

本节内容

二叉树的 线索化

王道考研/CSKAOYAN.COM

1

知识总览

二叉树线索化

中序线索化

先序线索化

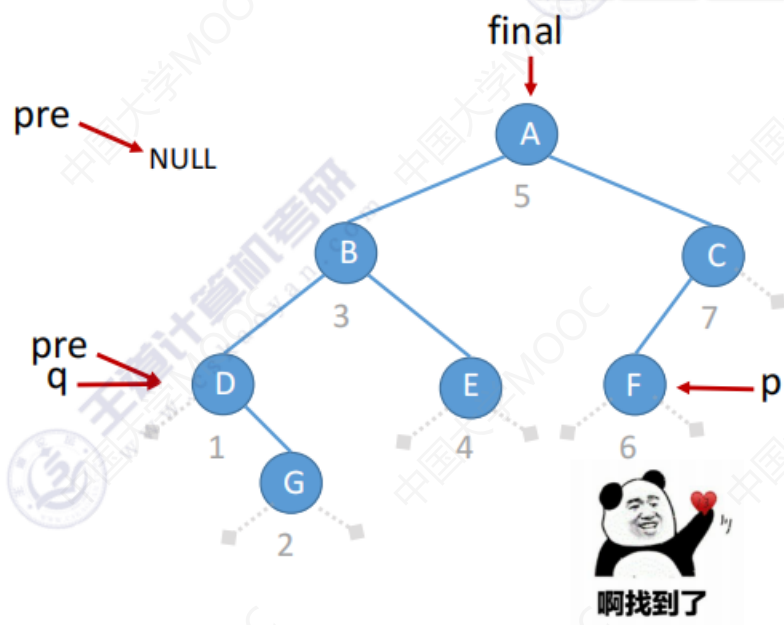
后序线索化

王道考研/CSKAOYAN.COM

2

用土办法找到中序前驱

最好改一个函数名，如 findPre



中序遍历序列: D G B E A F C

```
//中序遍历
```

```
void InOrder(BiTree T){
    if(T!=NULL){
        InOrder(T->lchild);    //递归遍历左子树
        visit(T);              //访问根结点
        InOrder(T->rchild);    //递归遍历右子树
    }
}

//访问结点q
void visit(BiTNode * q){
    if (q==p)                //当前访问结点刚好是结点p
        final = pre;        //找到p的前驱
    else
        pre = q;             //pre指向当前访问的结点
}
```

//辅助全局变量,用于查找结点p的前驱

```
BiTNode *p;           //p指向目标结点
BiTNode *pre=NULL;    //指向当前访问结点的前驱
BiTNode *final=NULL;  //用于记录最终结果
```

