CS203B - Assignment3 Explanation

谢玉博 12012515

tree.java

```
import java.util.*;

class Node {
    int val;
    Node left;
    Node right;

    public Node(int val) {
        this.val = val;
    }
}

public class tree {
    public static boolean ifBST(Integer[] arr, int r) {//r是输入数组的指针，从0开始，递归即每次加1，直到超过数组长度
        int leftChild = 2 * r + 1;
        int rightChild = 2 * r + 2;

        if (leftChild >= arr.length) { // 到二叉树第一个没有左子节点处跳出循环
            return true;
        }
        if (arr[rightChild] != -1 && arr[leftChild] != -1 && arr[r] != -1) {//均不为空
            if (arr[r] < arr[leftChild] || (rightChild < arr.length && arr[r] > arr[rightChild])) {
                return false;
            }
        } else if (arr[r] != -1 && arr[rightChild] != -1) {//左子结点为空
            if (rightChild < arr.length && arr[r] > arr[rightChild]) {
                return false;
            }
        } else if (arr[r] != -1 && arr[leftChild] != -1) {//右子节点为空
            if (arr[r] < arr[leftChild]) {
                return false;
            }
        } else {//剩余一种情况即结点本身为空，则无需比较
            return true;
        }
        return ifBST(arr, leftChild) && ifBST(arr, rightChild); // 递归检查左右子树
    }

    public static Node deleteNode(Node root, int key) {
        if (root.val == -1) {
```

```
            return new Node(-1);
        }
        if (root.val > key) {
            root.left = deleteNode(root.left, key);
            return root;
        }
        if (root.val < key) {
            root.right = deleteNode(root.right, key);
            return root;
        }
        if (root.val == key) {
            if (root.left.val == -1 && root.right.val == -1) {
                return new Node(-1);
            }
            if (root.right.val == -1) {
                return root.left;
            }
            if (root.left.val == -1) {
                return root.right;
            }
            Node successor = root.right;
            while (successor.left != null && successor.left.val != -1) {
                successor = successor.left;
            }
            root.right = deleteNode(root.right, successor.val);
            successor.right = root.right;
            successor.left = root.left;
            return successor;
        }
        return root;
    }

    public static Node buildTree(Integer[] nums, int i) {
        if (i >= nums.length || nums[i] == -1 || nums[i] == null) {
            return new Node(-1);
        }
        Node node = new Node(nums[i]);
        node.left = buildTree(nums, 2 * i + 1);
        node.right = buildTree(nums, 2 * i + 2);

        return node;
    }

    public static String serialize(Node root) {
        StringBuilder sb = new StringBuilder();
        Queue<Node> queue = new LinkedList<>();
        queue.offer(root);
        while (!queue.isEmpty()) {
            Node node = queue.poll();
            if (node == null) {
                sb.append("-1");
            } else {
                sb.append(node.val);
                queue.offer(node.left);
                queue.offer(node.right);
```

```
            }
            sb.append(" ");
        }
        sb.deleteCharAt(sb.length() - 1);
        return sb.toString();
    }


    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        Integer[] givenTree = new Integer[n];
        for (int i = 0; i < n; i++) {
            givenTree[i] = sc.nextInt();
        }

        int k = sc.nextInt();
        int[] toRemove = new int[k];
        for (int i = 0; i < k; i++) {
            toRemove[i] = sc.nextInt();
        }

        if (ifBST(givenTree, 0)) {
            System.out.println("Yes");
            Node root = buildTree(givenTree, 0);
            for (int j = 0; j < toRemove.length; j++) {
                root = deleteNode(root, toRemove[j]);
            }
            String string = serialize(root);
            String[] arr = string.split(" ");
            String newStr = String.join(" ", Arrays.copyOfRange(arr, 0, n));
            System.out.println(newStr);
        } else {
            System.out.println("No");
        }
    }
}
```

Explanation

在这段代码中，定义了一个名为`Node`的类，用于表示二叉树结点的值(`val`)以及它的左右子结点(`left`/`right`).

`ifBST()`方法使用递归，判断输入的二叉树是否为二叉搜索树，返回了一个Boolean值。

`deleteNode()`方法负责对输入的二叉树进行删除指定结点并构建新二叉查找树。

`buildTree()`方法把控制台输入的数组转换为Node类型的二叉树。

`serialize()`方法把二叉树按照层序表示顺序并分隔开，转换为字符串并输出至控制台。

在`main()`方法中，读取控制台里输入的二叉树和需要删除的结点，调用上述方法，构建二叉树，对输入的树进行判断；如果是二叉查找树则输出字符串"Yes"，并执行后续删除结点操作并输出新树，程序结束；如果不是二叉查找树，则输出字符串"No"，程序结束。

> **关于如何把输入数组转换成二叉树的问题，一开始我认为可以不对所有`-1`进行处理，直接构建二叉树，在后续方法里注意声明涉及`-1`的情况即可。后来测试程序时发现在测试第三组数据时就有问题，数字`63`没有出现在输出末尾，而是往前移了若干位。老师告诉我可能是`serialize()`方法的问题，且尽量不要把`-1`构建入二叉树中，否则后续声明条件会比较复杂。但自己测试时发现，若不把`-1`构建入二叉树中，通常会导致部分空结点缺失，这可能就是第三组样例出错，`63`没有处于正确位置的原因，所以最后我还是把`-1`放入了二叉树中，并相应的在其他方法里声明了大量关于`-1`的条件语句。**
>
> 其他少量注释在代码中。

