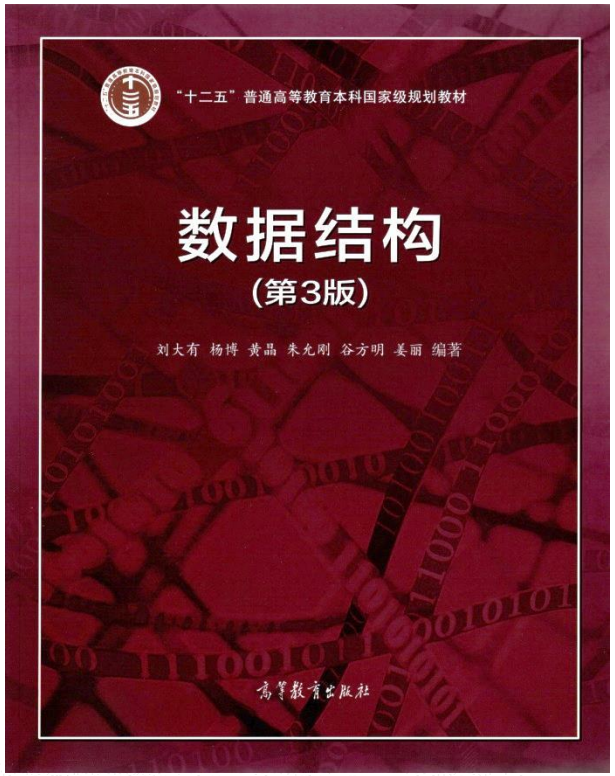


线索二叉树

动机和基本概念

基本操作

插入、删除和线索化

中序扩展二叉树

数据之法

结构之美

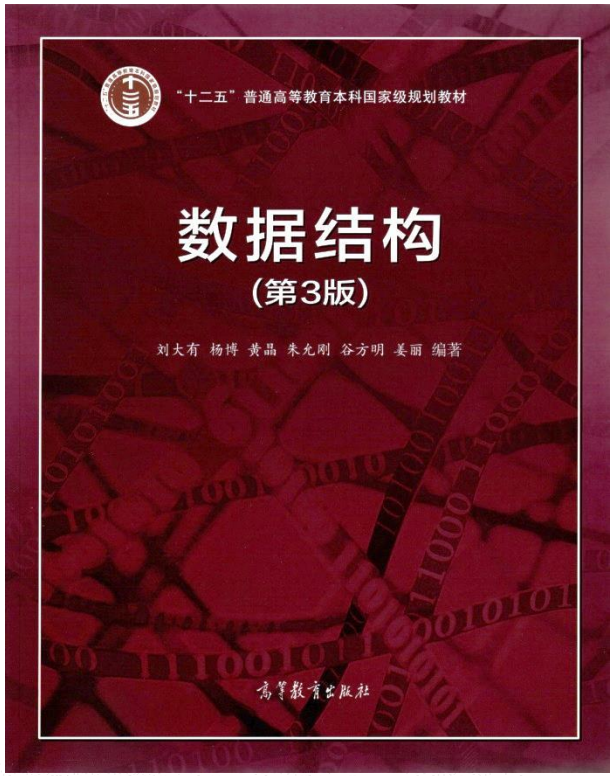
算法之道

zhuyungang@jlu.edu.cn

Last updated on 2022.10

七年前

七年后



线索二叉树

动机和基本概念

基本操作

中序线索化

中序扩展二叉树

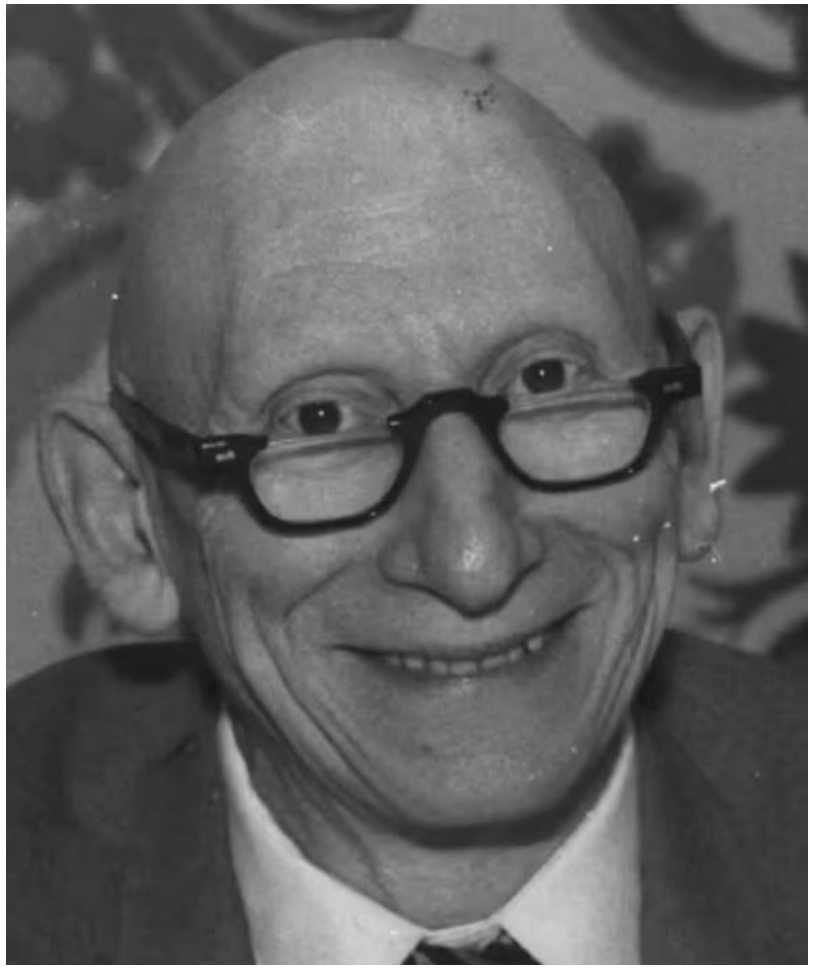
数据之法

结构之美

算法之道

zhuyungang@jlu.edu.cn

Last updated on 2022.10



Alan J. Perlis

(1922 - 1990)

