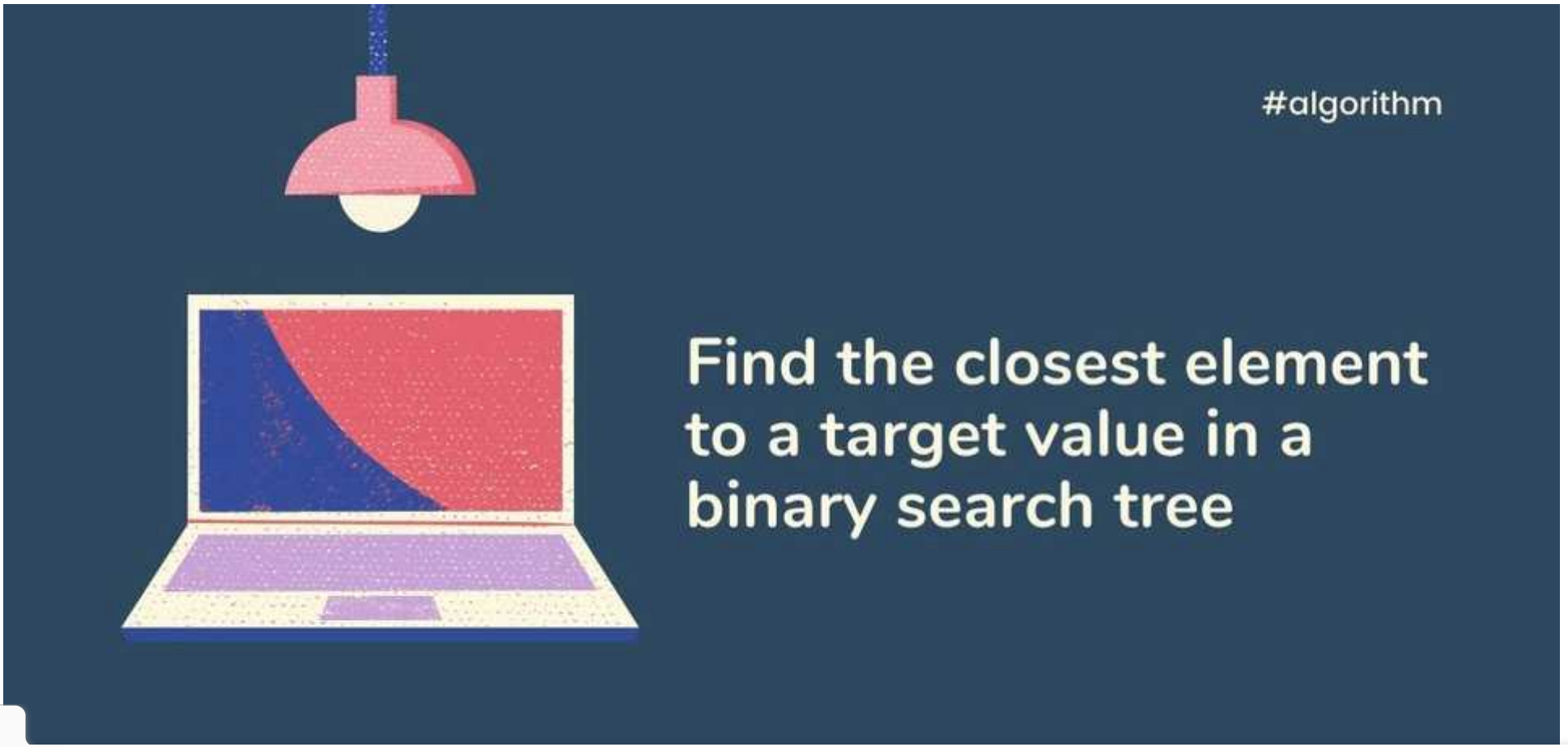
CALLICODER

Find the closest element to a target value in a binary search tree

by RAJEEV SINGH · ALGORITHMS · NOVEMBER 01, 2019 · 1 MINS READ



Example

```
Input
type: post
-----
Binary Search Tree:
     9
   /   \
  4      17
 / \      \
3   6      22
   / \     /
  5   7   20

target: 18


Output
type: post
------
17
```

- **Average:** O(log(n)) time | O(log(n)) space
- **Worst:** O(n) time | O(n) space

```
class TreeNode {
  int val;
  TreeNode left;
  TreeNode right;
  TreeNode(int val) {
    this.val = val;
  }
}

class ClosestElementInBST {
  private static TreeNode findClosestNode(TreeNode node, int target) {
    if (target < node.val && node.left != null) {
      // Closest node is either the current node or a node in the left subtree
      TreeNode closestNodeLeftSubtree = findClosestNode(node.left, target);
      return getClosestNode(node, closestNodeLeftSubtree, target);
    } else if (target > node.val && node.right != null){
      // Closest node is either the current node or a node in the right subtree
```

```
  }

  private static TreeNode getClosestNode(TreeNode node1, TreeNode node2, int target) {
    if(Math.abs(target - node1.val) < Math.abs(target - node2.val)) {
      return node1;
    } else {
      return node2;
    }
  }

  public static void main(String[] args) {
    TreeNode node = new TreeNode(9);
    node.left = new TreeNode(4);
    node.right = new TreeNode(17);

    node.left.left = new TreeNode(3);
    node.left.right = new TreeNode(6);
    node.left.right.left = new TreeNode(5);
    node.left.right.right = new TreeNode(7);

    node.right.right = new TreeNode(22);
```

```
  }
}
```

METHOD 2. Iterative solution

Complexity

- **Average:** O(log(n)) time | O(1) space
- **Worst:** O(n) time | O(1) space

```
class TreeNode {
  int val;
  TreeNode left;
  TreeNode right;
  TreeNode(int val) {
    this.val = val;
  }
}
```

```
  double minDiff = Double.MAX_VALUE;

  while(currentNode != null) {
    double currentDiff = Math.abs(target - currentNode.val);
    if(currentDiff < minDiff) {
      minDiff = currentDiff;
      closestValue = currentNode.val;
    }
    if(target < currentNode.val) {
      currentNode = currentNode.left;
    } else if (target > currentNode.val) {
      currentNode = currentNode.right;
    } else {
      break;
    }
  }
  return closestValue;
}

public static void main(String[] args) {
  TreeNode node = new TreeNode(9);
```

```
    node.left.right.left = new TreeNode(5);
    node.left.right.right = new TreeNode(7);

    node.right.right = new TreeNode(22);
    node.right.right.left = new TreeNode(20);

    System.out.println(findClosestValue(node, 18));
  }
}
```

CalliCoder