0173. 二叉搜索树迭代器

ITCharge 大约 2 分钟

- 标签：栈、树、设计、二叉搜索树、二叉树、迭代器
- 难度：中等

题目链接

- 0173. 二叉搜索树迭代器 - 力扣

题目大意

要求：实现一个二叉搜索树的迭代器 BSTIterator。表示一个按中序遍历二叉搜索树（BST）的迭代器：

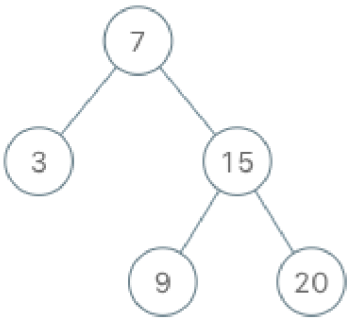
- `def __init__(self, root: TreeNode):` ：初始化 BSTIterator 类的一个对象，会给出二叉搜索树的根节点。
- `def hasNext(self) -> bool:` ：如 右指针遍历存在数字，则返回 True，否则返回 False。
- `def next(self) -> int:` ：将指针向右移动，返回指针处的数字。

说明：

- 指针初始化为一个不存在于 BST 中的数字，所以对 `next()` 的首次调用将返回 BST 中的最小元素。
- 可以假设 `next()` 调用总是有效的，也就是说，当调用 `next()` 时，BST 的中序遍历中至少存在一个下一个数字。
- 树中节点的数目在范围 $[1, 10^5]$ 内。
- $0 \leq Node.val \leq 10^6$。
- 最多调用 10^5 次 `hasNext` 和 `next` 操作。
- 进阶：设计一个满足下述条件的解决方案， `next()` 和 `hasNext()` 操作均摊时间复杂度为 `O(1)` ，并使用 `O(h)` 内存。其中 `h` 是树的高度。

示例：

- 示例 1：




```py
输入
["BSTIterator", "next", "next", "hasNext", "next", "hasNext", "next", "hasNext",
"next", "hasNext"]
[[[7, 3, 15, null, null, 9, 20]], [], [], [], [], [], [], [], [], []]
输出
[null, 3, 7, true, 9, true, 15, true, 20, false]
```

解题思路

思路 1：中序遍历二叉搜索树

中序遍历的顺序是：左、根、右。我们使用一个栈来保存节点，以便于迭代的时候取出对应节点。

- 初始的遍历当前节点的左子树，将其路径上的节点存储到栈中。
- 调用 next 方法的时候，从栈顶取出节点，因为之前已经将路径上的左子树全部存入了栈中，所以此时该节点的左子树为空，这时候取出节点右子树，再将右子树的左子树进行递归遍历，并将其路径上的节点存储到栈中。
- 调用 hasNext 的方法的时候，直接判断栈中是否有值即可。

思路 1：代码

```py
class BSTIterator:

    def __init__(self, root: TreeNode):
        self.stack = []
        self.in_order(root)

    def in_order(self, node):
```

```
        while node:
            self.stack.append(node)
            node = node.left

    def next(self) -> int:
        node = self.stack.pop()
        if node.right:
            self.in_order(node.right)
        return node.val

    def hasNext(self) -> bool:
        return len(self.stack) != 0
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$，其中 n 为树中节点数量。
- **空间复杂度**：$O(n)$。