首届图灵奖获得者

美国工程院院士

耶鲁大学教授

ACM美国计算机学会理事长

李凯

普林斯顿大学教授

美国工程院院士

中国工程院外籍院士

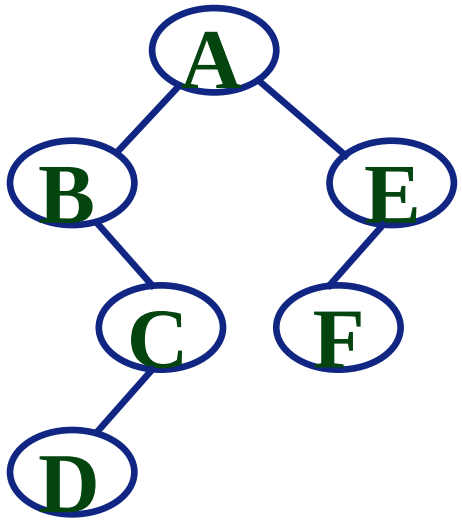
1954年生于吉林省长春市

本科毕业于吉林大学计算机系

耶鲁大学博士（师从Alan Perlis）

吉林大学计算机学科发展咨询委员会委员

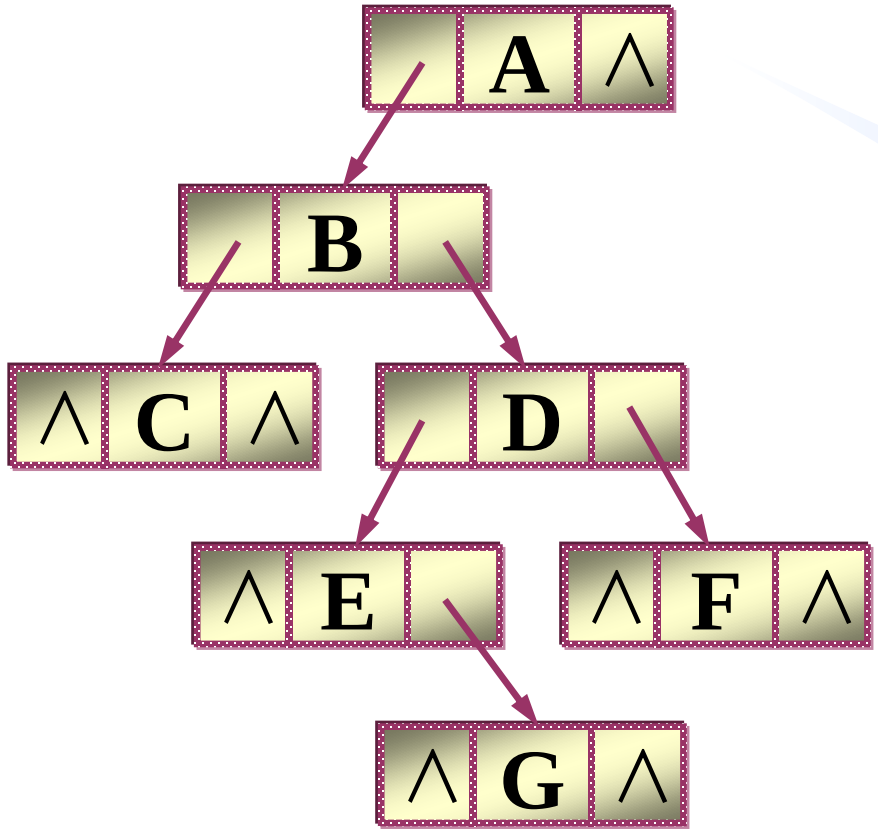
- 在二叉树上只能找到结点的左孩子、右孩子，找结点的中（先、后）根前驱和后继只能通过遍历。
- 能否更快速的找到给定结点的中（先、后）根前驱和后继？
- 二叉树的结点中有很多空指针，造成存储空间的浪费。



中根遍历序列：B D C A F E

中根序列中结点的前驱称作中根前驱，结点的后继称作中根后继。

包含n个结点的二叉树，在其$2n$个指针域中仅有$n-1$个被使用。



- 可以把这些空指针利用起来，让其指向结点的中根前驱或后继。

中根线索二叉树为例

Left	Data	Right

如果某结点

- 有子结点，则其Left/Right指向子结点
- 无子结点，则其Left/Right指向其中根前驱/后继

Thread=0

Thread=1

- 指向某结点中根前驱和后继的指针称为**线索**；
- 按中根遍历得到的线索二叉树称为**中序线索二叉树**；
  按先根遍历得到的线索二叉树称为**先序线索二叉树**；
  按后根遍历得到的线索二叉树称为**后序线索二叉树**。

线索二叉树的结点结构

LThread	Left	Data	Right	RThread

- 增加域 *LThread* 和 *RThread*，为二进制位，表示该结点的Left和Right是否为线索。
- 若结点*t*有左孩子，则*Left*指向*t*的左孩子，且*LThread*值为0；若 *t* 无左孩子，则*Left* 指向 *t* 的某一遍历序的前驱结点，且 *LThread* 值为1，此时称 *Left* 为线索。
- 若结点*t*有右孩子，则*Right*指向*t*的右孩子，且*RThread*值为0；若 *t* 无右孩子，则 *Right*指向*t*的后继结点，且 *RThread* 值为1，此时称 *Right* 为线索。

中根线索二叉树为例

LThread=

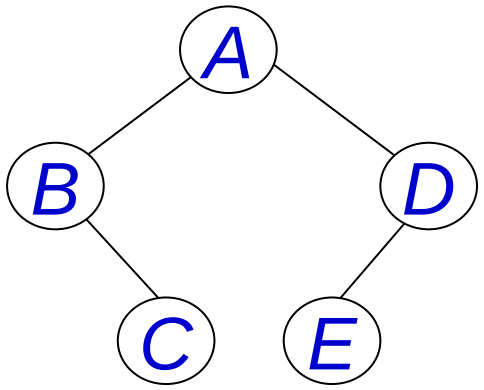
- 0 , left 域指示该结点的左孩子
- 1 , left 域指示该结点的中根前驱