Time Complexity

代码中包含的所有方法的时间复杂度如下：

- `ifBST()`：时间复杂度为`O(n)`，其中`n`为给定数组的长度。
- `deleteNode()`：时间复杂度取决于二叉搜索树的高度，最坏情况下为`O(n)`，其中`n`为树中的节点数。
- `buildTree()`：时间复杂度取决于二叉树的高度，最坏情况下为`O(n)`，其中`n`为树中的节点数。
- `serialize()`：时间复杂度为`O(n)`，其中`n`为树中的节点数。
- `main()`：主函数主要是调用其他函数，并且包含输入和输出，因此时间复杂度可以被忽略不计。

因此，这段代码的总体时间复杂度为`O(n)`，其中`n`为树中的节点数。

Random Test Code

用于生成随机测试数据的代码如下:

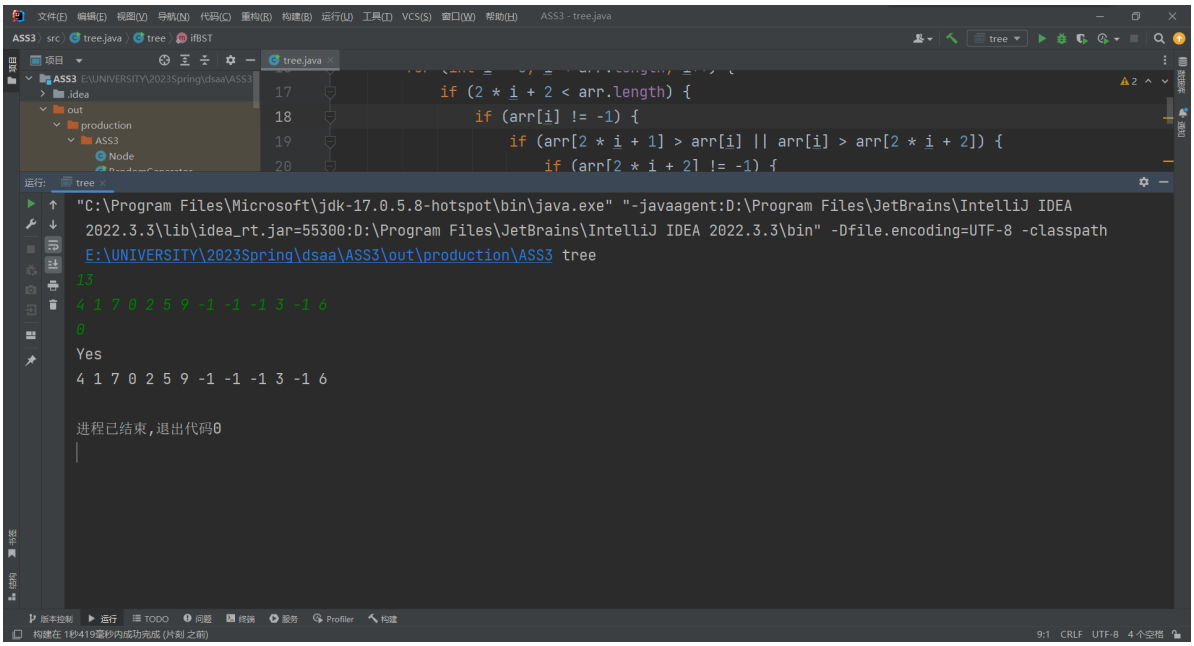
RandomGenerator.java

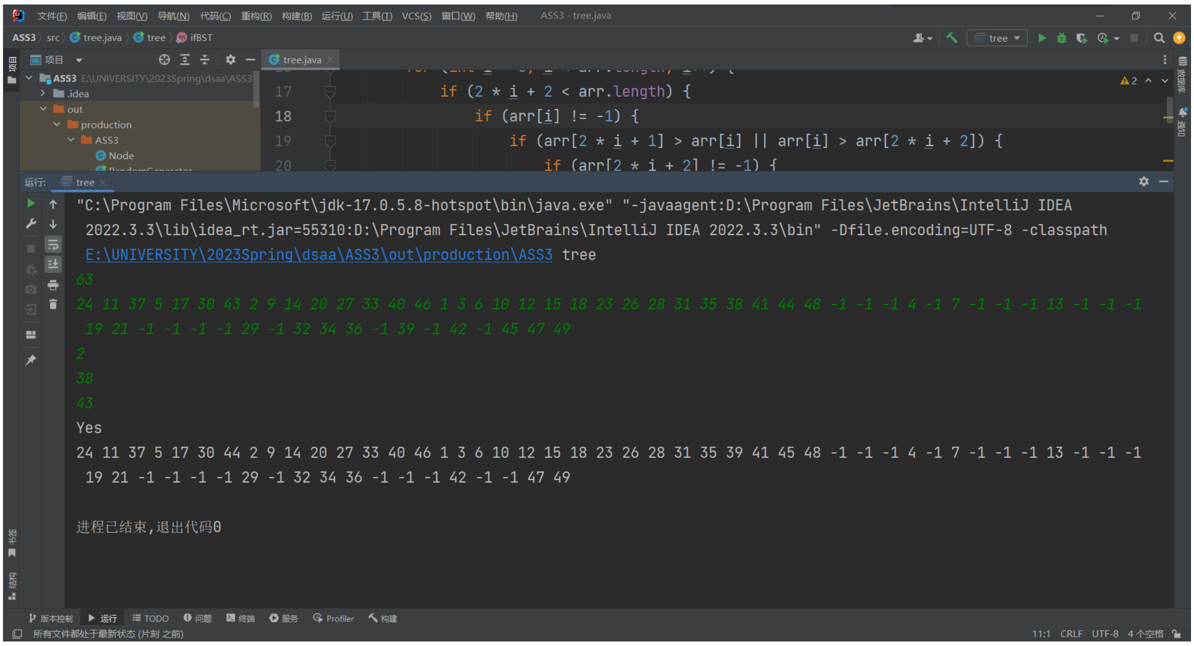
```
import java.io.*;
import java.util.*;

public class RandomGenerator {
    public static void main(String[] args) {
        int n = getRandomNumberInRange(1, 10000);
        Integer[] array = new Integer[n];
        for (int i = 0; i < n; i++) {
            array[i] = getRandomNumberInRange(-1, (int) Math.pow(10, 9));
        }
        int k = getRandomNumberInRange(1, 100);
        Integer[] kArray = new Integer[k];
        for (int i = 0; i < k; i++) {
            kArray[i] = getRandomNumberInRange(1, (int) Math.pow(10, 9));
        }
        try {
            File file = new File("output.txt");
            FileWriter writer = new FileWriter(file);
            writer.write(n + "\n");
            for (int i = 0; i < n; i++) {
                writer.write(array[i] + " ");
            }
            writer.write("\n" + k + "\n");
```

