16题

前序：`abcdefghijkl`

中序：`aefdcgihjklb`

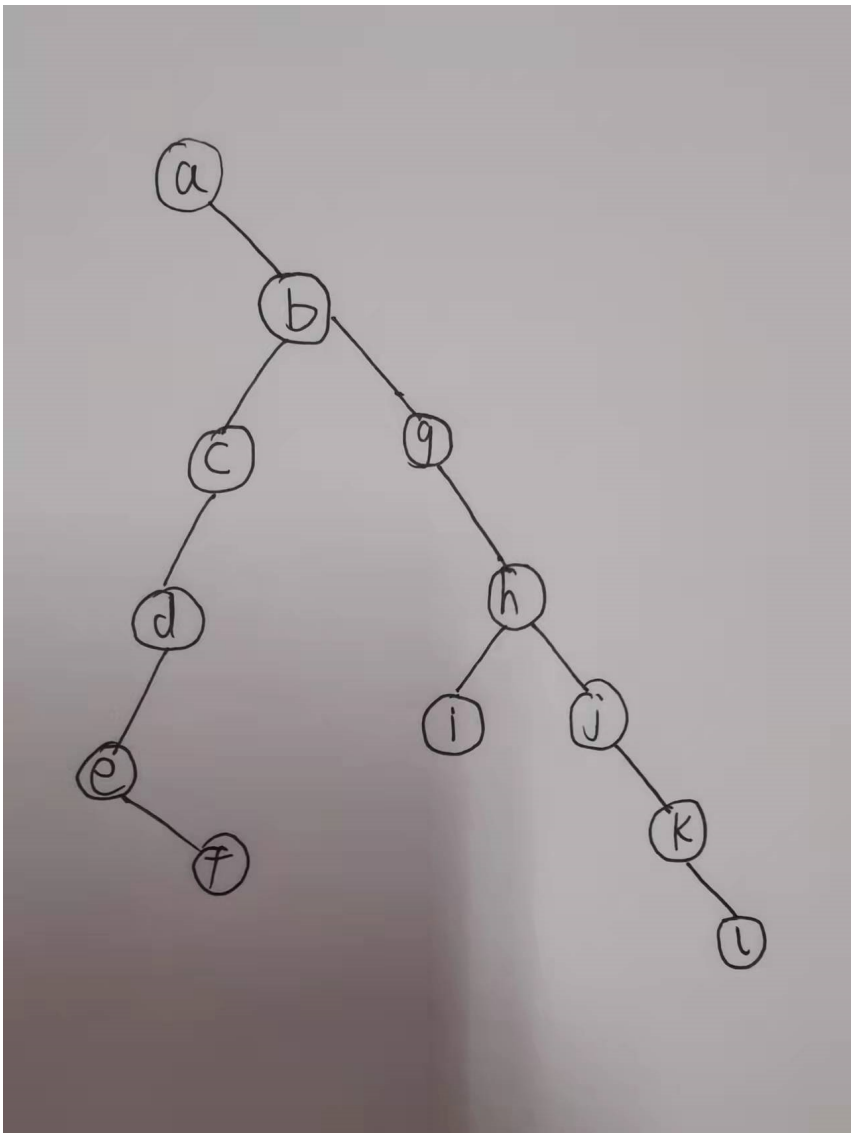
前序是根左右，中序是左根右，因此前序遍历的第一个字母就是根，再从中序中找到根的位置，左边是左子树，右边是右子树，递归调用就可以构造出来二叉树