RThread=

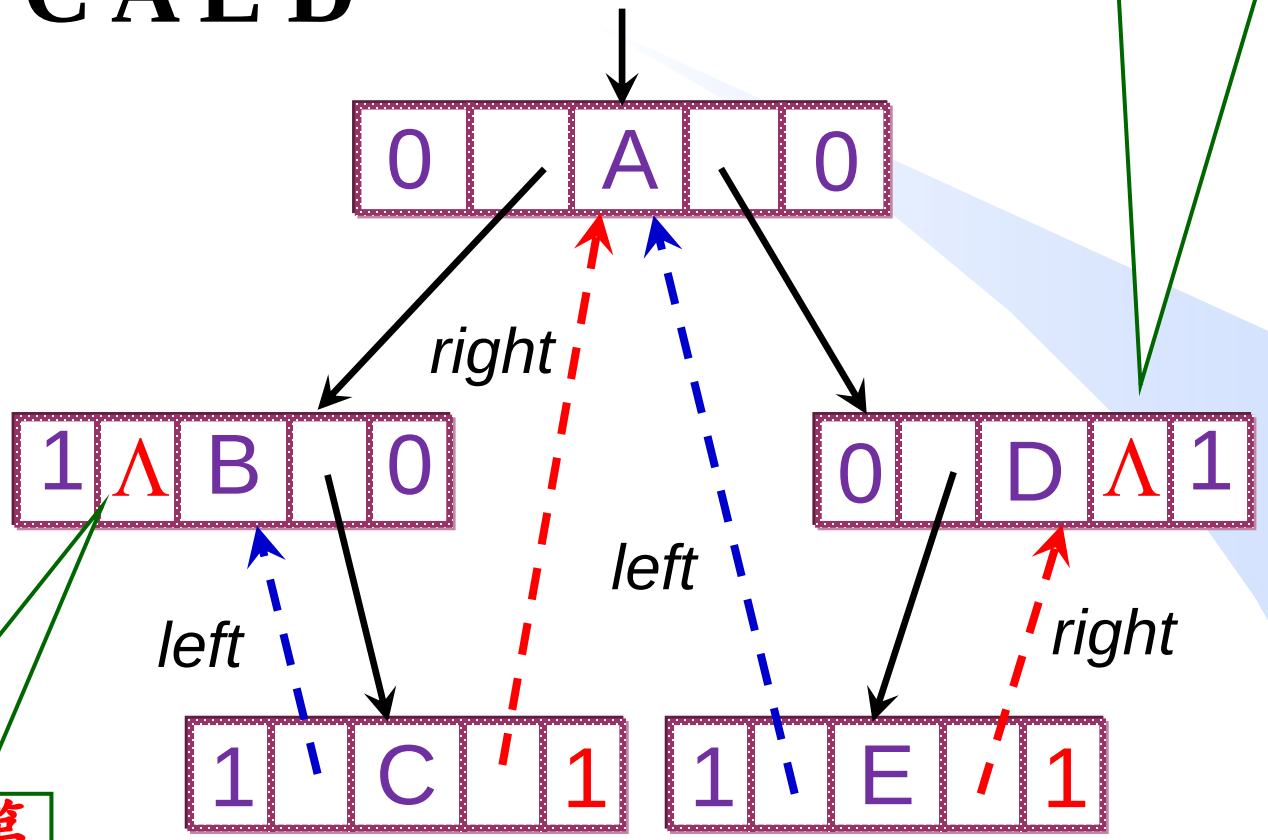
- 0 , right 域指示该结点的右孩子
- 1 , right 域指示该结点的中根后继