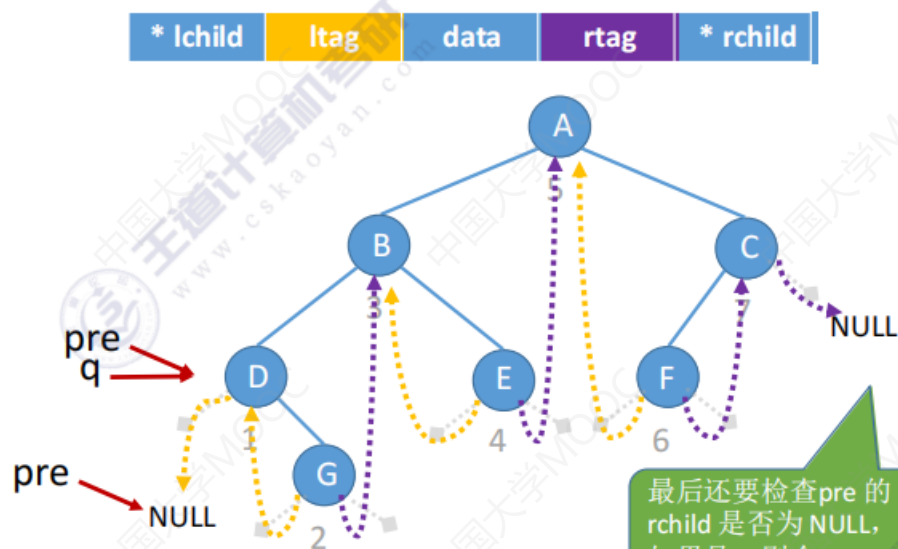
王道考研/CSKAQYAN.COM

王道考研/CSKAOYAN.COM

3

中序线索化

初步建成的树,
ltag、rtag=0



```
//线索二叉树结点
```

```
typedef struct ThreadNode{
    ElemType data;
    struct ThreadNode *lchild,*rchild;
    int ltag,rtag;           //左、右线索标志
}ThreadNode,* ThreadTree;
```

```
//中序遍历二叉树，一边遍历一边线索化
```

```
void InThread(ThreadTree T){
    if(T!=NULL){
        InThread(T->lchild);    //中序遍历左子树
        visit(T);               //访问根节点
        InThread(T->rchild);    //中序遍历右子树
    }
}
```

```
void visit(ThreadNode *q) {
    if(q->lchild==NULL){ //左子树为空，建立前驱线索
        q->lchild=pre;
        q->ltag=1;
    }
    if(pre!=NULL&&pre->rchild==NULL){
        pre->rchild=q; //建立前驱结点的后继线索
        pre->rtag=1;
    }
    pre=q;
}
```

//全局变量 *pre*, 指向当前访问结点的前驱

```
ThreadNode *pre=NULL;
```

王道考研/CSKAOYAN.COM

4

初步建成的树,
ltag、rtag=0

中序线索化

* lchild

ltag

data

rtag

* rchild

//全局变量 pre, 指向当前访问结点的前驱
ThreadNode *pre=NULL;

//中序线索化二叉树T
void CreateInThread(ThreadTree T){
 pre=NULL; //pre初始为NULL
 if(T!=NULL){ //非空二叉树才能线索化
 InThread(T); //中序线索化二叉树
 if (pre->rchild==NULL)
 pre->rtag=1; //处理遍历的最后一个结点
 }

 //线索二叉树结点
 typedef struct ThreadNode{
 ElemType data;
 struct ThreadNode *lchild,*rchild;
 int ltag,rtag; //左、右线索标志
 }ThreadNode,* ThreadTree;
}

//中序遍历二叉树, 一边遍历一边线索化
void InThread(ThreadTree T){
 if(T!=NULL){
 InThread(T->lchild); //中序遍历左子树
 visit(T); //访问根节点
 InThread(T->rchild); //中序遍历右子树
 }
}

void visit(ThreadNode *q) {
 if(q->lchild==NULL){//左子树为空, 建立前驱线索
 q->lchild=pre;
 q->ltag=1;
 }
 if(pre!=NULL&&pre->rchild==NULL){
 pre->rchild=q; //建立前驱结点的后继线索
 pre->rtag=1;
 }
 pre=q;
}

王道考研/CSKAOYAN.COM

5

中序线索二叉树

对应tag位为0时, 表示
指针指向其孩子

对应tag位为1时,
表示指针是“线索”

王道考研/CSKAOYAN.COM

6

中序线索化（王道教材版）

```
//中序线索化
void InThread(ThreadTree p, ThreadTree &pre){
    if(p!=NULL){
        InThread(p->lchild, pre); //递归，线索化左子树
        if(p->lchild==NULL){ //左子树为空，建立前驱线索
            p->lchild=pre;
            p->ltag=1;
        }
        if(pre!=NULL&&pre->rchild==NULL){ //建立前驱结点的后继线索
            pre->rchild=p;
            pre->rtag=1;
        }
        pre=p;
        InThread(p->rchild, pre);
    } //if(p!=NULL)
}
```

思考：为什么 pre 参数是引用类型？

思考：处理遍历的最后一个结点时，为什么没有判断 rchild 是否为NULL？

答：中序遍历的最后一个结点右孩子指针必为空。

```
//中序线索化二叉树T
void CreateInThread(ThreadTree T){
    ThreadTree pre=NULL;
    if(T!=NULL){ //非空二叉树，线索化
        InThread(T, pre); //线索化二叉树
        pre->rchild=NULL; //处理遍历的最后一个结点
        pre->rtag=1;
    }
}
```