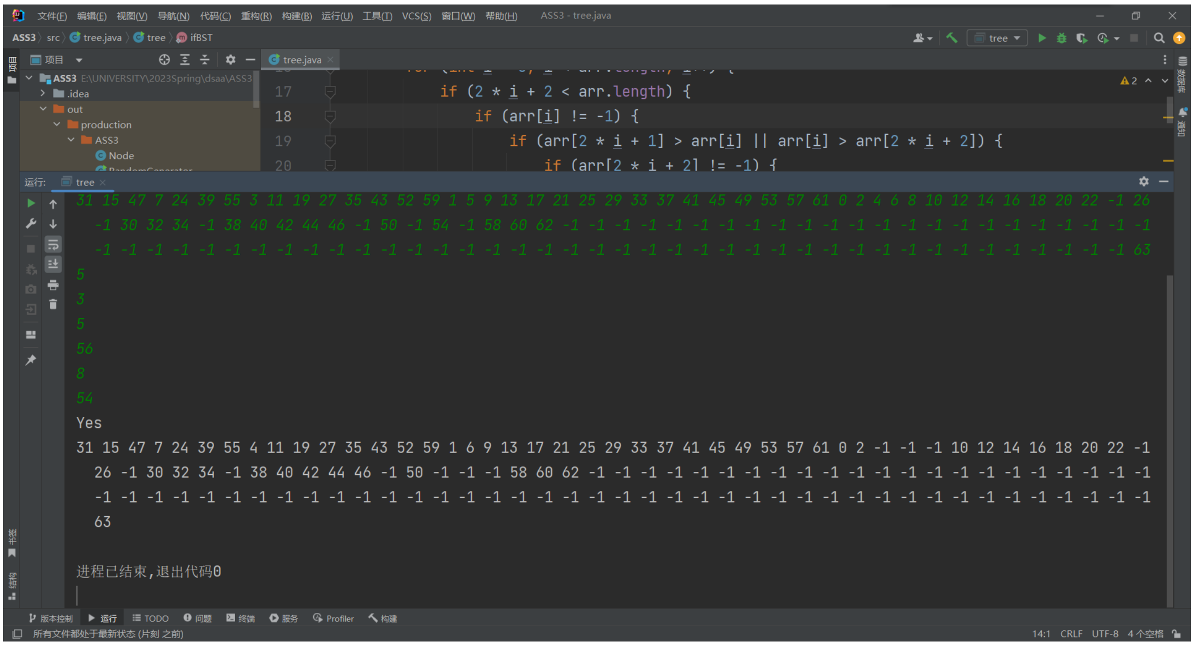
```
            for (int i = 0; i < k; i++) {
                writer.write(kArray[i] + " ");
            }
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static int getRandomNumberInRange(int min, int max) {
        if (min >= max) {
            throw new IllegalArgumentException("max must be greater than min");
        }
        Random r = new Random();
        return r.nextInt((max - min) + 1) + min;
    }
}
```

