数据结构

深圳技术大学
大数据与互联网学院

线索二叉树

一. 建立二叉树线索的方法

- 遍历二叉树是按一定的规则将树中的结点排列成一个线性序列，即是对非线性结构的线性化操作
  - 需要找到遍历过程中动态得到的每个结点的直接前驱和直接后继
  - 如何保存这些前驱后继信息

线索二叉树

一. 建立二叉树线索的方法

- 方法一：增加新指针是在每个结点中增加前驱(fwd)和后继(bkwd)指针，在做二叉树遍历（前、中、后序），将每个结点的前驱和后继信息添入fwd和bkwd域中

fwd	lChild	data	rChild	bkwd

- 方法二：设一棵二叉树有**n**个结点，则有**n-1**条边**(**指针连线**)**， 而**n**个结点共有**2n**个指针域**(Lchild**和**Rchild)**，显然有**n+1**个空闲指针域未用。在有n个结点的二叉树中，必定存在n+1个空链域，因为每个结点有两个链域（左、右孩子指针），因此共有2n个链域，除根结点外，每个结点都有且仅有一个分支相连，即n-1个链域被使用。所以这种方法无需增加额外空间

线索二叉树

一. 建立二叉树线索的方法

- 方法二：利用空指针是指在结点中增加两个标记位（LTag, RTag）
  - LTag=0, lChild域指示结点的左孩子
  - LTag=1, lChild域指示结点的前驱结点
  - RTag=0, lChild域指示结点的右孩子
  - RTag=1, lChild域指示结点的后继结点

LTag	lChild	data	rChild	RTag

线索二叉树

二. 线索二叉树的实现

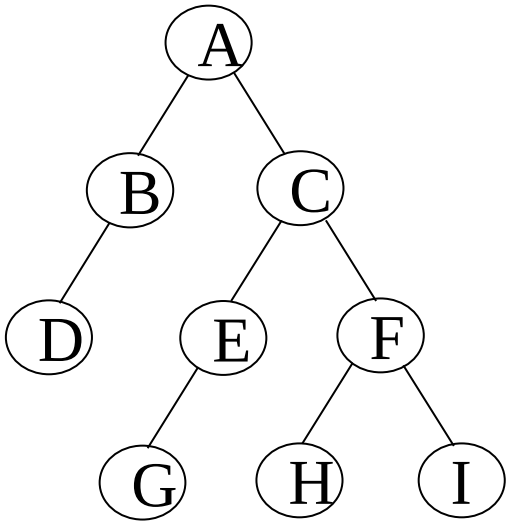
- 使用方法二，指向结点前驱和后继的指针叫做线索，加上线索的二叉树称之为线索二叉树

```
typedef enum PointTag {Link, Thread};
typedef struct BiThrNode
{
    ElemType  data;
    struct BiTreeNode *Lchild , *Rchild ;
    int  Ltag , Rtag ;
} BiThrNode ;
```