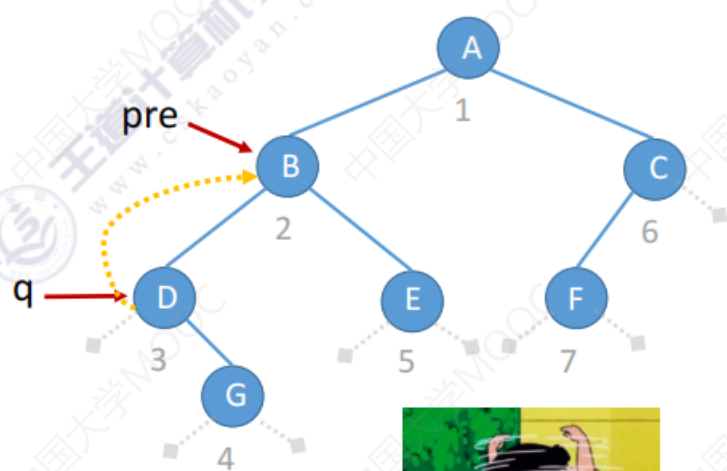
王道考研/CSKAOYAN.COM

7

先序线索化

初步建成的树，
ltag、rtag=0

* lchild ltag data rtag * rchild



爱滴魔力转圈圈

//先序遍历二叉树，一边遍历一边线索化

```
void PreThread(ThreadTree T){
    if(T!=NULL){
        visit(T); //先处理根节点
        PreThread(T->lchild);
        PreThread(T->rchild);
    }
}

void visit(ThreadNode *q) {
    if(q->lchild==NULL){ //左子树为空，建立前驱线索
        q->lchild=pre;
        q->ltag=1;
    }
    if(pre!=NULL&&pre->rchild==NULL){
        pre->rchild=q; //建立前驱结点的后继线索
        pre->rtag=1;
    }
    pre=q;
}
```

//全局变量 pre，指向当前访问结点的前驱

ThreadNode *pre=NULL;

王道考研/CSKAOYAN.COM

8

初步建成的树, ltag、rtag=0

先序线索化

| * lchild | ltag | data | rtag | * rchild |
|----------|------|------|------|----------|
|----------|------|------|------|----------|

```
//先序遍历二叉树，一边遍历一边线索化
void PreThread(ThreadTree T){
    if(T!=NULL){
        visit(T); //先处理根节点
        PreThread(T->lchild);
        PreThread(T->rchild);
    }
}
```

```
//先序遍历二叉树，一边遍历一边线索化
void PreThread(ThreadTree T){
    if(T!=NULL){
        visit(T); //先处理根节点
        if (T->ltag==0) //lchild不是前驱线索
            PreThread(T->lchild);
        PreThread(T->rchild);
    }
}
```

王道考研/CSKAOYAN.COM

9

初步建成的树, ltag、rtag=0

先序线索化

| * lchild | ltag | data | rtag | * rchild |
|----------|------|------|------|----------|
|----------|------|------|------|----------|

```
//全局变量 pre，指向当前访问结点的前驱
ThreadNode *pre=NULL;

//先序线索化二叉树T
void CreatePreThread(ThreadTree T){
    pre=NULL; //pre初始为NULL
    if(T!=NULL){ //非空二叉树才能线索化
        PreThread(T); //先序线索化二叉树
        if (pre->rchild==NULL)
            pre->rtag=1; //处理遍历的最后一个结点
    }
}
```

```
//先序遍历二叉树，一边遍历一边线索化
void PreThread(ThreadTree T){
    if(T!=NULL){
        visit(T); //先处理根节点
        if (T->ltag==0) //lchild不是前驱线索
            PreThread(T->lchild);
        PreThread(T->rchild);
    }
}
```

```
void visit(ThreadNode *q) {
    if(q->lchild==NULL){ //左子树为空，建立前驱线索
        q->lchild=pre;
        q->ltag=1;
    }
    if(pre!=NULL&&pre->rchild==NULL){
        pre->rchild=q; //建立前驱结点的后继线索
        pre->rtag=1;
    }
    pre=q;
}
```