[例] 中序线索二叉树。

中根遍历序列： **B C A E D**



(a)二叉树



(b) 中序线索二叉树

中根序列最后一个结点一定无右孩子

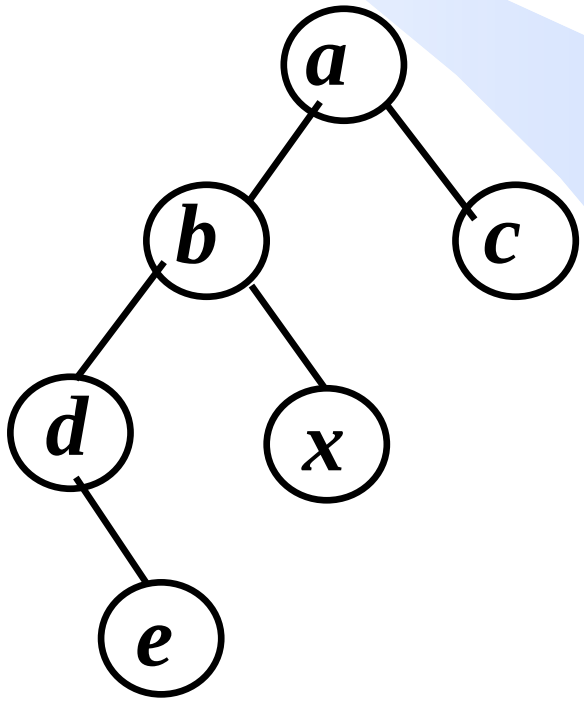
中根序列第一个结点一定无左孩子

课下思考

若对图中的二叉树进行中序线索化，则结点X的左右线索指向的结点分别是(　D　)【考研题全国卷】

A. *e, c*　　B. *e, a*

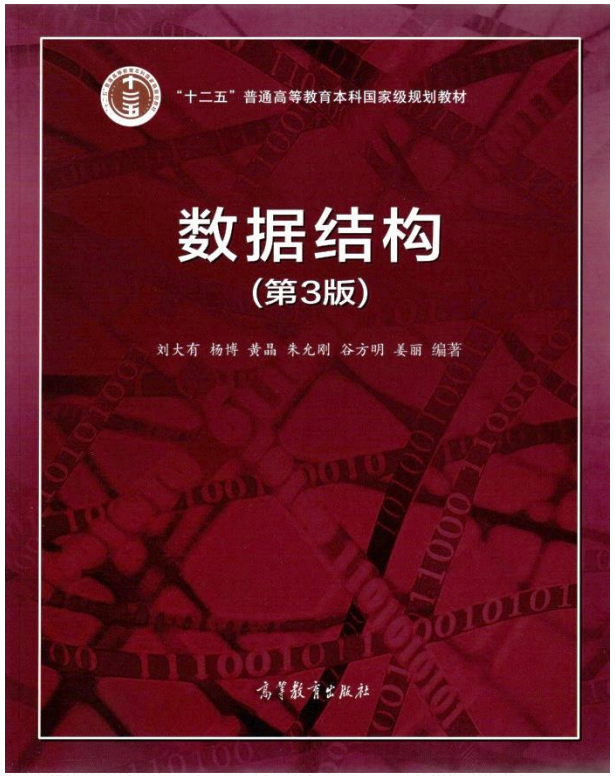
C. *d, c*　　D. *b, a*



线索二叉树的目的：

在**中序线索二叉树**中可以**方便的找到**给定结点的**中序前驱和中序后继**结点，并且不需要太多额外的空间。

线索二叉树中，一个结点是叶结点的充要条件为：左、右标志(LThread、RThread)均是1。



线索二叉树

动机和基本概念

基本操作

中序线索化

中序扩展二叉树

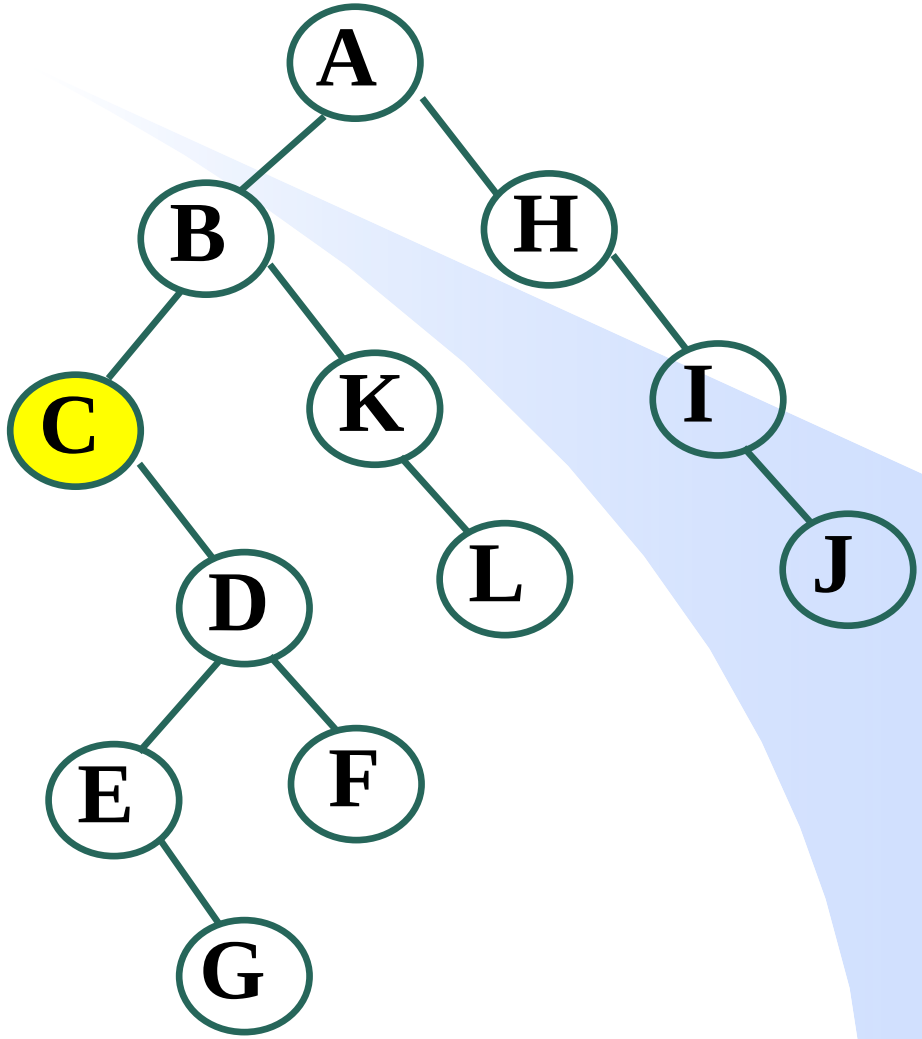
数据之法
结构之美
