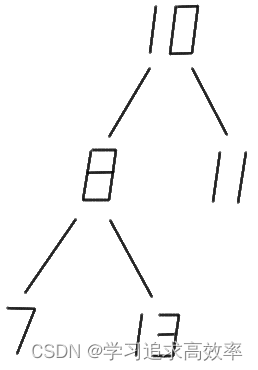
|  |  |
| --- | --- |
| **1** | **判别二叉树是否为二叉查找树（C++）** |

常用解决方法：

1. 利用二叉排序树的性质，左孩子结点小于根结点，右孩子结点大于根结点（注意：根结点的左半部分要全部小于根结点，右孩子类似）

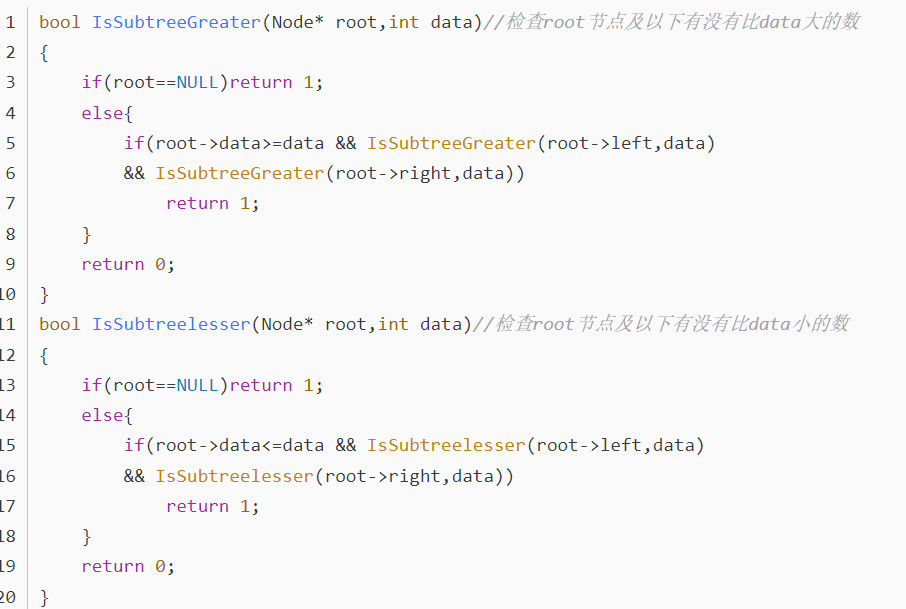
程序错误的话，下面这个图你会认为是二叉排序树



（2）对二叉树进行中序遍历，如果中序遍历是递增的也可以说明此二叉树是排序二叉树

核心代码参考





|  |  |
| --- | --- |
| **2** | **二叉查找树的后序遍历** |

输入一个整数数组，判断该数组是不是某二元查找树的后序遍历的结果。如果是返回true，否则返回false。

5 7 6 9 11 10 8

True

后序遍历，最后一个是根，所以8是根

8之前的序列可以分成2部分，1-6是比8小，9-10是比8大

继续判断6为根 前面5和7是分别比它小和大的树，10为根，9和11是它的左子树和右子树

int is\_BST(int num[], int len) {

int root = num[len - 1];

int lt\_len = 0, rt\_len = 0;

int i = 0;

while (i < len - 1) {

if (num[i] >= root)

break;

i++;

}

lt\_len = i;

rt\_len = len - lt\_len - 1;

while (i < len - 1) {

if (num[i] < root)

return 0;

i++;

}

get\_subtree(num, 0, lt\_len);

if (lt\_len > 1)

if (!is\_BST(subtree, lt\_len))

return 0;

get\_subtree(num, lt\_len, rt\_len);

if (rt\_len > 1)

if (!is\_BST(subtree, rt\_len))

return 0;

return 1;

}

简答题

1.

a

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  |  | 68 |  |  |  | 72 | **28** | 96 | **63** | 43 |

96%11=8 43%11=10 72%11=6 68%11=2 63%11=8（->9） 28%11=6（->7） 粗体表示有冲突

成功的平均查找次数：

（1次成功\*4+ 2次成功\*2）/ 成功的元素个数 = 8/6=4/3

(b) 上述最多查找2次，所以只有+1的跳跃，没有-1的跳跃，结果与（a)同

c）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | **28** | 68 |  |  |  | 72 | **63** | 96 |  | 43 |

96%11=8 43%11=10 72%11=6 68%11=2

63%11=8( 8号位冲突 63/11=5 (8+5)%11(注意这里的11代表表长)=2 2号位置冲突 （8+2\*5）%11(表长)=7 共比较3次)

28%11=6(6号位冲突 28/11=2 （6+2)%11=8 8号位置冲突 （6+2\*2）%11=10 10号位置冲突 （6+3\*2）%11=1 共比较4次)

成功的平均成功的查找次数：

（1次查找成功\*4+ 3次查找成功\*1+4次查找成功\*1）/ 成功的元素个数 =11/6

d.

|  |  |
| --- | --- |
| **0** |  |
| **1** |  |
| **2** | 68 |
| **3** |  |
| **4** |  |
| **5** |  |
| **6** | 72,28 |
| **7** |  |
| **8** | 96,63 |
| **9** |  |
| **10** | 43 |

查找成功： （1\*4+2\*2）/6=8/6=4/3

2.

多种解法：

（1）两个指针，一个在头，一个在尾；大则减，小则加。

（2） 遍历到x，查看k-x是不是存在

。。。。。。。

3.

**创建平衡二叉树**

依次输入关键字20，40，30，10，50，70，100，90，110，80，画出创建的平衡二叉树。