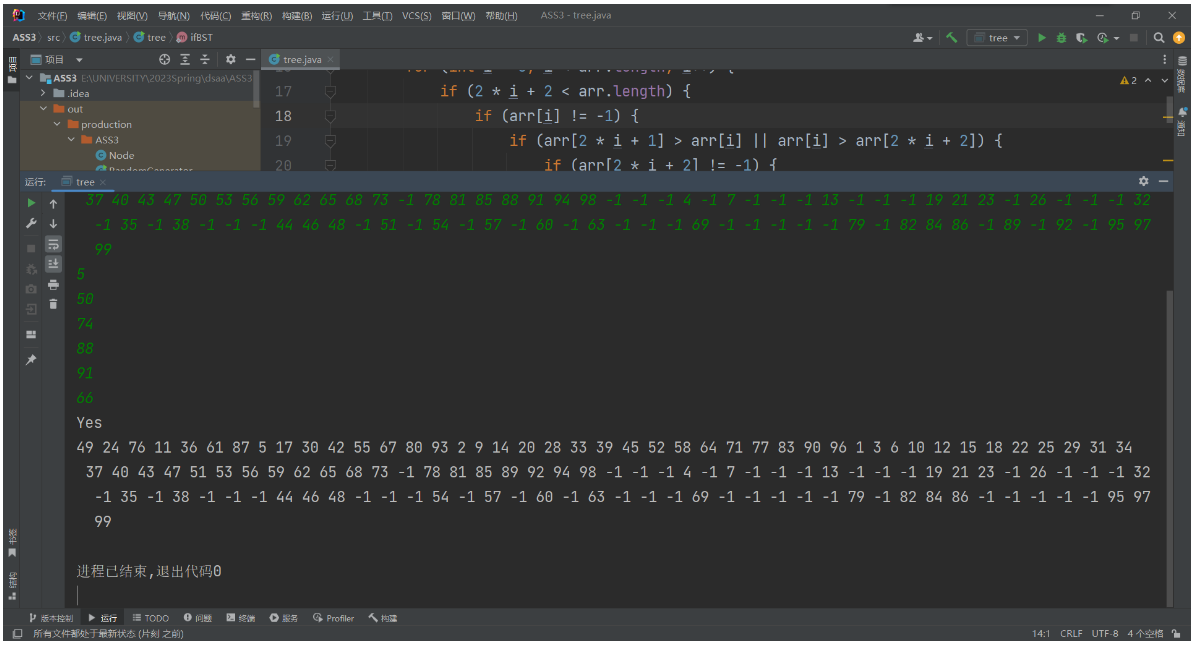
Correctness

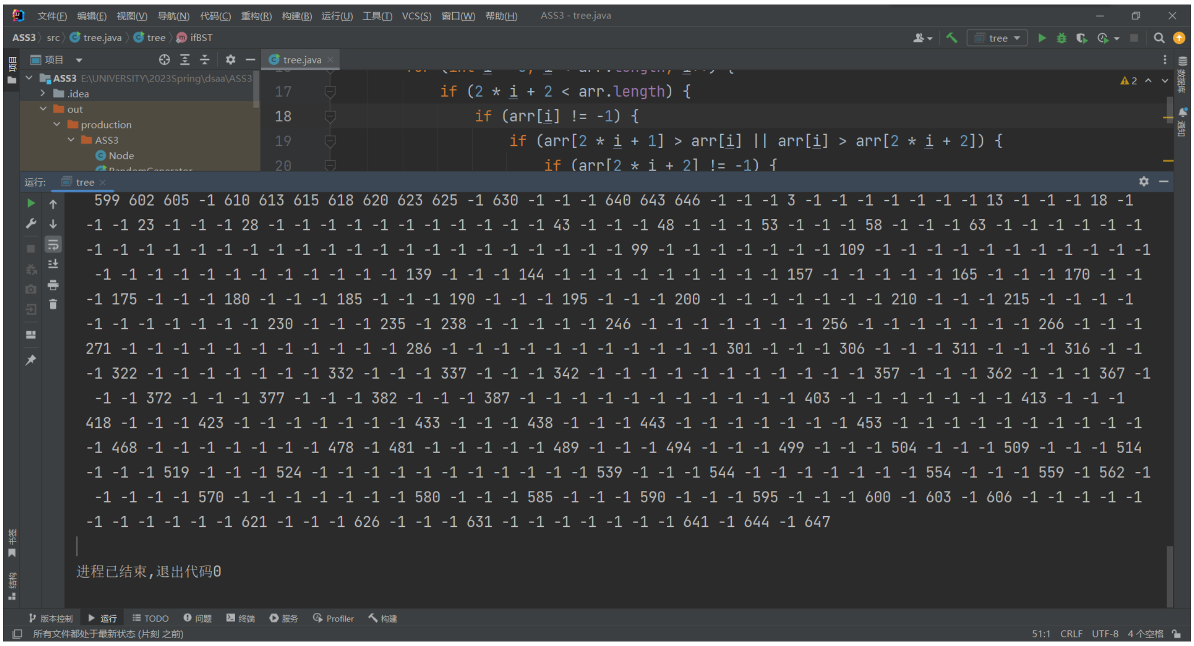
对于提供的十组测试数据，验证结果如下：



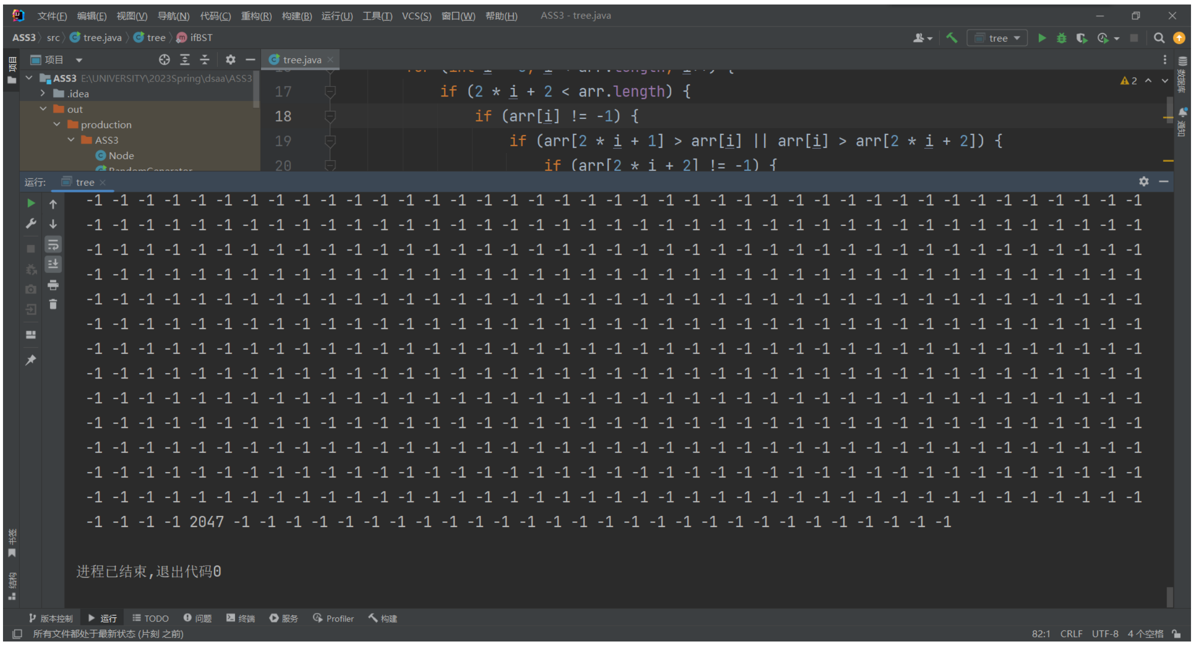

"C:\Program Files\Microsoft\jdk-17.0.5.8-hotspot\bin\java.exe" "-javaagent:D:\Program Files\JetBrains\IntelliJ IDEA 2022.3.3\lib\idea_rt.jar=55310:D:\Program Files\JetBrains\IntelliJ IDEA 2022.3.3\bin" -Dfile.encoding=UTF-8 -classpath E:\UNIVERSITY\2023Spring\dsaa\ASS3\out\production\ASS3 tree
63
24 11 37 5 17 30 43 2 9 14 20 27 33 40 46 1 3 6 10 12 15 18 23 26 28 31 35 38 41 44 48 -1 -1 -1 4 -1 7 -1 -1 -1 13 -1 -1 -1
19 21 -1 -1 -1 -1 29 -1 32 34 36 -1 39 -1 42 -1 45 47 49
2
38
43
Yes
24 11 37 5 17 30 44 2 9 14 20 27 33 40 46 1 3 6 10 12 15 18 23 26 28 31 35 39 41 45 48 -1 -1 -1 4 -1 7 -1 -1 -1 13 -1 -1 -1
19 21 -1 -1 -1 -1 29 -1 32 34 36 -1 -1 -1 42 -1 -1 47 49
进程已结束,退出代码0


31 15 47 7 24 39 55 3 11 19 27 35 43 52 59 1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 0 2 4 6 8 10 12 14 16 18 20 22 -1 26