算法之道

zhuyungang@jlu.edu.cn

Last updated on 2022.10

找线索二叉树的中根序列的第一个结点

从根结点出发，沿左分支下行，直到最深的结点，该结点是中根序列第一个结点。

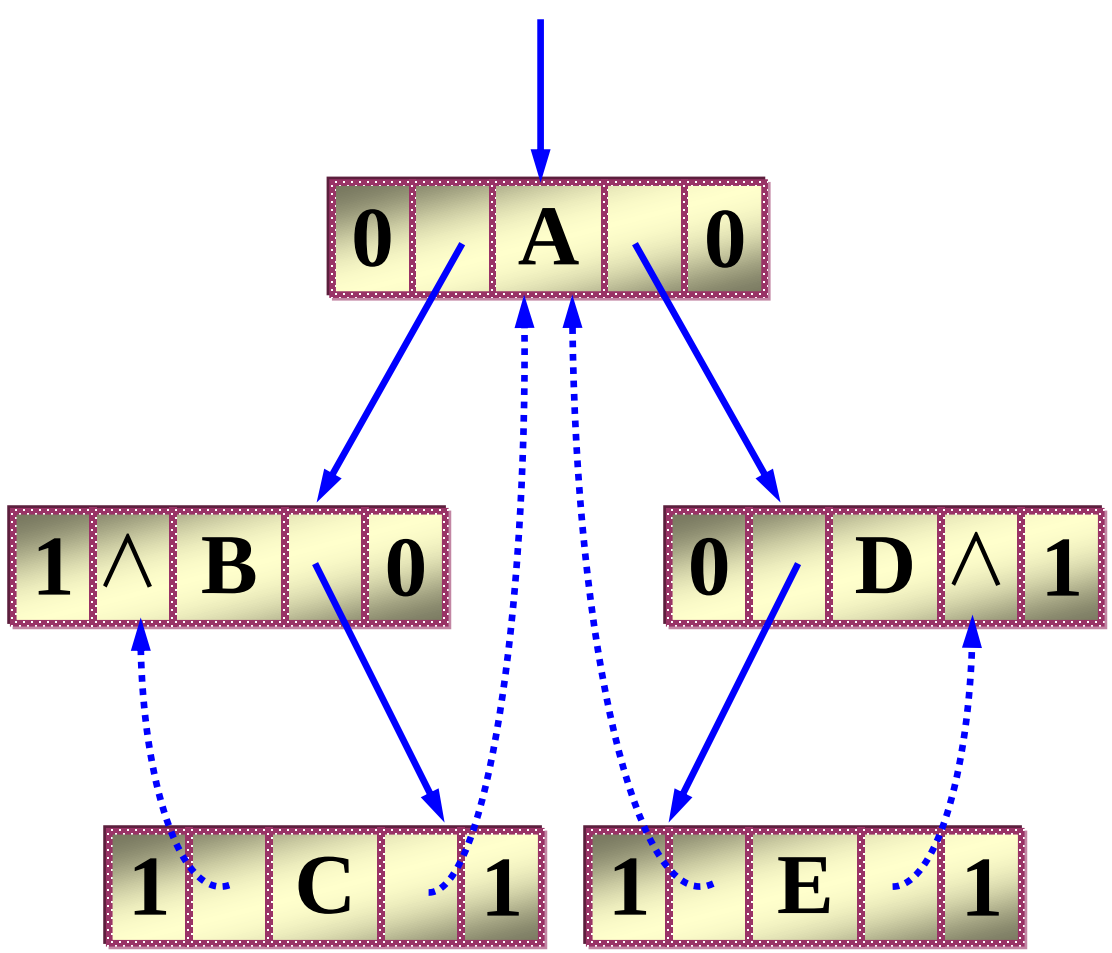


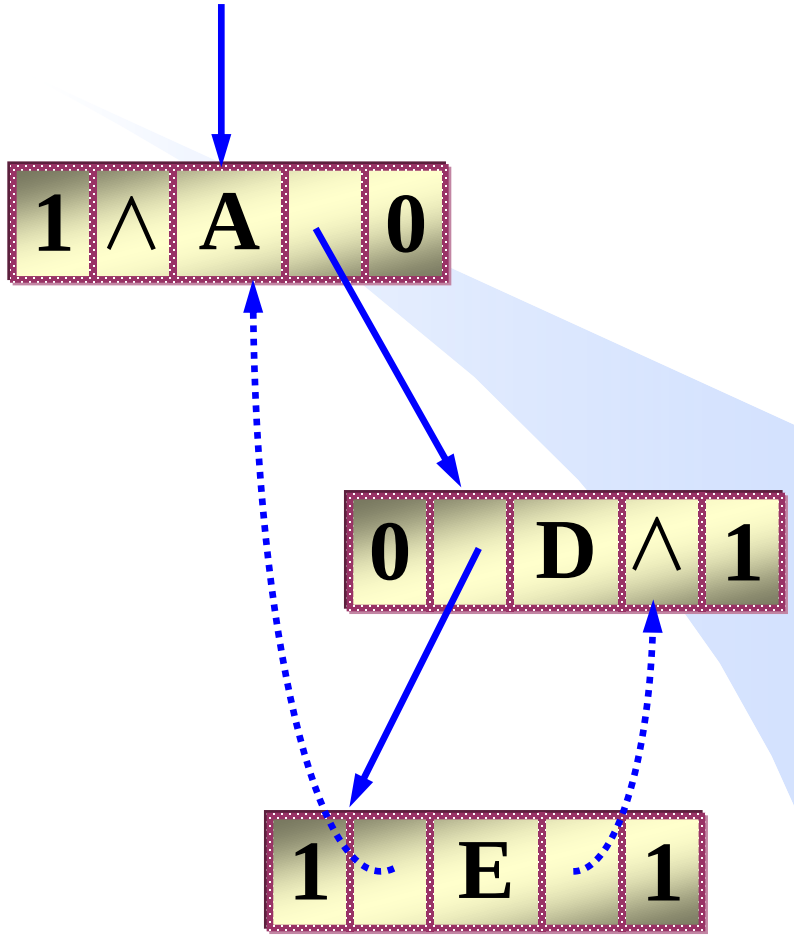
搜索以t为根的线索二叉树的中根序列的第一个结点

```
Node* FirstInOrder(Node* t){/*在以t为根的中序线索二叉树中找中根序列的首结点，并返回指向它的指针*/
    Node* p = t;
    while(p->Lthread == 0)
        p = p->left;
    return p;
}
```

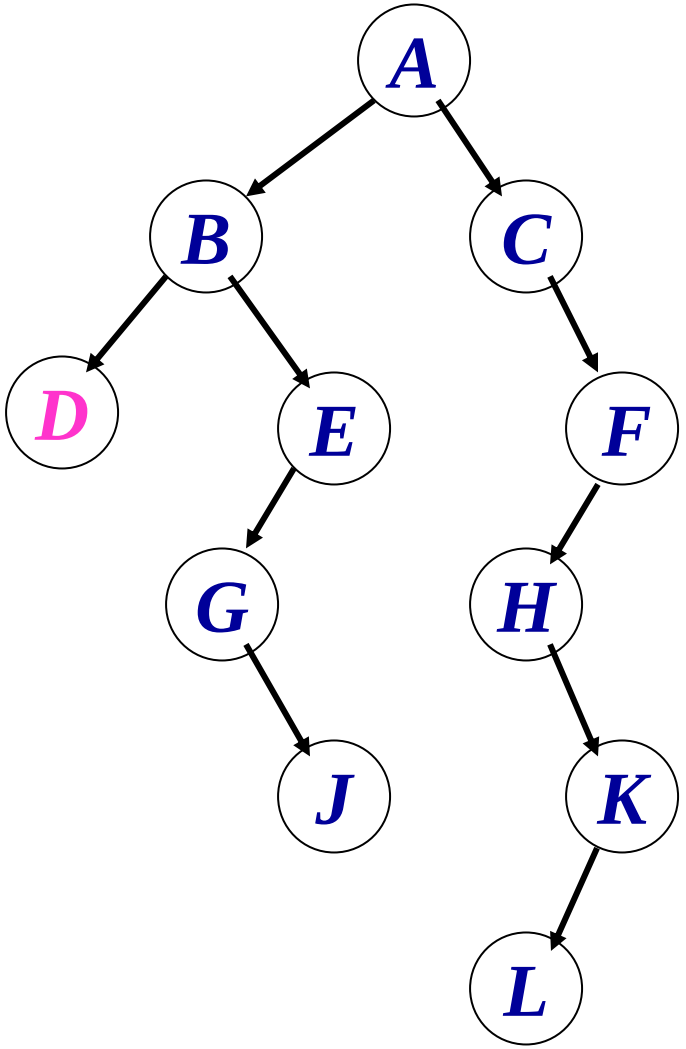
时间复杂度O(*h*)
***h*为二叉树高度**

```
while(p->Lthread == 0) p = p->left;
```





