数据结构作业(第十二次)

PB20111686 黄瑞轩

9.31

储存结构：

```
class Node {
public:
    int val = 0;
    Node* left = nullptr;
    Node* right = nullptr;
};
```

判定算法：

```
bool IsRegular(Node* root) {
    if (root == nullptr) {
        return true;
    }
    if (root->left) {
        if (root->left->val > root->val) {
            return false;
        }
    }
    if (root->right) {
        if (root->right->val < root->val) {
            return false;
        }
    }
    return IsRegular(root->left) & IsRegular(root->right);
}
```

9.33

要找到根不小于x，左孩子小于x的这个根，并记录此根的地址，此步最坏用时$O(\log_2 n)$。

```
Node* Find(Node* root, int x) {
    if (root) {
        if (root->val < x) {
            return Find(root->right, x);
        }
        if (root->val >= x) {
            if (root->left) {
                if (root->left->val < x) {
                    return root;
                }
                else {
                    return Find(root->left, x);
                }
            }
            else return nullptr;
        }
    }
    return nullptr;
}
```

中序遍历，判断条件改进，只会访问m个节点，时间复杂度是$O(m)$。

```
void InorderTraverse(Node* root, Node* stop) {
    if (root == stop || root == nullptr) return;
    if (root->left != nullptr) InorderTraverse(root->left, stop);
    cout << root->val << endl;
    if (root->right != nullptr) InorderTraverse(root->right, stop);
    return;
}
```

主调函数：

```
void Neuf_33(Node* root, int x) {
    auto t = Find(root, x);
    InorderTraverse(root, t);
    return;
}
```

二叉排序树的非递归查找算法

```
Node* Find(Node* root, int x) {
    Node* t = root;
    while (t != nullptr) {
        if (t->val == x) {
            return t;
        }
        else if (t->val < x) {
            t = t->right;
        }
        else {
            t = t->left;
        }
    }
    return nullptr;
}
```

9.11

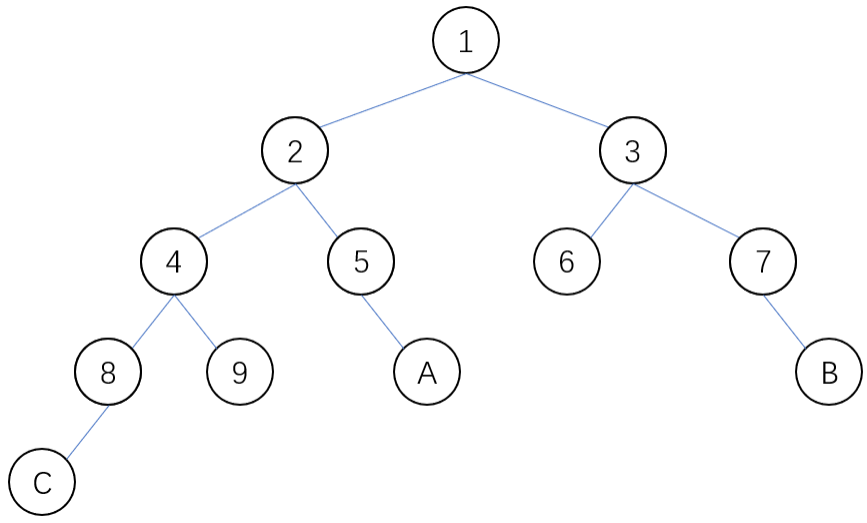
用类似递归的思路，设深度为n的平衡二叉树最少节点数为$N(n)$，将两棵子树合并，则

$$N(n) = N(n-1) + N(n-2) + 1$$

又$N(1) = 1, N(2) = 2$，则可以列表得出：

$$N(3) = 4, N(4) = 7, N(5) = 12, N(6) = 20$$

所以12个节点的平衡二叉树的最大深度是5，一个示例如下：



9.38

先写一个插入算法：

```
void Insert(Node* root, Node* x) {
    if (root) {
        if (root->val == x->val) return;
        else if (root->val > x->val) {
            if (root->left) {
                if (root->left->val < x->val) {
                    auto tmp = new Node(x->val);
                    tmp->left = root->left;
                    root->left = tmp;
                    return;
                }
                Insert(root->left, x);
            }
            else {
                root->left = new Node(x->val);
            }
        }
        else {
            if (root->right) {
```

```
                if (root->right->val > x->val) {
                    auto tmp = new Node(x->val);
                    tmp->right = root->right;
                    root->right = tmp;
                    return;
                }
                Insert(root->right, x);
            }
            else {
                root->right = new Node(x->val);
            }
        }

    }
    return;
}
```

合并两棵二叉树，这里不修改T，只修改了S，最后合并的结果为S。

```
void Merge(Node* S, Node* T) {
    if (T) Insert(S, T);
    if (T->left) Merge(S, T->left);
    if (T->right) Merge(S, T->right);
    return;
}
```

9.40

算法如下：

```
Node* Findk(Node* root, int k) {
    if (root) {
        if (k == root->lsize) {
            return root;
        }
        else if (k < root->lsize) {
            return Findk(root->left, k);
        }
        else {
            return Findk(root->right, k - root->lsize);
        }
    }
    return nullptr;
}
```