**75 Days of Code**
**Day 41**
**Problem no : 700**
**Problem Title : Search in a Binary Search Tree**
**Type : tree /BST**

**You are given the root of a binary search tree (BST) and an integer val.**

**Find the node in the BST that the node's value equals val and return the subtree rooted with that node. If such a node does not exist, return null.**

**Example 1:**

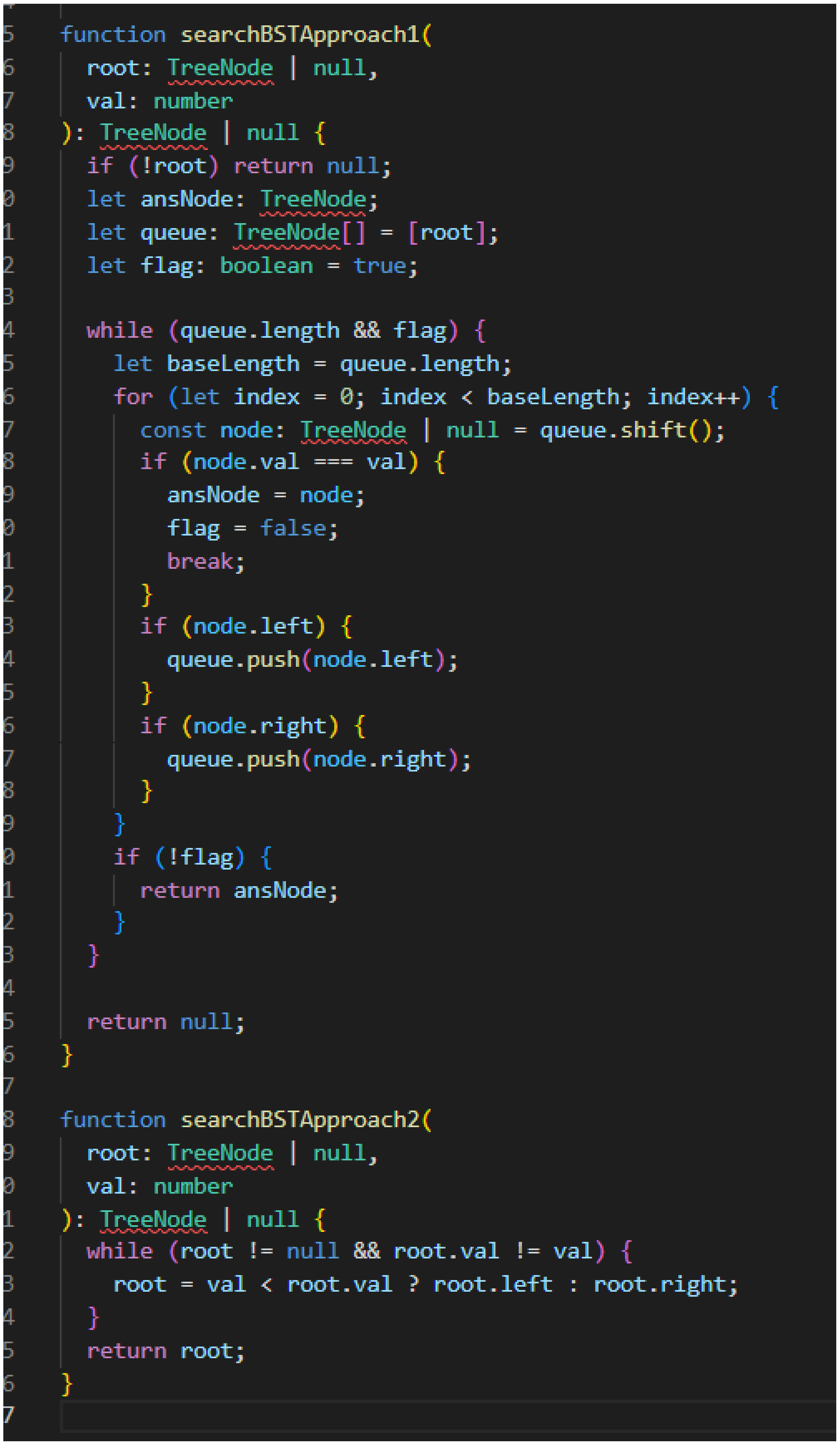
**Input: root = [4,2,7,1,3], val = 2**
**Output: [2,1,3]**
**Example 2:**

**Input: root = [4,2,7,1,3], val = 5**
**Output: []**

## Solution using BFS

Shubham Agrahari

1. Use Queue to enqueue (insert at first) and dequeue(delete at first)
2. Loop for each element (which acts as a level )

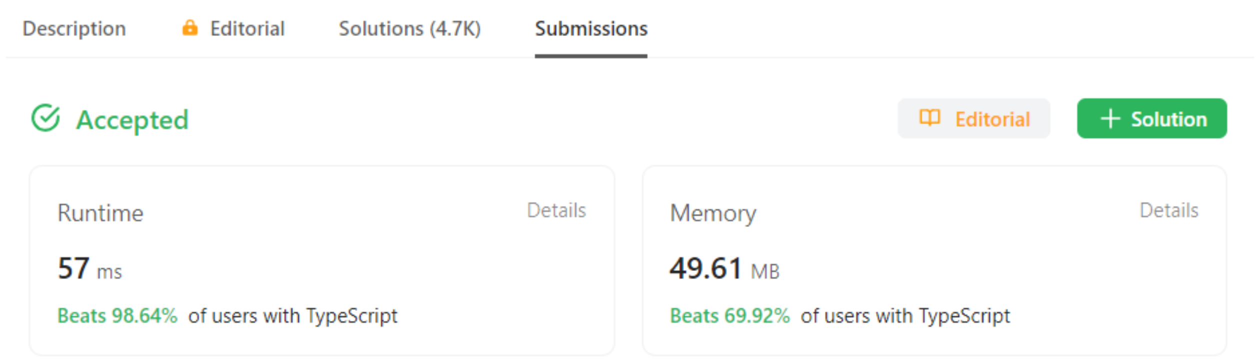

```
function searchBSTApproach1(
  root: TreeNode | null,
  val: number
): TreeNode | null {
  if (!root) return null;
  let ansNode: TreeNode;
  let queue: TreeNode[] = [root];
  let flag: boolean = true;

  while (queue.length && flag) {
    let baseLength = queue.length;
    for (let index = 0; index < baseLength; index++) {
      const node: TreeNode | null = queue.shift();
      if (node.val === val) {
        ansNode = node;
        flag = false;
        break;
      }
      if (node.left) {
        queue.push(node.left);
      }
      if (node.right) {
        queue.push(node.right);
      }
    }
    if (!flag) {
      return ansNode;
    }
  }

  return null;
}

function searchBSTApproach2(
  root: TreeNode | null,
  val: number
): TreeNode | null {
  while (root != null && root.val != val) {
    root = val < root.val ? root.left : root.right;
  }
  return root;
}
```

Shubham Agrahari



Accepted

Runtime

57 ms

Beats 98.64% of users with TypeScript

Memory

49.61 MB

Beats 69.92% of users with TypeScript

Shubham Agrahari