-1 30 32 34 -1 38 40 42 44 46 -1 50 -1 54 -1 58 60 62 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 63
5
3
5
56
8
54
Yes
31 15 47 7 24 39 55 4 11 19 27 35 43 52 59 1 6 9 13 17 21 25 29 33 37 41 45 49 53 57 61 0 2 -1 -1 -1 10 12 14 16 18 20 22 -1
26 -1 30 32 34 -1 38 40 42 44 46 -1 50 -1 -1 -1 58 60 62 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
63
进程已结束,退出代码0

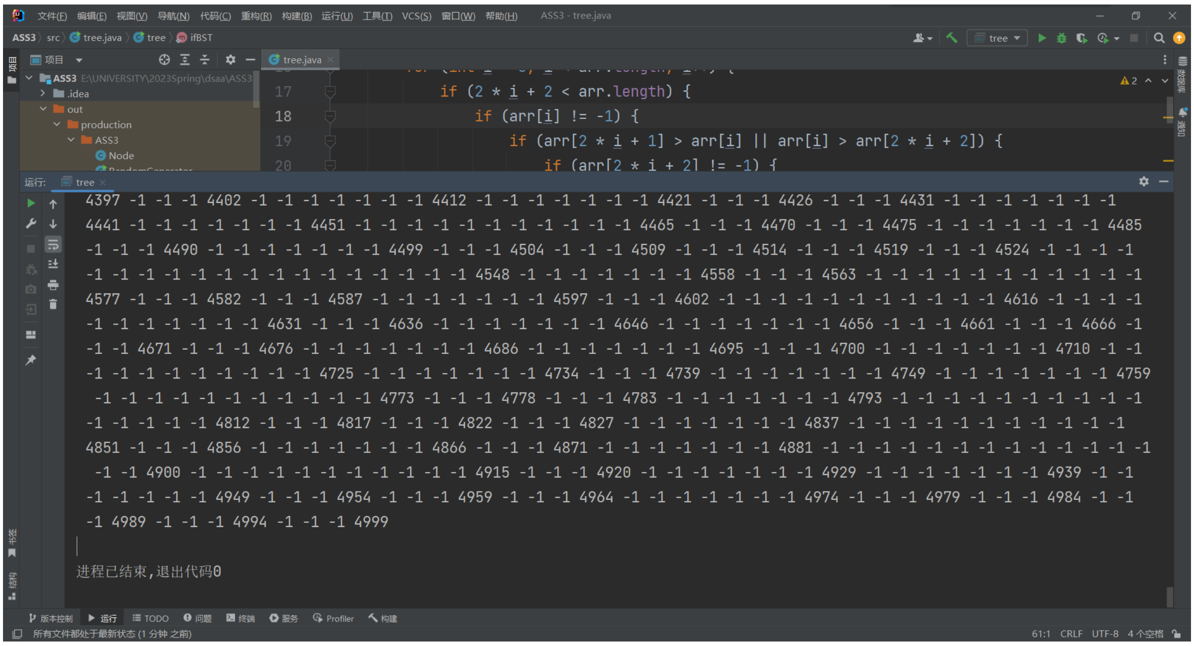

37 40 43 47 50 53 56 59 62 65 68 73 -1 78 81 85 88 91 94 98 -1 -1 -1 4 -1 7 -1 -1 -1 13 -1 -1 -1 19 21 23 -1 26 -1 -1 -1 32
-1 35 -1 38 -1 -1 -1 44 46 48 -1 51 -1 54 -1 57 -1 60 -1 63 -1 -1 -1 69 -1 -1 -1 -1 -1 79 -1 82 84 86 -1 89 -1 92 -1 95 97
99
5
50
74
88
91
66
Yes
49 24 76 11 36 61 87 5 17 30 42 55 67 80 93 2 9 14 20 28 33 39 45 52 58 64 71 77 83 90 96 1 3 6 10 12 15 18 22 25 29 31 34
37 40 43 47 51 53 56 59 62 65 68 73 -1 78 81 85 89 92 94 98 -1 -1 -1 4 -1 7 -1 -1 -1 13 -1 -1 -1 19 21 23 -1 26 -1 -1 -1 32
-1 35 -1 38 -1 -1 -1 44 46 48 -1 -1 -1 54 -1 57 -1 60 -1 63 -1 -1 -1 69 -1 -1 -1 -1 -1 79 -1 82 84 86 -1 -1 -1 -1 -1 95 97
99
进程已结束,退出代码0