找线索二叉树的中根序列的最后一个结点



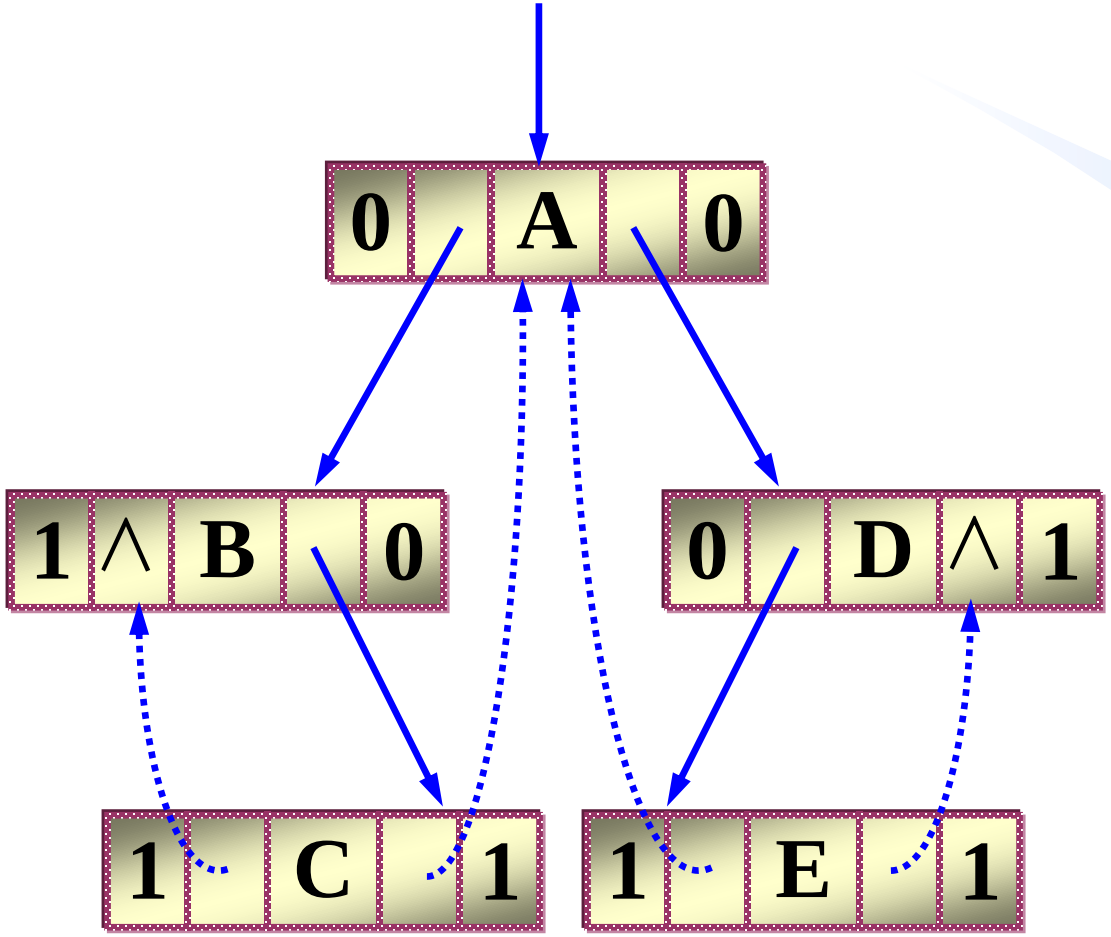
算法思想：

❖ 从根结点开始沿右分支下行，找第一个无右孩子的结点。

```
Node* LastInOrder(Node* t){
/*在以t为根的中序线索二叉树中找中根序列的末结点，并返回指向它的指针*/
    Node* p = t;
    while(p->Rthread == 0)
        p = p->right;
    return p;
}
```

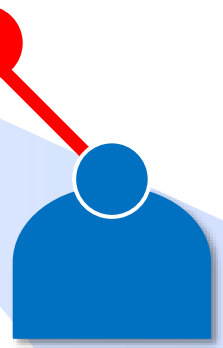
时间复杂度O(*h*)
***h*为二叉树高度**

```
while(p->Rthread == 0) p = p->right;
```



在中序线索二叉树中，找结点p的中根后继结点。

- 若$RThread(p)=1$，则$Right(p)$指向p的中根后继；
- 若$RThread(p)=0$，则p的中根后继为p的右子树的中根序列的首结点。

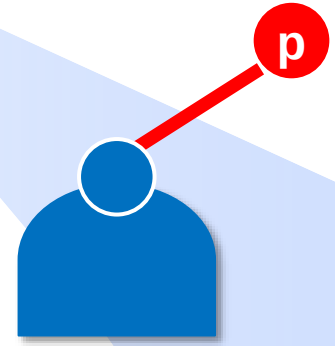


```
Node* NextInOrder(Node* t){
    if(p->Rthread==1) return p->right;
    return FirstInOrder(p->right);
}
```

时间复杂度$O(h)$
h为二叉树高度

在中序线索二叉树中，查找结点p的中根前驱结点

- 若$LThread(p)=1$，则$Left(p)$为左线索，直接指向p的中根前驱结点；
- 若$LThread(p)=0$，p的中根前驱结点是p之左子树中根序列的末结点。




```
Node* PreInOrder(Node* t){
    if(p->Lthread==1) return p->left;
    return LastInOrder(p->left);
}
```

时间复杂度O(h)
h为二叉树高度

回顾中序线索二叉树的操作

- 找中根序列的首结点　　FirstInOrder
- 找中根序列的末结点　　LastInOrder
- 找中序前驱结点　　PreInOrder
- 找中序后继结点　　NextInOrder

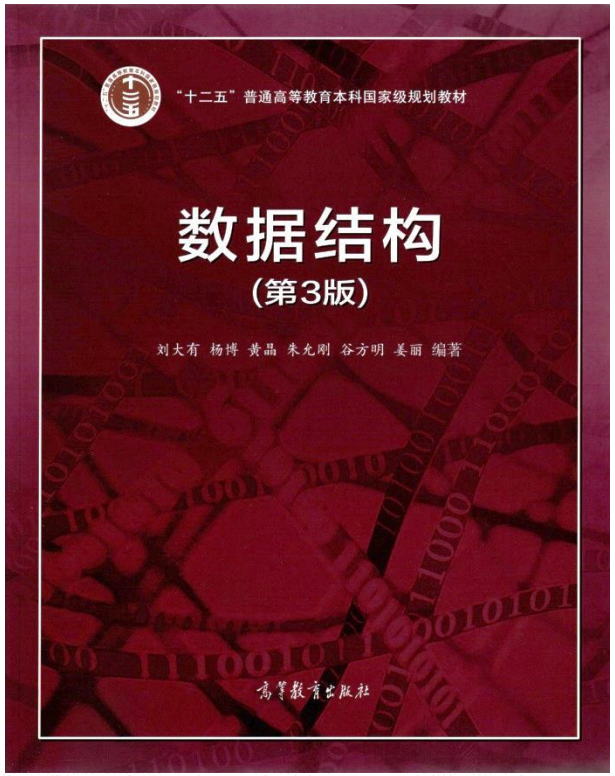
中根遍历线索二叉树

先访问中根序列中的第一个结点，然后依次访问结点的中根后继，直至其后继为空时为止。

```
void InOrder(Node* t){//中根遍历以t为根的中序线索二叉树
     Node* p = FirstInOrder(t); //找中根序列首结点
     while(p != NULL){
          printf("%d ", p->data);
          p = NextInOrder(p); //依次找p的中根后继
     }
}
```

把层次结构变成一个线性结构，遍历通过循环实现

```
void InOrder2(Node *t) { //版本2，用for循环实现
 for(Node *p=FirstInOrder(t);p!=NULL;p=NextInOrder(p))
     printf("%d ", p->data);
}
```



线索二叉树

动机和基本概念

基本操作

中序线索化

中序扩展二叉树

数据之法

结构之美

算法之道

zhuyungang@jlu.edu.cn

Last updated on 2022.10

二叉树的中序线索化

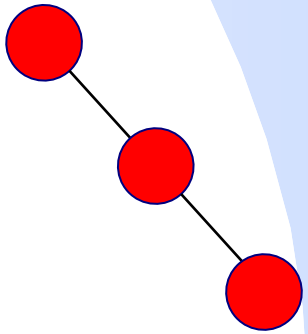
- 使二叉树变为线索二叉树的过程称为线索化。
- 将二叉树中根遍历算法中的“访问结点”操作具体化为“建立当前访问的结点与其中根前驱结点的线索关系”。
- 算法中指针*p*指示当前正在访问的结点，同时设置一个指针pre（作为全局变量）始终指向当前访问结点*p*的中根前驱结点，即中根遍历过程中在*p*之前访问的结点，pre的初值为NULL。

