```
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class BinaryTree {
  constructor(value) {
    this.value = value;
    this.left = null;
    this.right = null;
  }
}

// O(n) time | O(d) space - where n is the number of nodes in the Binary Tree and d is the depth (height) of the Binary Tree
function flattenBinaryTree(root) {
  const [leftMost, _] = flattenTree(root);
  return leftMost;
}

function flattenTree(node) {
  let leftMost, rightMost;

  if (node.left === null) {
    leftMost = node;
  } else {
    const [leftSubtreeLeftMost, leftSubtreeRightMost] = flattenTree(node.left);
    connectNodes(leftSubtreeRightMost, node);
    leftMost = leftSubtreeLeftMost;
  }

  if (node.right === null) {
    rightMost = node;
  } else {
    const [rightSubtreeLeftMost, rightSubtreeRightMost] = flattenTree(node.right);
    connectNodes(node, rightSubtreeLeftMost);
    rightMost = rightSubtreeRightMost;
  }

  return [leftMost, rightMost];
}

function connectNodes(left, right) {
  left.right = right;
  right.left = left;
}

exports.BinaryTree = BinaryTree;
exports.flattenBinaryTree = flattenBinaryTree;

```