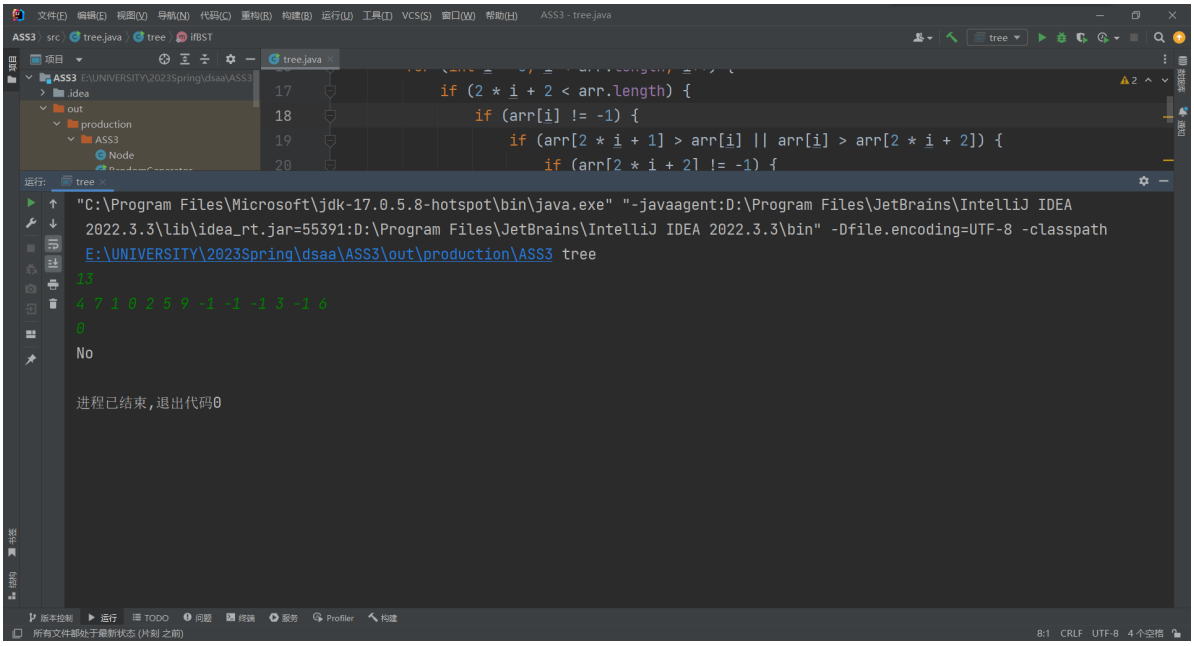
if (2 * i + 2 < arr.length) {
if (arr[i] != -1) {
if (arr[2 * i + 1] > arr[i] || arr[i] > arr[2 * i + 2]) {
599 602 605 -1 610 613 615 618 620 623 625 -1 630 -1 -1 -1 640 643 646 -1 -1 -1 3 -1 -1 -1 -1 -1 -1 -1 13 -1 -1 -1 18 -1
-1 -1 23 -1 -1 -1 28 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 48 -1 -1 -1 53 -1 -1 -1 58 -1 -1 -1 63 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 99 -1 -1 -1 -1 -1 -1 -1 109 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 139 -1 -1 -1 144 -1 -1 -1 -1 -1 -1 -1 -1 -1 157 -1 -1 -1 -1 -1 165 -1 -1 -1 170 -1 -1
-1 175 -1 -1 -1 180 -1 -1 -1 185 -1 -1 -1 190 -1 -1 -1 195 -1 -1 -1 200 -1 -1 -1 -1 -1 -1 -1 210 -1 -1 -1 215 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 230 -1 -1 -1 235 -1 238 -1 -1 -1 -1 -1 246 -1 -1 -1 -1 -1 -1 -1 256 -1 -1 -1 -1 -1 -1 -1 266 -1 -1 -1
271 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 286 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 301 -1 -1 -1 306 -1 -1 -1 311 -1 -1 -1 316 -1 -1
-1 322 -1 -1 -1 -1 -1 -1 -1 332 -1 -1 -1 337 -1 -1 -1 342 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 357 -1 -1 -1 362 -1 -1 -1 367 -1
-1 -1 372 -1 -1 -1 377 -1 -1 -1 382 -1 -1 -1 387 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 403 -1 -1 -1 -1 -1 -1 -1 413 -1 -1 -1
418 -1 -1 -1 423 -1 -1 -1 -1 -1 -1 -1 433 -1 -1 -1 438 -1 -1 -1 443 -1 -1 -1 -1 -1 -1 -1 453 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 468 -1 -1 -1 -1 -1 -1 -1 478 -1 481 -1 -1 -1 -1 -1 489 -1 -1 -1 494 -1 -1 -1 499 -1 -1 -1 504 -1 -1 -1 509 -1 -1 -1 514
-1 -1 -1 519 -1 -1 -1 524 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 539 -1 -1 -1 544 -1 -1 -1 -1 -1 -1 -1 554 -1 -1 -1 559 -1 562 -1
-1 -1 -1 -1 570 -1 -1 -1 -1 -1 -1 -1 580 -1 -1 -1 585 -1 -1 -1 590 -1 -1 -1 595 -1 -1 -1 600 -1 603 -1 606 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 621 -1 -1 -1 626 -1 -1 -1 631 -1 -1 -1 -1 -1 -1 -1 641 -1 644 -1 647
进程已结束,退出代码0