后序:`fedilkjhgcba`

知道了中序遍历的结果，我们用一个队列去存储，就可以得到层次遍历的结果

层次：`abcdgehfijkl`

树：



18题

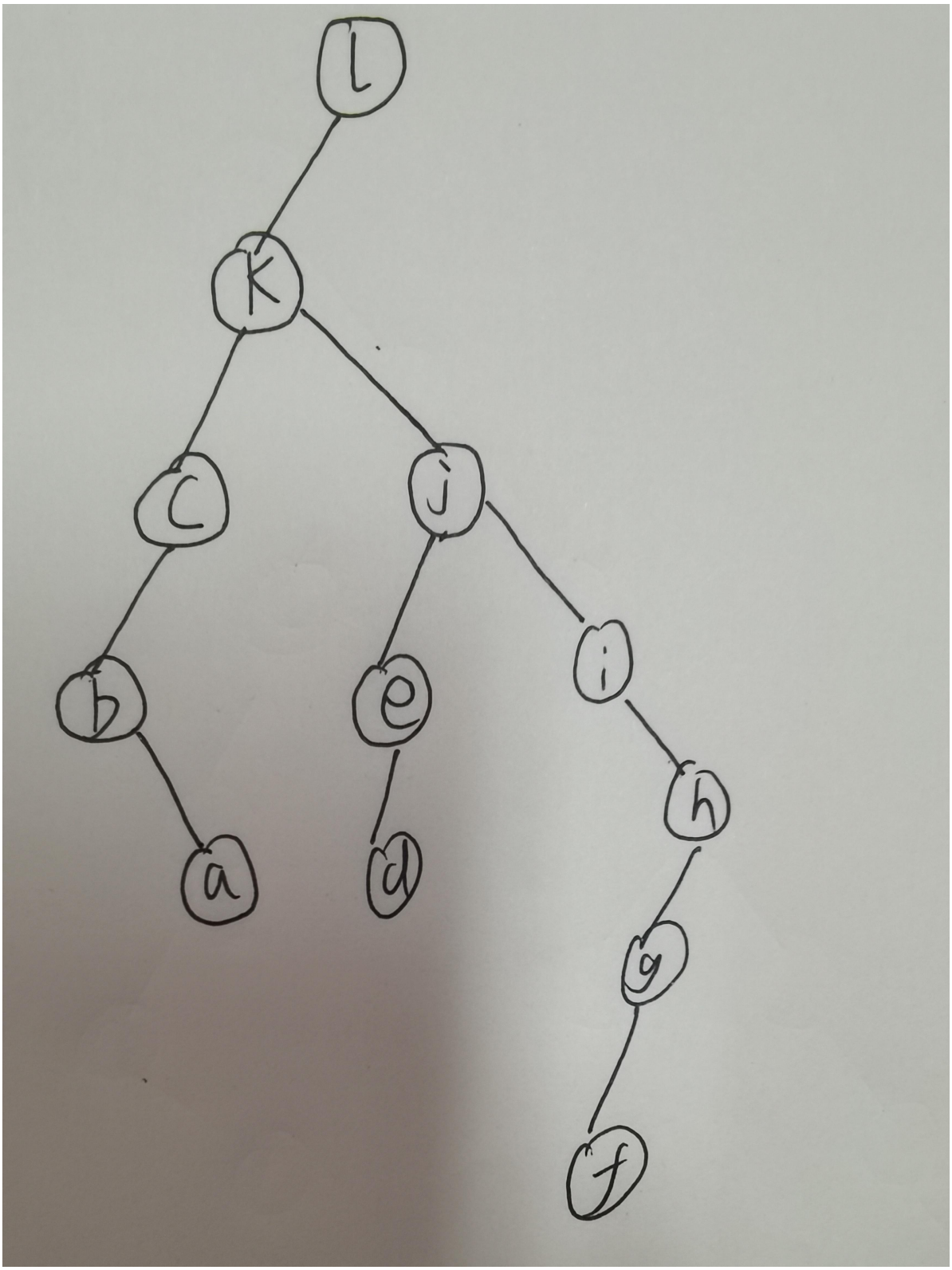
可以发现，这道题的思路其实与上一道题完全相同，都是先找到根的位置，再找到左子树与右子树

再递归调用进行构造

中序: `backdejifghl`

后序: `abcdefghijkl`

树:



前序: `lkcbajedihgf`

知道了中序遍历的结果，我们用一个队列去存储，就可以得到层次遍历的结果

层次: `lkcjbeiadhgf`

20题

```
template<class T>
void Preorder(pair<bool, T> a[], int last, int id)
{
    if (id <= last && a[i].first == true)//有效
    {
        cout << a[i].second << " ";//输出
        Preorder(a, last, 2 * i);//左孩子
        Preorder(a, last, 2 * i + 1);//右孩子
    }
}
```

