

LintCode领扣题解 (/problem) / 二叉查找树迭代器 · Binary Search Tree Iterator

二叉查找树迭代器 · Binary Search Tree Iterator

中文

- 微软 (/problem/?tags=microsoft)
- 脸书 (/problem/?tags=facebook)
- 领英 (/problem/?tags=linkedin)
- 谷歌 (/problem/?tags=google)
- 非递归 (/problem/?tags=non-recursion)
- LintCode 版权所有 (/problem/?tags=lintcode-copyright)
- 二叉查找树 (/problem/?tags=binary-search-tree)
- 二叉树 (/problem/?tags=binary-tree)

描述

设计实现一个带有下列属性的二叉查找树的迭代器：
next()返回BST中下一个最小的元素

- 元素按照递增的顺序被访问（比如中序遍历）
- next() 和 hasNext() 的询问操作要求均摊时间复杂度是O(1)

样例

样例 1:

输入: {10,1,11,#,6,#,12}
输出: [1, 6, 10, 11, 12]
解释:
二叉查找树如下：
10
/\n1 11
\ \n6 12
可以返回二叉查找树的中序遍历 [1, 6, 10, 11, 12]

样例 2:

输入: {2,1,3}
输出: [1,2,3]
解释:
二叉查找树如下：
2
/\n1 3
可以返回二叉查找树的中序遍历 [1,2,3]

挑战

额外空间复杂度是O(h)，其中h是这棵树的高度
Super Star：使用O(1)的额外空间复杂度

在线评测地址: <https://www.lintcode.com/problem/binary-search-tree-iterator/> (https://www.lintcode.com/problem/binary-search-tree-iterator/)

收起题目描述 ^

语言类型

ALL (35)

java (15)

python (14)

cpp (5)

javascript (1)

上传题解



令狐冲

更新于 6/9/2020, 7:03:45 AM

这是一个非常通用的利用 stack 进行 Binary Tree Iterator 的写法。

stack 中保存一路走到当前节点的所有节点，stack.peek() 一直指向 iterator 指向的当前节点。因此判断有没有下一个，只需要判断 stack 是否为空 获得下一个值，只需要返回 stack.peek() 的值，并将 stack 进行相应的变化，挪到下一个点。

挪到下一个点的算法如下：

1. 如果当前点存在右子树，那么就是右子树中“一路向西”最左边的那个点
2. 如果当前点不存在右子树，则是走到当前点的路径中，第一个左拐的点

访问所有节点用时O(n)，所以均摊下来访问每个节点的时间复杂度时O(1)

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
public class BSTIterator {
    private Stack<TreeNode> stack = new Stack<>();

    // @param root: The root of binary tree.
    public BSTIterator(TreeNode root) {
        while (root != null) {
            stack.push(root);
            root = root.left;
        }
    }

    // @return: True if there has next node, or false
    public boolean hasNext() {
        return !stack.isEmpty();
    }

    // @return: return next node
    public TreeNode next() {
        TreeNode curt = stack.peek();
        TreeNode node = curt;

        // move to the next node
        if (node.right == null) {
            node = stack.pop();
            while (!stack.isEmpty() && stack.peek().right == node) {
                node = stack.pop();
            }
        } else {
            node = node.right;
            while (node != null) {
                stack.push(node);
                node = node.left;
            }
        }

        return curt;
    }
}
```

👍 获赞 20

💬 22 条评论

你的口袋题库

2000+ 算法真题、国内外名企题库免费开放



九章算法APP



令狐冲

更新于 11/9/2020, 5:59:25 AM

比较常用的写法。初始化的时候把 stack 构造好。

python

java

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None

Example of iterate a tree:
iterator = BSTIterator(root)
while iterator.hasNext():
    node = iterator.next()
    do something for node
"""

class BSTIterator:
    """
    @param: root: The root of binary tree.
    """
    def __init__(self, root):
        self.stack = []
        while root != None:
            self.stack.append(root)
            root = root.left

    """
    @return: True if there has next node, or false
    """
    def hasNext(self):
        return len(self.stack) > 0

    """
    @return: return next node
    """
    def next(self):
        node = self.stack[-1]
        if node.right is not None:
            n = node.right
            while n != None:
                self.stack.append(n)
                n = n.left
        else:
            n = self.stack.pop()
            while self.stack and self.stack[-1].right == n:
                n = self.stack.pop()

        return node
```