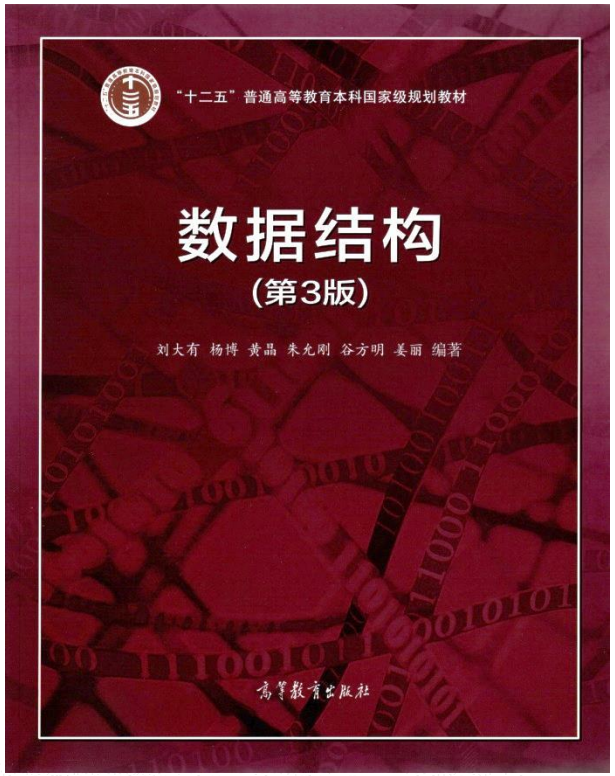
二叉树的中序线索化

```
void Inorder_threading(Node *p){//初始调用Inorder_threading(root)
    if (p==NULL) return;
    Inorder_threading(p->left); //中序线索化p的左子树
    //如果p没有左孩子,p的左指针是线索域,指向中根前驱pre
    if(p->left==NULL){ p->Lthread=1;  p->left=pre; }
    else p->Lthread=0;
    //如果pre没有右孩子, pre的右指针是线索域,指向其中根后继p
    if(pre!=NULL && pre->right==NULL){pre->Rthread=1;pre->right=p;}
    else if(pre!=NULL) pre->Rthread=0;
    pre=p;   //将当前访问的结点作为pre
    Inorder_threading(p->right); //中序线索化p的右子树
}
```

算法结束后要调整中序最后一个结点（pre指向）的右线索：

```
pre->Rthread=1;
```





线索二叉树

动机和基本概念

基本操作

中序线索化

中序扩展二叉树

数据之法

结构之美

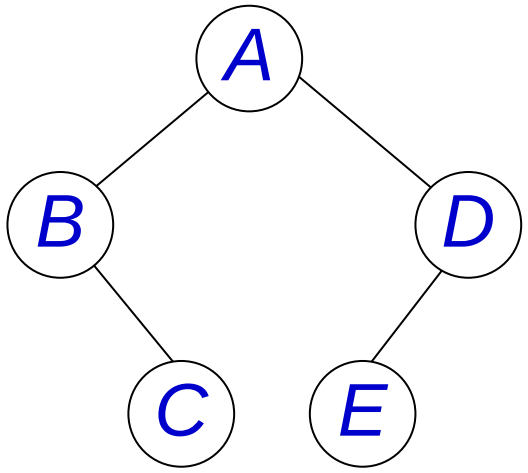
算法之道

zhuyungang@jlu.edu.cn

Last updated on 2022.10

扩展二叉树的结点结构

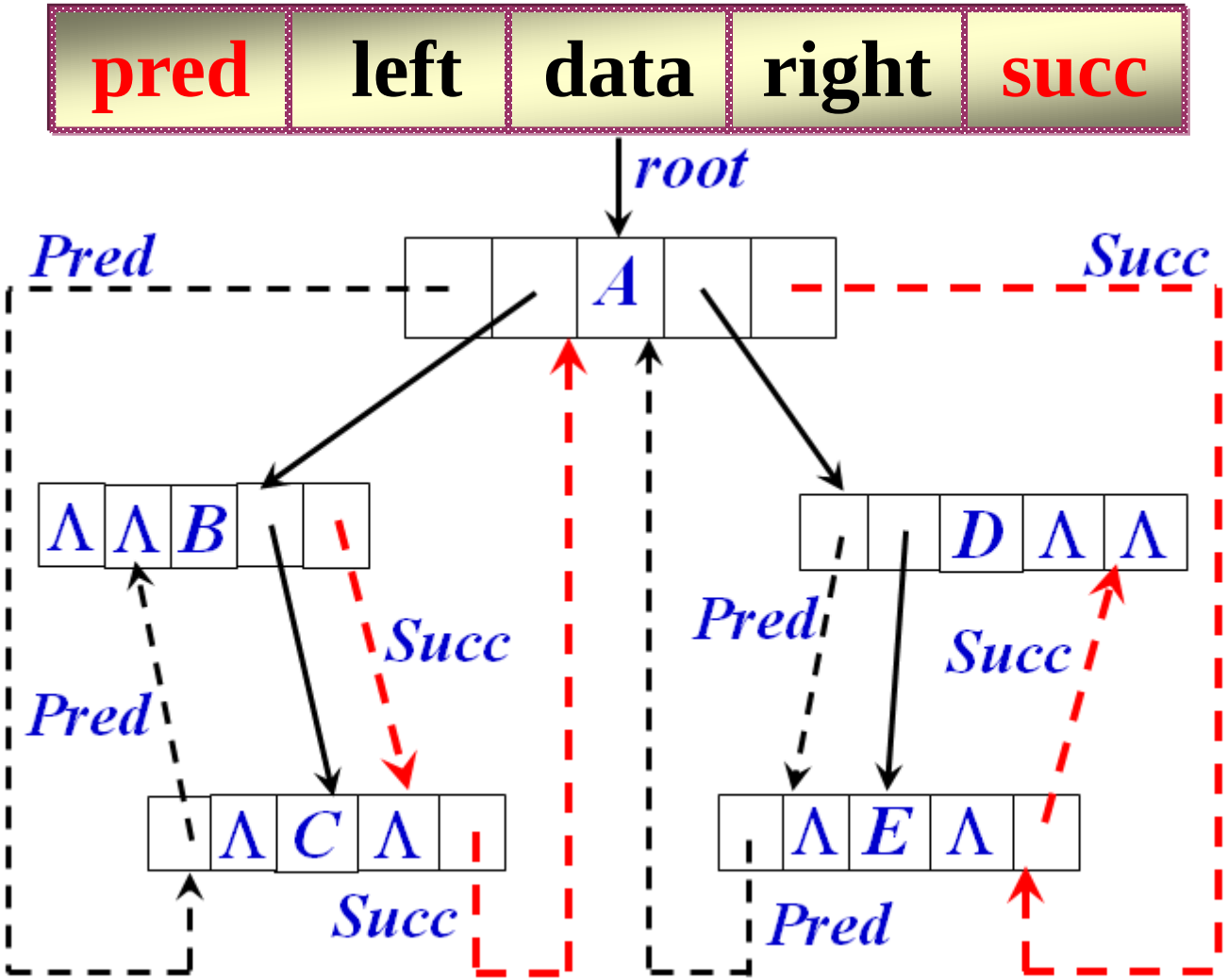
- 线索二叉树查找结点的前驱或后继最坏时间复杂度是O(*h*)，如果需要经常查找结点的前驱或后继，效率不高。
- 空间换时间，把*LThread* 和 *Rthread*换成指针域；



