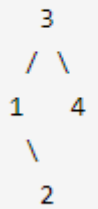


第10题——二叉搜索树的第K大结点

给定一棵二叉搜索树，请找出其中第k大的节点。

示例 1:

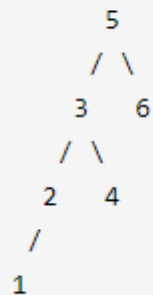
输入: root = [3,1,4,null,2], k = 1



输出: 4

示例 2:

输入: root = [5,3,6,2,4,null,null,1], k = 3



输出: 4

限制: $1 \leq k \leq$ 二叉搜索树元素个数

递归法 (1)

解题思路

以 右->根->左 的方式遍历整棵树，将值存进数组，返回第 k-1 下标的数组元素即可

```
void func(struct TreeNode* root,int *size,int res[])
{
    if(root!=NULL)
    {
        func(root->right,size,res);    //优先递归访问右子树
        res[(*size)++]=root->val;      //访问结点的值，并存储到res
        func(root->left,size,res);     //递归访问左子树
        // ..... //可以将逆序遍历的结果存入数组
    }
}

int kthLargest(struct TreeNode* root,int k)
{
    int *res=malloc(sizeof(int)*10000); //堆区开辟int*10000的空间
    int size=0;
```

```
func(root,&size,res);//调用函数
return res[k-1];    //返回数组中的第k项
}
```

递归法 (2)

这个方法更加节省内存，不需要预先开辟10000的空间。

而是把K传入，再递归逆序遍历，

1. 递归遍历右子树

2. k=k-1，然后判断其是否为0，如果为0，将当前结点的值取出

3. 递归遍历左子树

如此一来，函数调用完毕之后，res的值就已经是我们要找的值了。

```
void func(struct TreeNode* root,int *val,int *res)
{
    if(root!=NULL)
    {
        func(root->right,val,res);    //优先递归访问右子树
        if(--(*val)==0)
        {
            return *res=root->val;//找到值，返回
        }
        func(root->left,val,res);    //递归访问左子树
    }
}

int kthLargest(struct TreeNode* root,int k)
{
    int val=k;
    int res=0;
    func(root,&val,&res);//调用函数
    return res;
}
```