👍 获赞 13

💬 8 条评论



讲座

更新于 6/9/2020, 7:03:47 AM

Tree Iterator

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
public class BSTIterator {
    private Stack<TreeNode> stack = new Stack<>();
    TreeNode next = null;

    void AddNodeToStack(TreeNode root) {
        while (root != null) {
            stack.push(root);
            root = root.left;
        }
    }

    // @param root: The root of binary tree.
    public BSTIterator(TreeNode root) {
        next = root;
    }

    // @return: True if there has next node, or false
    public boolean hasNext() {
        if (next != null) {
            AddNodeToStack(next);
            next = null;
        }
        return !stack.isEmpty();
    }

    // @return: return next node
    public TreeNode next() {
        if (!hasNext()) {
            return null;
        }
        TreeNode cur = stack.pop();
        next = cur.right;
        return cur;
    }
}
```

👍 获赞 4

💬 添加评论



令狐冲

更新于 8/16/2020, 1:05:04 AM

更简洁的代码

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None

Example of iterate a tree:
iterator = BSTIterator(root)
while iterator.hasNext():
    node = iterator.next()
    do something for node
"""

class BSTIterator:
    """
    @param: root: The root of binary tree.
    """
    def __init__(self, root):
        dummy = TreeNode(0)
        dummy.right = root
        self.stack = [dummy]
        self.next()

    """
    @return: True if there has next node, or false
    """
    def hasNext(self):
        return bool(self.stack)

    """
    @return: return next node
    """
    def next(self):
        node = self.stack.pop()
        next_node = node
        if node.right:
            node = node.right
            while node:
                self.stack.append(node)
                node = node.left
        return next_node
```

👍 获赞 3

💬 4 条评论



令狐冲

更新于 6/9/2020, 7:03:48 AM

这是一个非常通用的利用 stack 进行 Binary Tree Iterator 的写法。

stack 中保存一路走到当前节点的所有节点, stack.peek() 一直指向 iterator 指向的当前节点。因此判断有没有下一个, 只需要判断 stack 是否为空 获得下一个值, 只需要返回 stack.peek() 的值, 并将 stack 进行相应的变化, 挪到下一个点。

挪到下一个点的算法如下:

1. 如果当前点存在右子树, 那么就是右子树中“一路向西”最左边的那个点
2. 如果当前点不存在右子树, 则是走到当前点的路径中, 第一个左拐的点

访问所有节点用时 $O(n)$, 所以均摊下来访问每个节点的时间复杂度为 $O(1)$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None

Example of iterate a tree:
iterator = BSTIterator(root)
while iterator.hasNext():
    node = iterator.next()
    do something for node
"""

class BSTIterator:
    # @param root: The root of binary tree.
    def __init__(self, root):
        self.stack = []
        self.curt = root

    # @return: True if there has next node, or false
    def hasNext(self):
        return self.curt is not None or len(self.stack) > 0

    # @return: return next node
    def next(self):
        while self.curt is not None:
            self.stack.append(self.curt)
            self.curt = self.curt.left

