# 二叉查找树中查找第k小节点

## 题目介绍

给定一棵二叉搜索树，请找出其中的第k小的结点。

例如， （5，3，7，2，4，6，8） 中，按结点数值大小顺序第三小结点的值为4。

## 思路分析

**理论知识：按照中序遍历的顺序遍历一棵二叉查找树，则遍历序列的数值是递增排序的**。

因此，该题目就是按照中序遍历二叉搜索树即可。按照中序遍历的顺序k逐渐减1，直到k==1，即返回该节点即可。

## Java代码

### 递归方式

首先展示递归的中序遍历：

/\*\*

\* 递归的中序遍历

\*/

public void inOrderTraversal(TreeNode root){

**if(root == null) return;**

inOrderTraversal(root.left);

System.out.println("val = " + root.val);

inOrderTraversal(root.right);

}

**对照，编写第k小节点的Java代码：**

TreeNode KthNode(TreeNode pRoot, int k){

return KthNode( pRoot, new int[]{k});

}

/\*\*

\* BinaryTree

\* 给定一棵二叉搜索树，请找出其中的第k小节点。

\* 例如， （5，3，7，2，4，6，8） 中，

\* 按节点数值大小顺序第三小节点的值为4。

\* 递归思想：首先从左子树中寻找，若返回非null，表示寻找到，直接返回该节点；

\* 否则表示左子树中没有该节点；然后判断中间节点，即k是否为1(并减1)，是的话就返回当前节点；

\* 否则判断右子树。

\* 注意：二叉树中的任意一个节点都是中间节点，即使叶子节点也有两个null左右子树，

\* 因此二叉树中的任意一个节点都会经过k的判断，且经过1个，k就减1，正好找到那个k为1的节点。

\*/

TreeNode KthNode(TreeNode pRoot, int[] k){

**//递归终止条件**

**if(pRoot == null||k[0] < 1) return null;**

//首先遍历左子树,若结果不为null，则返回；否在往下执行

**TreeNode result = KthNode(pRoot.left,k);**

**if(result != null) return result;**

**//遍历中节点**

**if(k[0]-- == 1) return pRoot;**

**//遍历右节点**

**return KthNode(pRoot.right,k);**

}

### 非递归方式：栈

/\*\*

\* 非递归方式：利用栈和while循环

\* @param pRoot

\*/

TreeNode KthNode(TreeNode pRoot, int k){

return KthNodeWhile( pRoot, new int[]{k});

}

public TreeNode KthNodeWhile(TreeNode root,int[] k){

Deque<TreeNode> stack = new ArrayDeque<TreeNode>();

while(!(stack.isEmpty()&&root == null)){

if(root != null){

stack.push(root);

root = root.left;

}else{

root = stack.pop();

**if(k[0]-- == 1) return root;**

root = root.right;

}

}

return null;

}