HW7 林宸昊 PB20000034

p29 p30

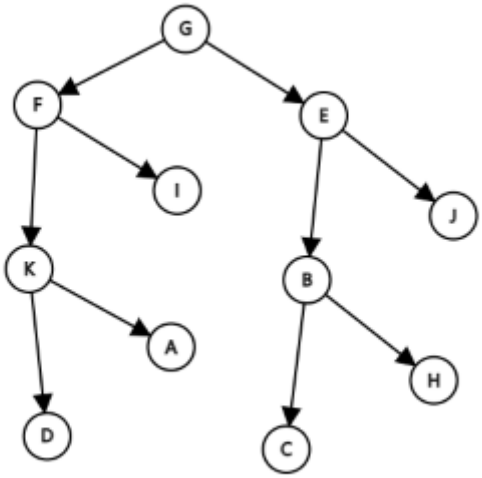
- 先序遍历的非递归算法

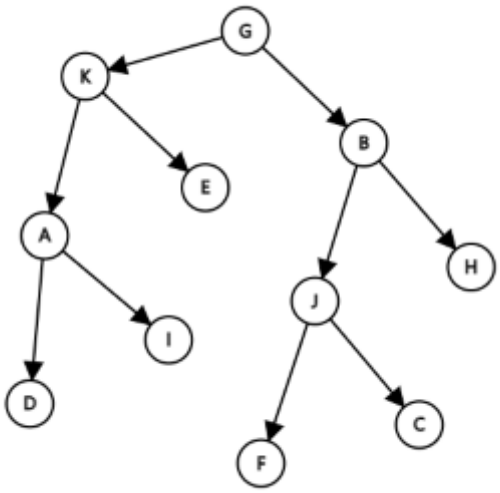
```
Status PreOrderTraverse(BiTree T, Status(*Visit)(TElemtype e)){
    InitStack(S);
    Push(S, T);//根节点入栈
    while(!StackEmpty(S)){
        GetTop(S, p);
        Visit(p->data); //访问根节点
        Pop(S, p);      //根节点出栈
        //按顺序依次以右节点左节点的顺序入栈，保证每一次访问都是先左后右
        if(p->rchild)
            Push(S, p->rchild);
        if(p->lchild)
            Push(S, p->lchild);
    }
    return ok;
}
```

- 后序遍历的非递归算法

```
Status PostOrderTraverse(BiTree T, Status(*Visit)(TElemtype e)){
    InitStack(S);
    Push(S, T);
    while(!StackEmpty(S)){
        q = NULL;
        while(GetTop(S, p) && p){
            if(p->rchild != q){
                Push(S, p->rchild);
            }
            else {
                Push(S, NULL);
                break;
            }
            Push(S, p->lchild);
        }
        Pop(S, p);
        if(!StackEmpty(S)){
            Pop(S, q);
            Visit(q->data);
        }
    }
}
```

6.23





6.71

- 注意到从上往下为先序遍历，并且打印时前面空格数等于结点所在层数。

```
void print(Tree t, int i){//i初始值应为1
    if(t){
        //输出空格和字符
        for(int j = 1; j < i; j++) printf(" ");
        printf(t->data);
        putchar(10);
        print(t->firstchild, i + 1); //打印孩子，层数加一
        print(t->nextsibiling, i); //打印兄弟，层数不变
    }
}
```