

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

Solution 2

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 import java.util.*;
4
5 class Program {
6     // O(n) time | O(d) space - where n is the number of nodes in the B
7     // and d is the depth (height) of the Binary Tree
8     public static BinaryTreeNode flattenBinaryTree(BinaryTreeNode root) {
9         flattenTree(root);
10        return getLeftMost(root);
11    }
12
13    public static BinaryTreeNode[] flattenTree(BinaryTreeNode node) {
14        BinaryTreeNode leftMost;
15        BinaryTreeNode rightMost;
16
17        if (node.left == null) {
18            leftMost = node;
19        } else {
20            BinaryTreeNode[] leftAndRightMostNodes = flattenTree(node.left);
21            connectNodes(leftAndRightMostNodes[1], node);
22            leftMost = leftAndRightMostNodes[0];
23        }
24
25        if (node.right == null) {
26            rightMost = node;
27        } else {
28            BinaryTreeNode[] leftAndRightMostNodes = flattenTree(node.right);
29            connectNodes(node, leftAndRightMostNodes[0]);
30            rightMost = leftAndRightMostNodes[1];
31        }
32
33        return new BinaryTreeNode[] {leftMost, rightMost};
34    }
35}
```

Solution 1

Solution 2

Solution 3

```
1 class Program {
2     public static BinaryTreeNode flattenBinaryTree(BinaryTreeNode root) {
3         // Write your code here.
4         return root;
5     }
6
7     // This is the class of the input root. Do not edit it.
8     static class BinaryTreeNode {
9         int value;
10        BinaryTreeNode left = null;
11        BinaryTreeNode right = null;
12
13        public BinaryTreeNode(int value) {
14            this.value = value;
15        }
16    }
17 }
18
```

Our Tests

Custom Output

Submit Code

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