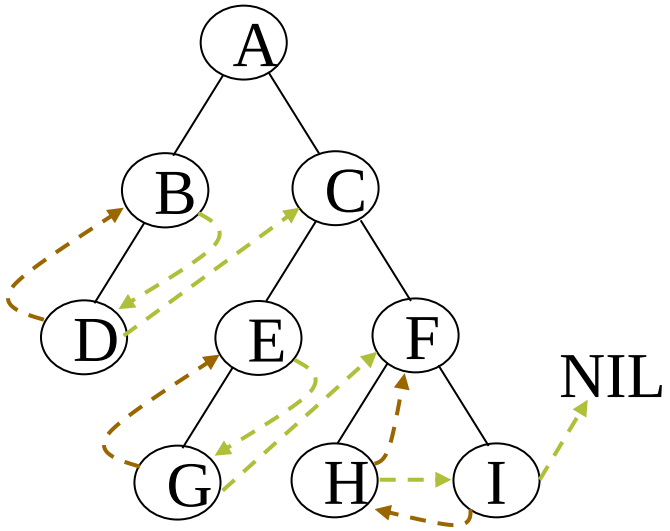
线索二叉树

二. 线索二叉树的实现

- 不同遍历方法产生不同的线索二叉树



(a) 二叉树

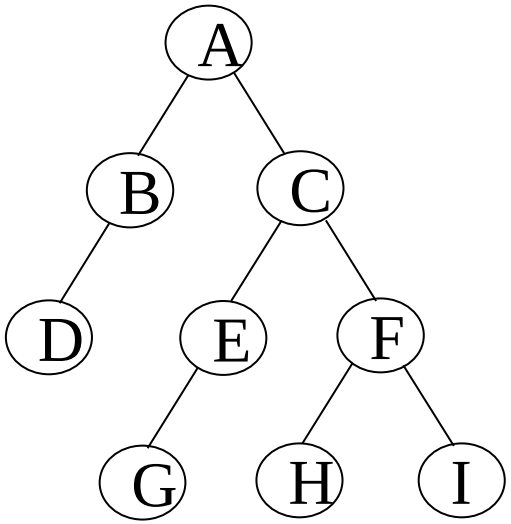


(b) 先序线索树的逻辑形式
结点序列：ABDCEGFHI

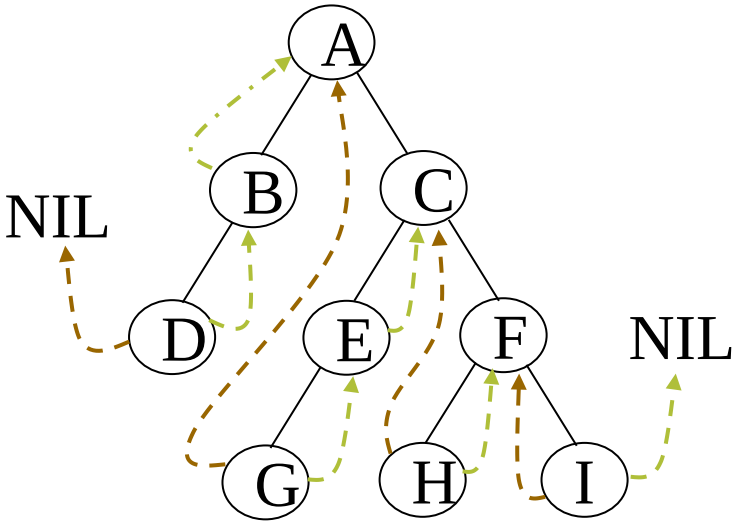
线索二叉树

二. 线索二叉树的实现

- 不同遍历方法产生不同的线索二叉树



(a) 二叉树

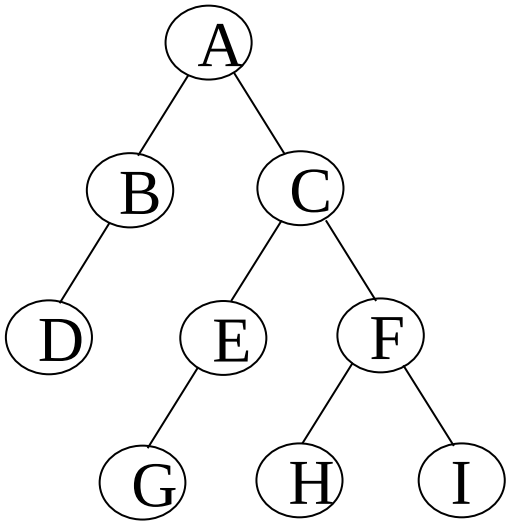


(c) 中序线索树的逻辑形式
结点序列：DBAGECHFI

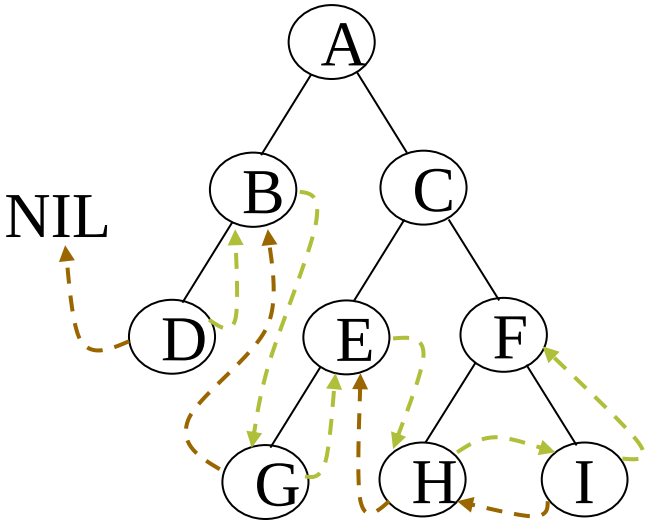
线索二叉树