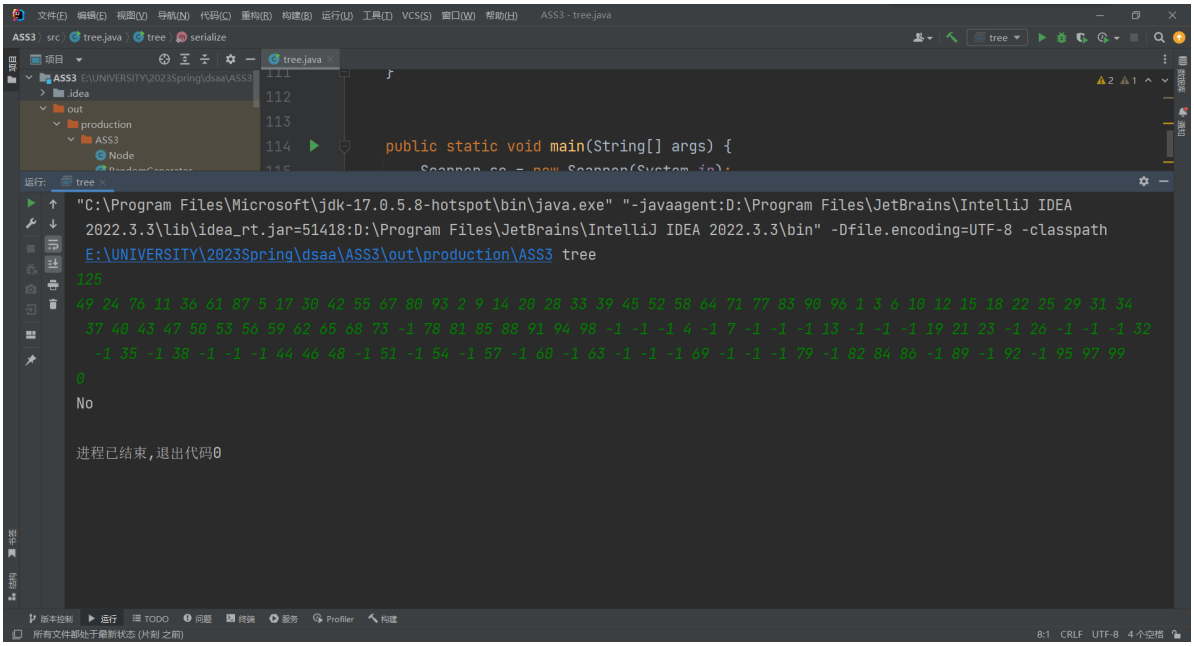


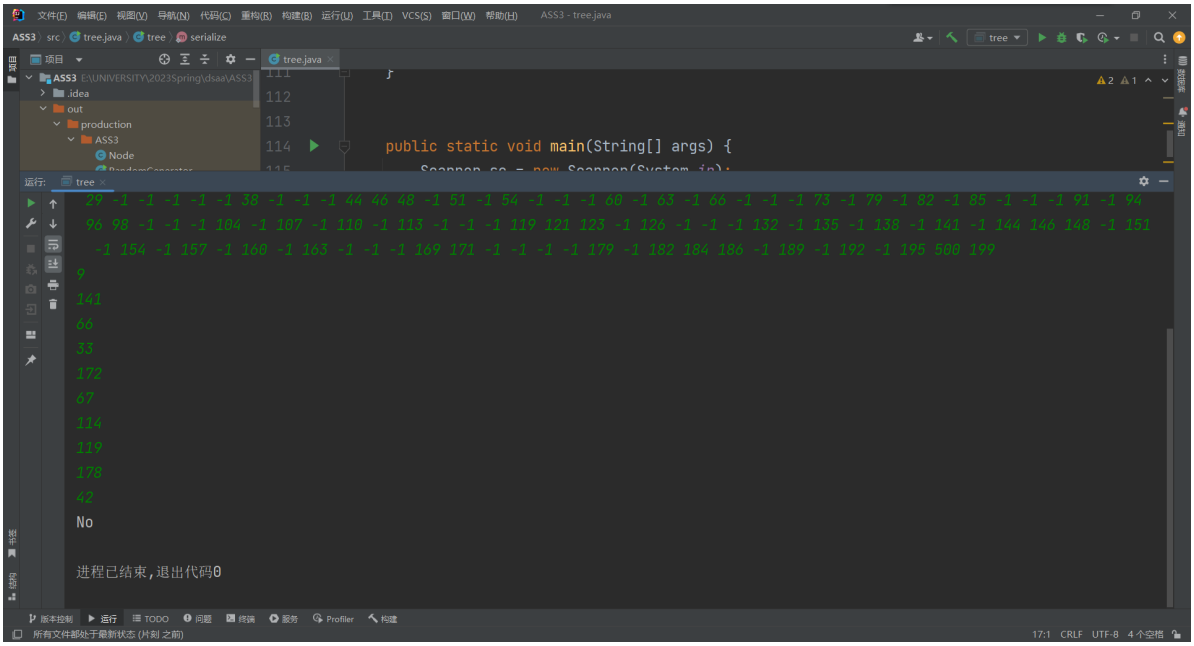
if (2 * i + 2 < arr.length) {
if (arr[i] != -1) {
if (arr[2 * i + 1] > arr[i] || arr[i] > arr[2 * i + 2]) {
-1 -1 -1 -1 2047 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
进程已结束,退出代码0



