《数据结构与算法》第十四周作业反馈

助教-陈孙一硕

A 判断是否二叉搜索树

先根据前序遍历建树，然后中序遍历判断是否为升序序列即可。

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
string s;
char x;
int idx=0;
struct node{
    char val;
    node* lchild=NULL,*rchild=NULL;
};
node* make(){
    //if (idx==s.size()) return NULL;
    x=s[idx++];
    if (x=='*') return NULL;
    node *nd=new node;
    nd->val=x;
    nd->lchild=make();
    nd->rchild=make();
    return nd;
}
char pre='0';
bool flag=1;
void DFS(node* nd){
    if (!flag) return;
```

```
    if (nd==NULL) return;
    DFS(nd->lchild);
    if (nd->val<=pre){cout<<"NO"<<endl;flag=0;}
    pre=nd->val;
    DFS(nd->rchild);
}
void solve(){
    pre='0'; flag=1; idx=0;
    node* root=make();
    DFS(root);
    if (flag) cout<<"YES"<<endl;
}
signed main(){
    while (cin>>s) solve();
    return 0;
}
```

B 二叉搜索树的删除操作

按函数功能一一进行封装，封装 del 函数时候需要特别注意递归删除的逻辑。

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
struct node{
    int val;
    node* lchild=NULL,*rchild=NULL;
};
void ist(node* root,int x){
    node* tmp=root;
    node *nd=new node;
    nd->val=x;
    while (true){
        if (x>tmp->val){
            if (tmp->rchild==NULL) {tmp->rchild=nd;break;}
            else tmp=tmp->rchild;
```

```
        }else{
            if (tmp->lchild==NULL) {tmp->lchild=nd;break;}
            else tmp=tmp->lchild;
        }
    }
}
void del(node* root,int x){
    node* tar=root,*pre=NULL;
    while (tar->val!=x){
        pre=tar;
        if (x>tar->val) tar=tar->rchild;
        else tar=tar->lchild;
    }
    if (tar->lchild==NULL&&tar->rchild==NULL){
        if (tar==pre->lchild) pre->lchild=NULL;
        else pre->rchild=NULL;
    }else if (tar->lchild!=NULL){
        node* tmp=tar->lchild;
        while (tmp->rchild!=NULL) tmp=tmp->rchild;
        int tp=tmp->val;
        del(tar,tmp->val);
        tar->val=tp;
    }else{
        node* tmp=tar->rchild;
        while (tmp->lchild!=NULL) tmp=tmp->lchild;
        int tp=tmp->val;
        del(tar,tmp->val);
        tar->val=tp;
    }
}
void levelorder(node* root){
    queue<node*> q;
    q.push(root);
    while (!q.empty()){
        node* t=q.front();
        q.pop();
```

```
        if (t==NULL) continue;
        if (t!=root) cout<<" ";
        cout<<t->val;
        q.push(t->lchild),q.push(t->rchild);
    }
}
signed main(){
    int n,k,x;
    cin>>n;
    node* root=new node;
    cin>>x;
    root->val=x;
    for (int i=2;i<=n;i++){
        cin>>x;
        ist(root,x);
    }
    cin>>k;
    while (k--) cin>>x,del(root,x);
    levelorder(root);
    return 0;
}
```