二. 线索二叉树的实现

- 不同遍历方法产生不同的线索二叉树



(a) 二叉树



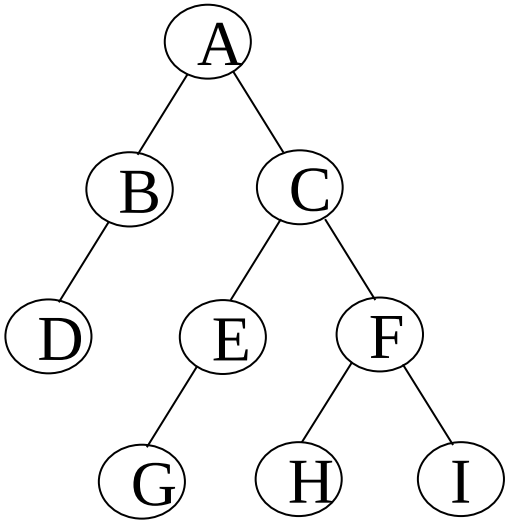
(d) 后序线索树的逻辑形式
结点序列：DBGEHIFCA

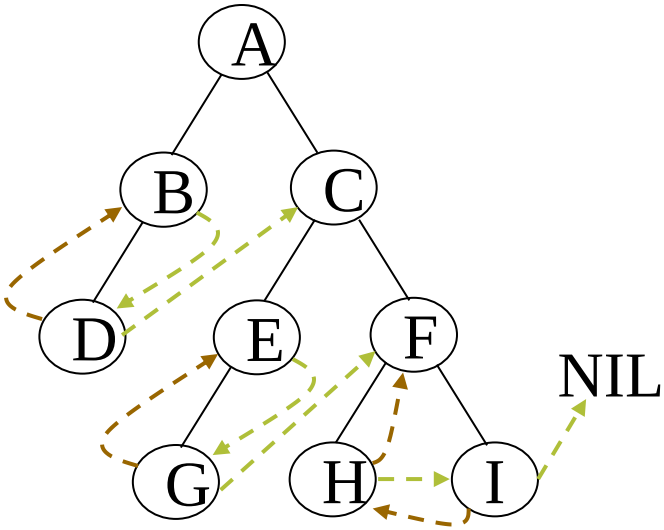
线索二叉树

三. 线索二叉树的实现

- 线索二叉树的先序遍历-非递归

```
void preorder_Thread(BiThrNode *T)
{  BiThrNode  *p=T ;
   while (p!=NULL)
    {  visit(p->data) ;
       if (p->Ltag==0)  p=p->Lchild ;
       else  p=p->Rchild
    }
}
```