二叉树

中序扩展二叉树

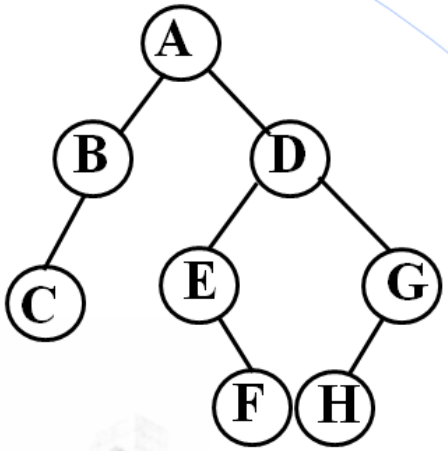
中序序列：BCAED

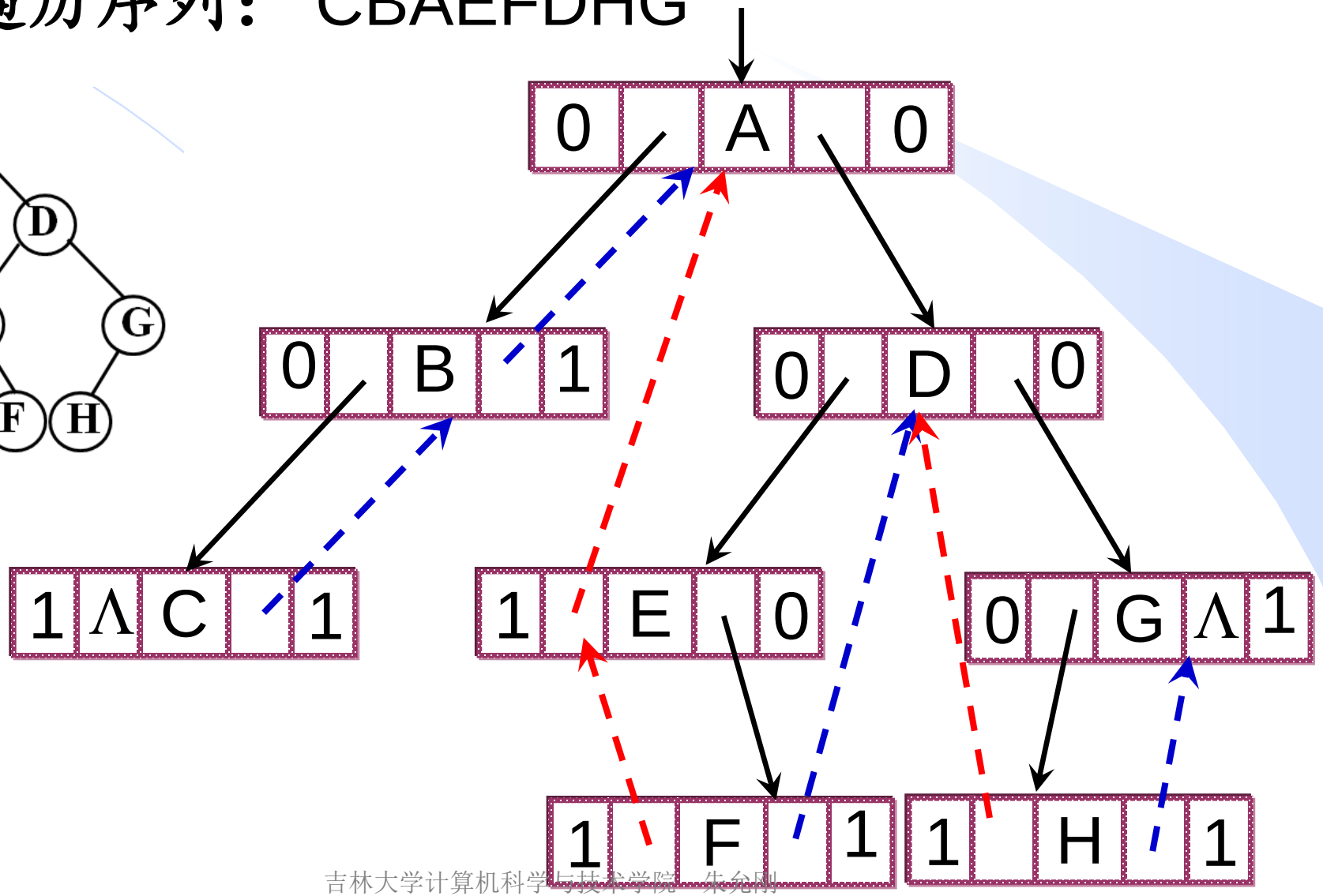


图中虚线箭头表示线索

课下练习：画出左图二叉树的中序线索二叉树

中根遍历序列： CBAEFDHG

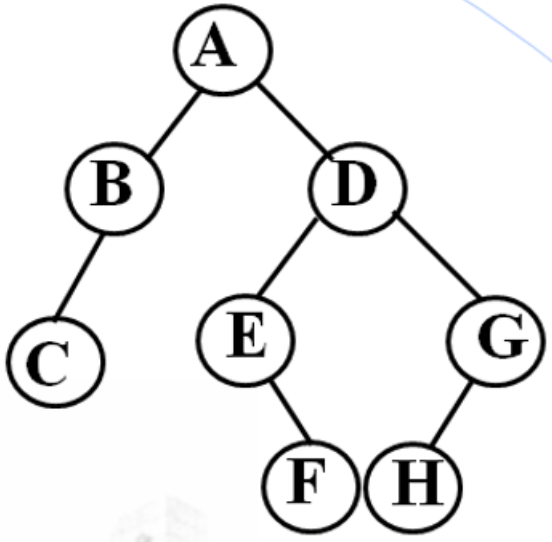




有一棵二叉树的中根遍历序列为CBAEFDHG，层次遍历序列为ABDCEGFH，请（1）给出此二叉树的后根遍历序列。（2）画出此二叉树的中序线索二叉树的链接结构图。【2018级期末考试题（8分）】

（1）二叉树后根序列，4 分

CBFEHGDA ↵



（2）中序线索二叉树，4 分