, because it needs to store `0` to represent empty nodes. So we need to adjust the size of the `ArrayTree` to prevent the index from going out of bounds.
- In order to find the minimum ArrayTree array size, I need to know how much space the ArrayTree needs at least to store the formed tree when an array of size n is input.



- When we use an array with size of n as a param, if the array is ascending order, nodes need to be inserted all the way to the right. At this time, `ArrayTree` also requires the largest capacity.

- so when `size of input array = n`, we should have at least `2^n - 1` size to storage the `Arraytree`.

```
public ArrayTree(int[] array) {
    tree = new int[(int) Math.pow(2, array.length) - 1];
    for (int n : array) {
        insertNode(n, root);
    }
}
```

- change the size to `Math.pow(2, array.length) - 1`.

Re-test

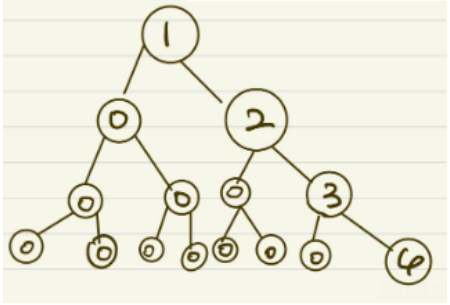
```
79250@Alice MINGW64 ~/Desktop/CSE15/cse-15l-lab7 (main)
$ bash test.sh
JUnit version 4.13.2
.
Time: 0.011

OK (1 test)
```

- re-run the test and get success.

```
@Test(timeout = 500)
public void testArrayToBST34() {
    int[] treeNodes = { 1, 2, 3, 4 };
    ArrayTree newTree = new ArrayTree(treeNodes);
    assertArrayEquals(new int[] { 1, 0, 2, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 4 }, newTree.printTree());
}
```

- Create a new test4 for a asending array. The input array is `[1, 2, 3, 4]`, and it will return a tree array `[1, 0, 2, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 4]`.



- Here's the tree and `0` node mean the empyty node `null`.

Final Test

```
79250@Alice MINGW64 ~/Desktop/CSE15/cse-15l-lab7 (main)
$ bash test.sh
JUnit version 4.13.2
....
Time: 0.013

OK (4 tests)
```

- All the 4 test are correct. The bug is fixed!

Final file after debug

`ArrayTree.java`

```java
import java.lang.Math;
import java.util.Arrays;

// a binary tree containing node value, left and right child
class ArrayTree {
    static int root = 0;
    static int[] tree = {};

    public ArrayTree(int[] array) {
        tree = new int[(int) Math.pow(2, array.length) - 1];
        for (int n : array) {
            insertNode(n, root);
        }
    }

    public void insertNode(int key, int node) {
        if (tree[node] == 0) {
            tree[node] = key;
            return;
        } else if (key <= tree[node]) { // insert to its left subtree
            node = (node * 2) + 1;
            insertNode(key, node);
        } else { // insert to its right subtree
            node = (node * 2) + 2;
            insertNode(key, node);
        }
    }

    // print Tree without the rest 0
    public int[] printTree() {
        if (tree.length == 0)
            return tree;
        int n = 0;
        for (int i = tree.length - 1; i > 0; i--) {
            if (tree[i] != 0) {
                n = i;
                break;
            }
        }
        return Arrays.copyOfRange(tree, 0, n + 1);
    }
}
```

`ArrayTreeTests.java`

```java
import static org.junit.Assert.*;
import org.junit.*;

public class ArrayTreeTests {

    @Test(timeout = 500)
    public void testArrayToBST() {
        int[] treeNodes = { 2, 3, 1 };
        ArrayTree newTree = new ArrayTree(treeNodes);
        assertArrayEquals(new int[] { 2, 1, 3 }, newTree.printTree());
    }

    @Test(timeout = 500)
    public void testArrayToBST2() {
        int[] treeNodes = {};
        ArrayTree newTree = new ArrayTree(treeNodes);
        assertArrayEquals(new int[] {}, newTree.printTree());
    }

    @Test(timeout = 500)
    public void testArrayToBST3() {
        int[] treeNodes = { 4, 5, 6, 2, 3, 1 };
        ArrayTree newTree = new ArrayTree(treeNodes);
        assertArrayEquals(new int[] { 4, 2, 5, 1, 3, 0, 6 }, newTree.printTree());
    }

    @Test(timeout = 500)
    public void testArrayToBST34() {
        int[] treeNodes = { 1, 2, 3, 4 };
        ArrayTree newTree = new ArrayTree(treeNodes);
        assertArrayEquals(new int[] { 1, 0, 2, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 4 }, newTree.printTree());
    }
}
```

`test.sh`

```
test.sh
1   javac -cp ".;lib/hamcrest-core-1.3.jar;lib/junit-4.13.2.jar" *.java
2   java -cp ".;lib/hamcrest-core-1.3.jar;lib/junit-4.13.2.jar" org.junit.runner.JUnitCore ArrayTreeTests
```

`debug`

```
debug.sh
1   javac -g -cp ".;lib/hamcrest-core-1.3.jar;lib/junit-4.13.2.jar" *.java
2   jdb -classpath ".;lib/hamcrest-core-1.3.jar;lib/junit-4.13.2.jar" org.junit.runner.JUnitCore ArrayTreeTests
```

Reflection

- The grading script is most cool thing I learn from the labs. It allows me to grade other people's code just if I know their github url. The grading script will automatically complete a series of operations, like git clone, creating and copying the files, and give the students feedback. The process of writing script is also interesting. We organize the commands in a .sh file, can check the running status based on the current error code and consisder what feedback we should give. What's more surprising is that we can also run our program on a remote server.

- Learning to use jdb to debug code is the most useful knowledge for me. I always thought that debugging on the server is very difficult, but this is not the case. At least we can use `jdb` to set breakpoints and obtain variable information.
- I think the `github` operations learned in this course are also very useful. Although I have used github before and understand how to push and pull code. But I have not used the issues feature. During the lab, I had the opportunity to discussed code with my teammates, and created issues with each other and made a pull request on github. I believe that these things I learned will be of great help to me in my future work.