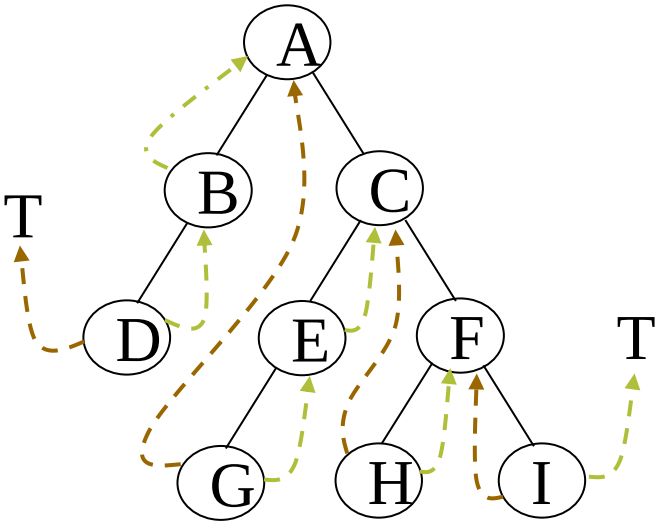




线索二叉树

三. 线索二叉树的实现

- 线索二叉树的中序遍历-非递归
  - 结点T是一个头结点
  - T的左孩子指向根节点
  - T的右孩子指向中序遍历的尾节点

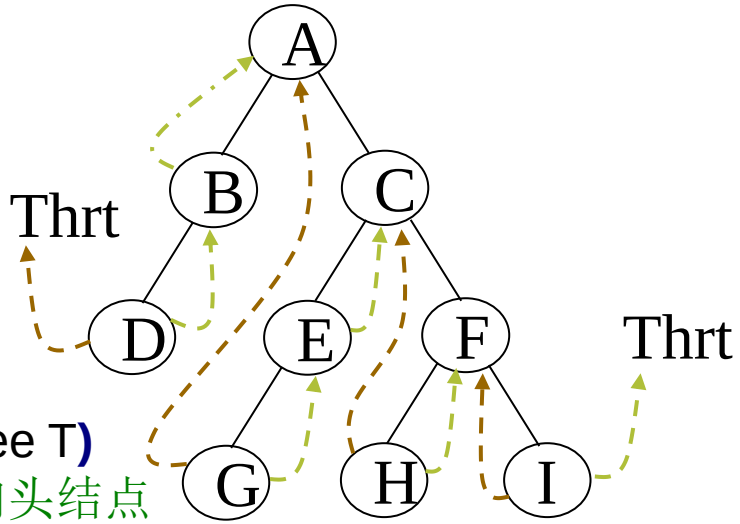



```
Status InOrder (BiThrTree T) {
  BiThrTree p;
  p = T->lchild;                    // p指向根结点
  while (p != T) {   // 空树或遍历结束时，p==T
    while (p->LTag==Link) p = p->lchild;
    cout<<p->data;
    while (p->RTag==Thread && p->rchild!=T) {
      p = p->rchild;
      cout<<p->data;
    }
    p = p->rchild;  // p进至其右子树根
  }
  return OK;
}
```

线索二叉树

三. 线索二叉树的实现

- 线索二叉树按中序遍历构建
- 程序部分I

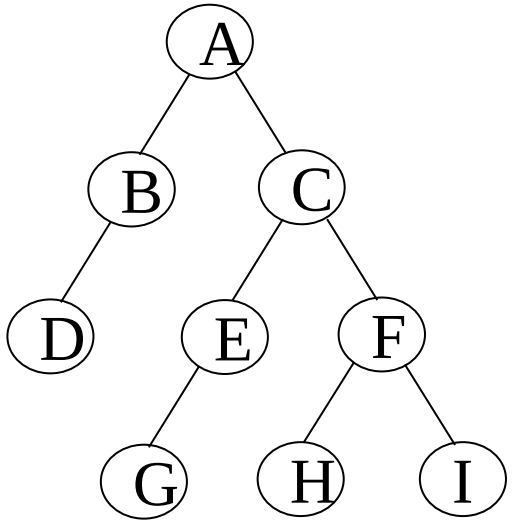



```
Status InOrderThreading(BiThrTree &Thrt, BiThrTree T)
{ // 中序遍历二叉树T，并将其中序线索化，Thrt指向头结点
  if (!(Thrt = (BiThrTree) malloc(sizeof(BiThrNode))))
     exit(OVERFLOW);
  Thrt->LTag = Link;  Thrt->RTag =Thread;  // 建头结点
  Thrt->rchild = Thrt;              // 右指针回指
  if (!T) Thrt->lchild = Thrt;      // 若二叉树空，则左指针回指
  else {
    Thrt->lchild = T;    pre = Thrt;
    InThreading(T);  // 算法6.7：中序遍历进行中序线索化
    pre->rchild = Thrt;
    pre->RTag = Thread; // 最后一个结点线索化
    Thrt->rchild = pre;
  }
  return OK;
} // InOrderThreading
```

线索二叉树

三. 线索二叉树的实现

- 线索二叉树按中序遍历构建
- 程序部分II




```
void InThreading(BiThrTree p) {  // 算法6.7
  if (p) {
    InThreading(p->lchild);    // 左子树线索化
    if (!p->lchild)   // 建前驱线索
      { p->LTag = Thread;  p->lchild = pre; }
    if (!pre->rchild) // 建后继线索
      { pre->RTag = Thread;  pre->rchild = p; }
    pre = p;                  // 保持pre指向p的前驱
    InThreading(p->rchild);    // 右子树线索化
  }
} // InThreading
```