# 题目

实现一个二叉搜索树迭代器。你将使用二叉搜索树的根节点初始化迭代器。

调用 next() 将返回二叉搜索树中的下一个最小的数。

**示例：**

BSTIterator iterator = new BSTIterator(root);

iterator.next(); // 返回 3

iterator.next(); // 返回 7

iterator.hasNext(); // 返回 true

iterator.next(); // 返回 9

iterator.hasNext(); // 返回 true

iterator.next(); // 返回 15

iterator.hasNext(); // 返回 true

iterator.next(); // 返回 20

iterator.hasNext(); // 返回 false

**提示：**

next() 和 hasNext() 操作的时间复杂度是 O(1)，并使用 O(h) 内存，其中 h 是树的高度。

你可以假设 next() 调用总是有效的，也就是说，当调用 next() 时，BST 中至少存在一个下一个最小的数。

# 分析

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode(int x) : val(x), left(NULL), right(NULL) {}

\* };

\*/

class BSTIterator {

public:

vector <TreeNode \*> Vec;

BSTIterator(TreeNode\* root) {

while (root != nullptr) {

Vec.push\_back(root);

root = root -> left;

}

return;

}

/\*\* @return the next smallest number \*/

int next() {

TreeNode \*tmp = Vec.back();

int ret = tmp -> val;

Vec.pop\_back();

TreeNode \*res = tmp -> right;

while(res != nullptr) {

Vec.push\_back(res);

res = res -> left;

}

return ret;

}

/\*\* @return whether we have a next smallest number \*/

bool hasNext() {

if (!Vec.empty()) {

return true;

}

return false;

}

};

/\*\*

\* Your BSTIterator object will be instantiated and called as such:

\* BSTIterator\* obj = new BSTIterator(root);

\* int param\_1 = obj->next();

\* bool param\_2 = obj->hasNext();

\*/