        self.curt = self.stack.pop()
        nxt = self.curt
        self.curt = self.curt.right
        return nxt
```

👍 获赞 3

💬 1 条评论



助教-咖啡蛇

更新于 12/19/2020, 9:23:38 AM

C++

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
```

```

*/
/**
 * Definition of Doubly-ListNode
 * class DoublyListNode {
 * public:
 *     int val;
 *     DoublyListNode *next, *prev;
 *     DoublyListNode(int val) {
 *         this->val = val;
 *         this->prev = this->next = NULL;
 *     }
 * } * Definition of TreeNode:
 * class TreeNode {
 * public:
 *     int val;
 *     TreeNode *left, *right;
 *     TreeNode(int val) {
 *         this->val = val;
 *         this->left = this->right = NULL;
 *     }
 * }
 */

class Solution {
public:
    /**
     * @param root: The root of tree
     * @return: the head of doubly list node
     */
    stack<TreeNode*> Stack;
    vector<TreeNode*> order;
    DoublyListNode * bstToDoublyList(TreeNode * root) {
        // write your code here
        if (root == nullptr) {
            return nullptr;
        }

        while(root != nullptr){
            Stack.push(root);
            root = root->left;
        }

        DoublyListNode* dummy = new DoublyListNode(0);
        DoublyListNode* last = dummy;

        while(!Stack.empty()) {
            TreeNode* node = Stack.top();
            last->next = new DoublyListNode(node->val);
            last->next->prev = last;
            last = last->next;
            if (node->right != nullptr) {
                TreeNode* temp = node->right;
                while (temp != nullptr) {
                    Stack.push(temp);
                    temp = temp->left;
                }
            } else {
                TreeNode* temp = Stack.top();
                Stack.pop();
                while (!Stack.empty() && temp == Stack.top()->right) {
                    temp = Stack.top();
                    Stack.pop();
                }
            }
        }

        dummy->next->prev = NULL;
        return dummy->next;
    }
}

```



```
}  
};
```

👍 获赞 0

💬 添加评论



令狐冲

更新于 6/9/2020, 7:04:29 AM

这是一个非常通用的利用 stack 进行 Binary Tree Iterator 的写法。

stack 中保存一路走到当前节点的所有节点, stack.peek() 一直指向 iterator 指向的当前节点。因此判断有没有下一个, 只需要判断 stack 是否为空 获得下一个值, 只需要返回 stack.peek() 的值, 并将 stack 进行相应的变化, 挪到下一个点。

挪到下一个点的算法如下:

1. 如果当前点存在右子树, 那么就是右子树中“一路向西”最左边的那个点
2. 如果当前点不存在右子树, 则是走到当前点的路径中, 第一个左拐的点

访问所有节点用时 $O(n)$, 所以均摊下来访问每个节点的时间复杂度时 $O(1)$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
/**
 * Definition for binary tree
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class BSTIterator {
public:
    stack<TreeNode*> myStack;
    TreeNode *current;

    BSTIterator(TreeNode *root) {
        while (!myStack.empty()) {
            myStack.pop();
        }
        current = root;
    }

    /** @return whether we have a next smallest number */
    bool hasNext() {
        return (current != NULL || !myStack.empty());
    }

    /** @return the next smallest number */
    TreeNode* next() {
        while (current != NULL) {
            myStack.push(current);
            current = current->left;
        }
        current = myStack.top(); myStack.pop();
        TreeNode *nxt = current;
        current = current->right;
        return nxt;
    }
};

/**
 * Your BSTIterator will be called like this:
 * BSTIterator i = BSTIterator(root);
 * while (i.hasNext()) cout << i.next();
 */
```

👍 获赞 0

💬 1 条评论



九章用户KEQVTZ

更新于 12/27/2020, 8:45:03 AM

和九章的方法差不多但是 next() 的写法更清晰明了

解析: 每次 next 的时候把 stack 的头拿出来, 如果这个节点有右子树的话, 把右子树里的左边这条 path 全都给加到 stack 里面。(BST 节点的是顺序是这样的) 比如:

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */

    5
   / \
  2   8
 /   /
1   6
```

上面这个树，在 initialize 的时候我们得到节点5，然后把左边这一条 path 所有节点 [5,2,1] 全都放进去。next 的时候的顺序就会是 1->2->5 当再次走到5的时候，发现有右子树。所以我们将右子树的左边 path [8,6] 就丢进去。然后5之后的 next 就是 5->6->8

如果有帮助请同学们麻烦点个赞！谢谢！

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
/*
 * @param root: The root of binary tree.
 */
public BSTIterator(TreeNode root) {
    // do initialization if necessary
    stack = new Stack<>();
    while (root != null) {
        stack.push(root);
        root = root.left;
    }
}

Stack<TreeNode> stack;

/*
 * @return: True if there has next node, or false
 */
public boolean hasNext() {
    return !stack.isEmpty();
}