时间复杂度：二叉树的遍历，每个结点经过且经过`2`次，（一次去，一次回来）

所以时间复杂度为`O(n)`

27题

```
template<class T>
int Height(binaryTreeNode<T>* sourceNode)
{
    if (sourceNode)
    {
        int l_height = Height(sourceNode->left);//左孩子
        int r_height = Height(sourceNode->right);//右孩子
        return max(l_height, r_height) + 1;//高度等于左右子树的最高的高度+1
    }
    return 0;//如果是空，那么高度为0
}
```

28题

```
template<class T>
int getNode(binaryTreeNode<T>* sourceNode)
{
    if (sourceNode)
    {
        int l_Node = Height(sourceNode->left);//左孩子
        int r_Node = Height(sourceNode->right);//右孩子
        return l_Node+ r_Node + 1;//个数等于左右子树的个数和+1
    }
    return 0;//如果是空，那么个数为0
}
```

29题

```cpp
template<class T>
int getMaxnode(binaryTreeNode<T>* sourceNode)
{
    int n = getNode(sourceNode);
    int* cnt = new int[n];
    queue<Node<T>* >q;
    q.push(Node{sourceNode,0});//压入root
    a[0]++;//第0层+1
    while (!q.empty())
    {
        Node<T>* front = q.front();
        if (front->ptr->left)//左结点不空
        {
            q.push(Node{ front->ptr->left ,level+1});
            a[front->ptr->left->level]++;//左孩子的Level++
        }
        if (front->ptr->right)//右结点不空
        {
            q.push(Node{ front->ptr->right ,level + 1 });
            a[front->ptr->right->level]++;//右孩子的Level++
        }
        q.pop();
    }
    int max = 0;
    int level = 0;
    for (int i = 0; i < n; i++)
    {
        if (a[i] > max)
        {
            level = i;//level保存结点最多的层数
            max = a[i];
        }
    }

```

33题

```cpp
Node* makeTree(vector<int> pre, vector<int> vin) {
    if (pre.empty() || vin.empty())
        return NULL;

    vector<int> l1, l2, r1, r2;
    Node* root = (Node*)malloc(sizeof(Node));
    int x = pre[0];
    int p = 0;
    while (vin[p] != x)//找到根结点
        p++;

    for (int i = 1; i <= p; i++)
        l1.push_back(pre[i]);//左子树前序
    for (int i = 0; i <= p - 1; i++)
        l2.push_back(vin[i]);//左子树中序
    for (int i = p + 1; i < pre.size(); i++)
        r1.push_back(pre[i]);//右子树前序
    for (int i = p + 1; i < pre.size(); i++)
```

```
        r2.push_back(vin[i]);//右子树中序

    root->val = x;
    cout << x << endl;//暑促
    root->left = makeTree(l1, l2);//递归调用
    root->right = makeTree(r1, r2);

    return root;
}
```

45题

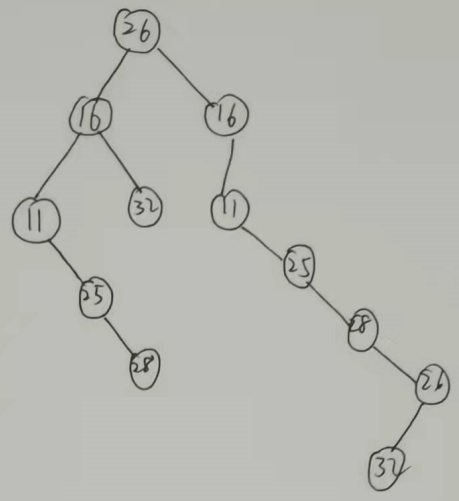
```
template<class E>
bool linkedBinaryTree<E>::compare(binaryTreeNode<E>* t,binaryTreeNode<E>*p)
{
    if (!t && !this->ptr) return true;//都空，那么相等
    if (!t || !this->ptr) return false;//一个空，那么不等
    if (t->element != this->element) return false;//两个都不空，但是值不相等，那么返回false
    return compare(t->left, this->left) && compare(t->right, this->right);//如果以上都不是，//那么需要看一下左子树和右子树的结果
}
```

时间复杂度:`O(n)`

56题

11-17

由题意、



11-19