if (2 * i + 2 < arr.length) {
if (arr[i] != -1) {
if (arr[2 * i + 1] > arr[i] || arr[i] > arr[2 * i + 2]) {
4397 -1 -1 -1 4402 -1 -1 -1 -1 -1 -1 -1 4412 -1 -1 -1 -1 -1 -1 -1 4421 -1 -1 -1 4426 -1 -1 -1 4431 -1 -1 -1 -1 -1 -1 -1
4441 -1 -1 -1 -1 -1 -1 -1 4451 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 4465 -1 -1 -1 4470 -1 -1 -1 4475 -1 -1 -1 -1 -1 -1 -1 4485
-1 -1 -1 4490 -1 -1 -1 -1 -1 -1 -1 4499 -1 -1 -1 4504 -1 -1 -1 4509 -1 -1 -1 4514 -1 -1 -1 4519 -1 -1 -1 4524 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 4548 -1 -1 -1 -1 -1 -1 -1 4558 -1 -1 -1 4563 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
4577 -1 -1 -1 4582 -1 -1 -1 4587 -1 -1 -1 -1 -1 -1 -1 4597 -1 -1 -1 4602 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 4616 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 4631 -1 -1 -1 4636 -1 -1 -1 -1 -1 -1 -1 4646 -1 -1 -1 -1 -1 -1 -1 4656 -1 -1 -1 4661 -1 -1 -1 4666 -1
-1 -1 4671 -1 -1 -1 4676 -1 -1 -1 -1 -1 -1 -1 4686 -1 -1 -1 -1 -1 -1 -1 4695 -1 -1 -1 4700 -1 -1 -1 -1 -1 -1 -1 4710 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 4725 -1 -1 -1 -1 -1 -1 -1 4734 -1 -1 -1 4739 -1 -1 -1 -1 -1 -1 -1 4749 -1 -1 -1 -1 -1 -1 -1 4759
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 4773 -1 -1 -1 4778 -1 -1 -1 4783 -1 -1 -1 -1 -1 -1 -1 4793 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 4812 -1 -1 -1 4817 -1 -1 -1 4822 -1 -1 -1 4827 -1 -1 -1 -1 -1 -1 -1 4837 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
4851 -1 -1 -1 4856 -1 -1 -1 -1 -1 -1 -1 4866 -1 -1 -1 4871 -1 -1 -1 -1 -1 -1 -1 4881 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 4900 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 4915 -1 -1 -1 4920 -1 -1 -1 -1 -1 -1 -1 4929 -1 -1 -1 -1 -1 -1 -1 4939 -1 -1
-1 -1 -1 -1 -1 4949 -1 -1 -1 4954 -1 -1 -1 4959 -1 -1 -1 4964 -1 -1 -1 -1 -1 -1 -1 4974 -1 -1 -1 4979 -1 -1 -1 4984 -1 -1
-1 4989 -1 -1 -1 4994 -1 -1 -1 4999
进程已结束,退出代码0


if (2 * i + 2 < arr.length) {
if (arr[i] != -1) {
if (arr[2 * i + 1] > arr[i] || arr[i] > arr[2 * i + 2]) {
"C:\Program Files\Microsoft\jdk-17.0.5.8-hotspot\bin\java.exe" "-javaagent:D:\Program Files\JetBrains\IntelliJ IDEA 2022.3.3\lib\idea_rt.jar=55391:D:\Program Files\JetBrains\IntelliJ IDEA 2022.3.3\bin" -Dfile.encoding=UTF-8 -classpath E:\UNIVERSITY\2023Spring\dsaa\ASS3\out\production\ASS3 tree
13
4 7 1 0 2 5 9 -1 -1 -1 3 -1 6
0
No
进程已结束,退出代码0


public static void main(String[] args) {
"C:\Program Files\Microsoft\jdk-17.0.5.8-hotspot\bin\java.exe" "-javaagent:D:\Program Files\JetBrains\IntelliJ IDEA 2022.3.3\lib\idea_rt.jar=55391:D:\Program Files\JetBrains\IntelliJ IDEA 2022.3.3\bin" -Dfile.encoding=UTF-8 -classpath E:\UNIVERSITY\2023Spring\dsaa\ASS3\out\production\ASS3 tree
125
49 24 76 11 36 61 87 5 17 30 42 55 67 80 93 2 9 14 20 28 33 39 45 52 58 64 71 77 83 90 96 1 3 6 10 12 15 18 22 25 29 31 34 37 40 43 47 50 53 56 59 62 65 68 73 -1 78 81 85 88 91 94 98 -1 -1 -1 4 -1 7 -1 -1 -1 13 -1 -1 -1 19 21 23 -1 26 -1 -1 -1 32 -1 35 -1 38 -1 -1 -1 44 46 48 -1 51 -1 54 -1 57 -1 60 -1 63 -1 -1 -1 69 -1 -1 -1 79 -1 82 84 86 -1 89 -1 92 -1 95 97 99
0
No
进程已结束,退出代码0


public static void main(String[] args) {
29 -1 -1 -1 -1 -1 38 -1 -1 -1 44 46 48 -1 51 -1 54 -1 -1 -1 60 -1 63 -1 66 -1 -1 -1 73 -1 79 -1 82 -1 85 -1 -1 -1 91 -1 94 96 98 -1 -1 -1 104 -1 107 -1 110 -1 113 -1 -1 -1 119 121 123 -1 126 -1 -1 -1 132 -1 135 -1 138 -1 141 -1 144 146 148 -1 151 -1 154 -1 157 -1 160 -1 163 -1 -1 -1 169 171 -1 -1 -1 -1 179 -1 182 184 186 -1 189 -1 192 -1 195 500 199
9
141
66
33
172
67
114
119
178
42
No
进程已结束,退出代码0

在这十个测试样例中的结果中，只有第六组数据测试结果与答案不一致，且不同之处仅在于数字 `2047` 没有位于输出结果的末端。暂不清楚原因如何，但猜测可能和前面所述测试到第三组数据时 `63` 位置不对的原因类似。但为何现在 `63` 位置正确， `2047` 位置错误，猜测可能是第六组测试数据数组长度过大，最后存在过多空结点，而代码没有考虑到类似情况，从而出现错误。

The End, thank you!