/*
 * @return: return next node
 */
public TreeNode next() {
    if (stack.isEmpty())
        return null;

    TreeNode node = stack.pop();
    // push all left path of right subtree
    TreeNode right = node.right;
    while (right != null) {
        stack.push(right);
        right = right.left;
    }

    return node;
}
```

👍 获赞 38

💬 5 条评论



明尼苏达大角羊

更新于 8/4/2020, 9:44:12 PM

与老师在课上的思路和复杂度一样, $O(1)$ 的平均时间, 简化了next()中的一些算法。

【确认条件】 (1) 沟通BST的定义。 (2) 确认元素是升序遍历的。 (3) 输入格式与异常值检测。

【解题思路】 (1) 初始化时从root开始一路向左走到最末端的节点, 即全局最小值, 并将一路访问过的节点加入到stack中。 (2) 实现hasNext(): stack-1 () 一直存放 iterator 指向的当前节点。因此在判断有没有下一个节点时, 只需要判断 stack 是否为空。 (3) 实现next(): 弹出当前stack的栈顶元素(即所求的next节点), 并将stack 进行相应的变化, 移动到下一个点。找到下一个点的算法为: 如果当前点存在右子树, 那么下一节点就是其右子树中一路向左走到底的那个点, 并且需要把路上经过的节点加入stack (因为这些节点都比下一个节点大, 尚未访问过); 如果当前点不存在右子树, 则弹出当前节点后新的栈顶 (如果不为空的话) 自然就是下一个节点。

(4) 特别注意: 为什么在实现next()时, 算法看起来和九章的标准答案有些不同? 因为课上的算法是: 取得当前栈顶stack-1 ()作为返回元素。如果当前点存在右子树, 那么下一节点就是其右子树中一路向左走到底的那个点, 并且需要把路上经过的节点加入stack; 如果当前点不存在右子树, 则沿着栈顶向前回溯到第一个左节点, 同时将路过的元素全部弹出。(这些点全部小于当前点, 而且回溯已完成, 不需再保留。)

我们可以看出差别: 九章课上的算法中, 在当前元素含有右子树时, 并不会将当前元素弹出, 而是暂时保留在stack中; 而只有当前节点不含右子树, 需要向前回溯最近左节点时, 才统一弹出。因此stack往往即保留了一些尚未访问的节点, 也保留了一些访问过但回溯时需要用到的节点。

在本题的解法中, stack中永远只保留尚未访问的节点, 每次获取下一节点时, 无论是否有左右子树, 当前节点都会被弹出。正如解题思路中解释的: “如果当前点不存在右子树, 则弹出当前节点后新的栈顶 (如果不为空的话) 自然就是下一个节点。”因此省去了专门回溯的过程。和课上的解法相比, 复杂度是一样的, 就是省了几行字。

更详细的英文参考可以看这篇文章: <https://leetcode.com/articles/binary-search-tree-iterator/> (<https://leetcode.com/articles/binary-search-tree-iterator/>)

【实现要点】 (1) 初始化时直接建立stack, 并将iterator移动到最小值位置。(2) 实现next()时, 注意先寄存pop()出来的当前节点, 再移动stack, 最后返回之前寄存的当前节点。(3) 如思路中解释的, 只需处理当前节点包含右子树的情况; 不包含右子树的情况不需要额外处理。(不含右子树时, _to_next_min()会直接返回)

(4) 可以把“找到最左位置并把路过节点加入stack”的操作封装进子函数。

【复杂度】 时间复杂度: hasNext()为O(1); next()平均为O(1), 因为遍历整棵树的时间为O(n) 空间复杂度: O(h)

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class BSTIterator:
    def __init__(self, root):
        self.stack = []
        self._to_next_min(root)

    def hasNext(self):
        return len(self.stack) > 0

    def next(self):
        if len(self.stack) == 0:
            return None
        next_node = self.stack.pop()
        self._to_next_min(next_node.right)
        return next_node

    def _to_next_min(self, root):
        while root:
            self.stack.append(root)
            root = root.left
```