王道考研/CSKAOYAN.COM

10

先序线索化（王道教材Style）

```
//先序线索化
void PreThread(ThreadTree p, ThreadTree &pre){
    if(p!=NULL){
        if(p->lchild==NULL){           //左子树为空，建立前驱线索
            p->lchild=pre;
            p->ltag=1;
        }
        if(pre!=NULL&&pre->rchild==NULL){
            pre->rchild=p;           //建立前驱结点的后继线索
            pre->rtag=1;
        }
        pre=p;                       //标记当前结点
        if(p->ltag==0)                //左子树不为空
            PreThread(p->lchild, pre); //递归，先序遍历左子树
        PreThread(p->rchild, pre);    //递归，先序遍历右子树
    } //if(p!=NULL)
}
```



```
//先序线索化二叉树T
void CreatePreThread(ThreadTree T){
    ThreadTree pre=NULL;
    if(T!=NULL){
        PreThread(T, pre); //非空二叉树，线索化
        if(pre->rchild==NULL) //处理遍历的最后一个结点
            pre->rtag=1;
    }
}
```

王道考研/CSKAOYAN.COM

11

后序线索化

初步建成的树，
ltag、rtag=0

* lchild ltag data rtag * rchild

//全局变量 pre，指向当前访问结点的前驱
ThreadNode *pre=NULL;

```
//后序线索化二叉树T
void CreatePostThread(ThreadTree T){
    pre=NULL; //pre初始为NULL
    if(T!=NULL){ //非空二叉树才能线索化
        PostThread(T); //后序线索化二叉树
        if(pre->rchild==NULL) //处理遍历的最后一个结点
            pre->rtag=1;
    }
}
```

```
//后遍历二叉树，一边遍历一边线索化
void PostThread(ThreadTree T){
    if(T!=NULL){
        PostThread(T->lchild); //后序遍历左子树
        PostThread(T->rchild); //后序遍历右子树
        visit(T); //访问根结点
    }
}
```

```
void visit(ThreadNode *q){
    if(q->lchild==NULL){ //左子树为空，建立前驱线索
        q->lchild=pre;
        q->ltag=1;
    }
    if(pre!=NULL&&pre->rchild==NULL){
        pre->rchild=q; //建立前驱结点的后继线索
        pre->rtag=1;
    }
    pre=q;
}
```

王道考研/CSKAOYAN.COM

12

后序线索化（王道教材Style）

//后序线索化

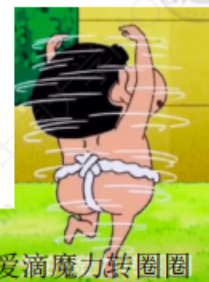
```
void PostThread(ThreadTree p, ThreadTree &pre){
    if(p!=NULL){
        PostThread(p->lchild, pre); //递归，线索化左子树
        PostThread(p->rchild, pre); //递归，线索化右子树
        if(p->lchild==NULL){ //左子树为空，建立前驱线索
            p->lchild=pre;
            p->ltag=1;
        }
        if(pre!=NULL && pre->rchild==NULL){
            pre->rchild=p;
            pre->rtag=1;
        }
        pre=p;
    } //if(p!=NULL)
}
```

//后序线索化二叉树T

```
void CreatePostThread(ThreadTree T){
    ThreadTree pre=NULL;
    if(T!=NULL){ //非空二叉树，线索化
        PostThread(T, pre); //线索化二叉树
        if(pre->rchild==NULL) //处理遍历的最后一个结点
            pre->rtag=1;
    }
}
```



不存在的



爱滴魔力转圈圈

王道考研/CSKAOYAN.COM

13

知识回顾与重要考点

二叉树线索化

中序线索化 ⊖ 得到中序线索二叉树

先序线索化 ⊖ 得到先序线索二叉树

后序线索化 ⊖ 得到后序线索二叉树

核心

中序/先序/后序遍历算法的改造，当访问一个结点时，连接该结点与前驱结点的线索信息
用一个指针 pre 记录当前访问结点的前驱结点

易错点

最后一个结点的 rchild、rtag 的处理

先序线索化中，注意处理爱滴魔力转圈圈问题，当 ltag==0 时，才能对左子树先序线索化

王道考研/CSKAOYAN.COM

14