数据结构作业(第八次)

PB20111686 黄瑞轩

6.68

节点:

```
class BTNode {
public:
    int val = 0;
    BTNode* FirstChild = nullptr;
    BTNode* NextSibling = nullptr;

    BTNode();
    BTNode(int v) {
        this->val = v;
    }
}
```

算法:

```
BTNode* Child_Siblingfy(int val[], int degree[], int size) {
    //节点缓存
    vector<BTNode*> NodeBuff;
    //预处理度数序列
    for (int i = 1; i < size; i++) degree[i]--;
    //创建结点
    for (int i = 0; i < size; i++) NodeBuff.push_back(new BTNode(val[i]));
    //父亲Travel指针
    int ptr_parent = 0;
    //计数器，表示ptr_parent找到了第i个孩子
    int cnt_child = 0;
    for (int i = 1; i < size; i++) {
        //为缓存中每一个节点找前驱
        if (degree[ptr_parent] == 0 || cnt_child == degree[ptr_parent]) {
            //如果这个父亲没有孩子，或者其孩子已被找完
            cnt_child = 0;
            ptr_parent++;
            //这个节点已经不可能是后面任何一个NodeBuff[i]的父亲
            continue;
        }
        // 这个父亲有孩子，即i是ptr_parent的孩子
        cnt_child++;
        if (cnt_child == 1)
            NodeBuff[ptr_parent]->FirstChild = NodeBuff[i];
        else
            NodeBuff[i - 1]->NextSibling = NodeBuff[i];
        //第i个孩子是第i-1个孩子的右兄弟
    }
    return NodeBuff[0];
}
```

6.71

```
void helper(BTNode* Node, int SpaceNum) {
    for (int i = 0; i < SpaceNum; i++)cout << ' ';
    cout << Node->val;
    if (Node->FirstChild) helper(Node->FirstChild, SpaceNum + 1);
    if (Node->NextSibling) helper(Node->NextSibling, SpaceNum);
    return;
}
// 调用下面这个函数
void Print_Child_Sibling_Tree(BTNode* root) {
    if (root) helper(root, 0);
    return;
}
```