👍 获赞 6

💬 1 条评论



zhqfox

更新于 7/24/2020, 7:39:55 AM

使用Morris Traversal, 在遍历的时候, 利用predecessor的右空指针, 将其指向当前节点。平均时间复杂度O(1), 空间O(1)。参考链接:
https://en.wikipedia.org/wiki/Threaded_binary_tree (https://en.wikipedia.org/wiki/Threaded_binary_tree)

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
public class BSTIterator {
    TreeNode pre;
    TreeNode current;
    /**
     * @param root: The root of binary tree.
     */
    public BSTIterator(TreeNode root) {
        current = root;
    }

    /**
     * @return: True if there has next node, or false
     */
    public boolean hasNext() {
        // write your code here
        return current != null;
    }

    /**
     * @return: return next node
     */
    public TreeNode next() {
        TreeNode res = null;
        while (current != null) {
            if (current.left == null) { // current has reached the left most node
                res = current; // find the required node
                current = current.right; // update current and break
                break;
            } else {
                pre = current.left;
                while (pre.right != null && pre.right != current) {
                    pre = pre.right;
                }
                if (pre.right == null) { // pre.right has not yet linked to its successor
                    pre.right = current; // link to current
                    current = current.left; // update current and continue;
                } else {
                    pre.right = null; // pre.right = current, therefore, set it to null to restore the tree
                    res = current; // find the required node.
                    current = current.right; // update the current and break;
                    break;
                }
            }
        }
        return res;
    }
}
```

👍 获赞 6

💬 添加评论

**xxw289**

更新于 6/9/2020, 7:03:46 AM

这个解法跟@九章算法发的第一个解法基本一致, 只不过.next()函数稍有不同。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
public class BSTIterator {
    /**
     * @param root: The root of binary tree.
     */
    Stack<TreeNode> stack;
    public BSTIterator(TreeNode root) {
        // do initialization if necessary
        stack = new Stack<TreeNode>();
        if (root == null) return;
        while (root != null) {
            stack.push(root);
            root = root.left;
        }
    }

    /**
     * @return: True if there has next node, or false
     */
    public boolean hasNext() {
        // write your code here
        return !stack.isEmpty();
    }

    /**
     * @return: return next node
     */
    public TreeNode next() {
        // write your code here
        TreeNode cur = stack.pop();
        if (cur.right != null) {
            TreeNode leftChild = cur.right;
            while (leftChild != null) {
                stack.push(leftChild);
                leftChild = leftChild.left;
            }
        }
        return cur;
    }
}
```

👍 获赞 5

💬 4 条评论

**Tin**

更新于 11/25/2020, 6:48:32 PM

要点: 栈顶总是放着 Next 最小 node, 取next操作时间复杂性就是取successor。

另, 原题有误导, 除非原树是平衡二叉树, next 不可能是O(1)

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class BSTIterator:

    def __init__(self, root):
        self.stack = []
        while root:
            self.stack.append(root)
            root = root.left

    def hasNext(self, ):

        return self.stack

    def next(self, ):

        if self.stack:
            root = self.stack.pop()
            node = root.right
            while node:
                self.stack.append(node)
                node = node.left
            return root
```

👍 获赞 4

💬 添加评论

[加载更多题解](#)

进阶课程

视频+互动

直播+互动

直播+互动

互动课

九章算法班 2021 版

8周时间精通 57 个核心高频考点, 9 招击破 FLAG、BATJ 算法面试。22...

系统架构设计 System Design 2021 版

成为百万架构师必上。30 课时带你快速掌握 18 大系统架构设计知识点与面...

九章算法面试高频题冲刺班

每期更新 15% 题目, 考前押题, 一举拿下 FLAG & BATJ Offer

面向对象设计 OOD

应届生及亚马逊面试必考, IT 求职必备基础

首页 (/?skip_redirect=true)


联系我们 (mailto:info@jiuzhang.com)


加入我们 (/joinus)

Copyright © 2013-2020 九章算法 浙ICP备19045946号-1

(http://www.miibeian.gov.cn/)

商务合作: fukesu@jiuzhang.com (mailto:fukesu@jiuzhang.com)

 (http://weibo.com/ninechapter)

 知

(https://www.zhihu.com/people/crackinterview/)

(/)