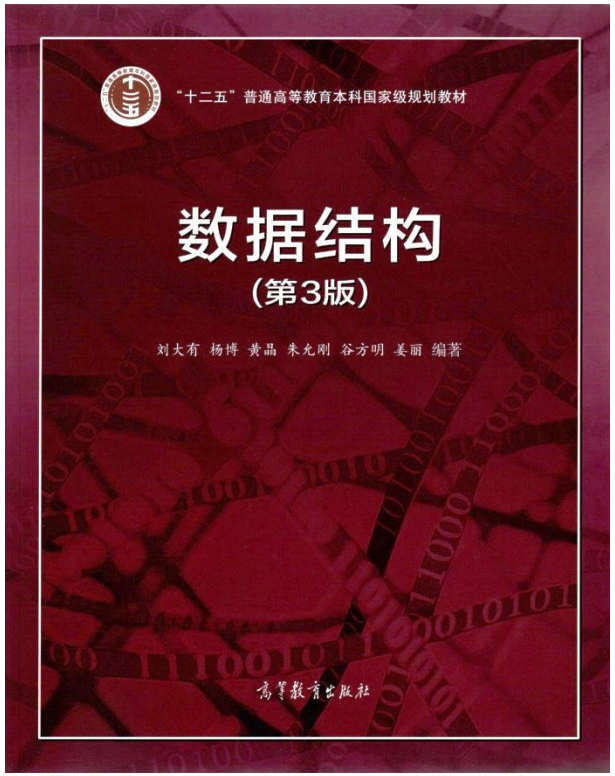


线索二叉树

动机和基本概念

中序线索二叉树的基本操作

问题与拓展

数据之法

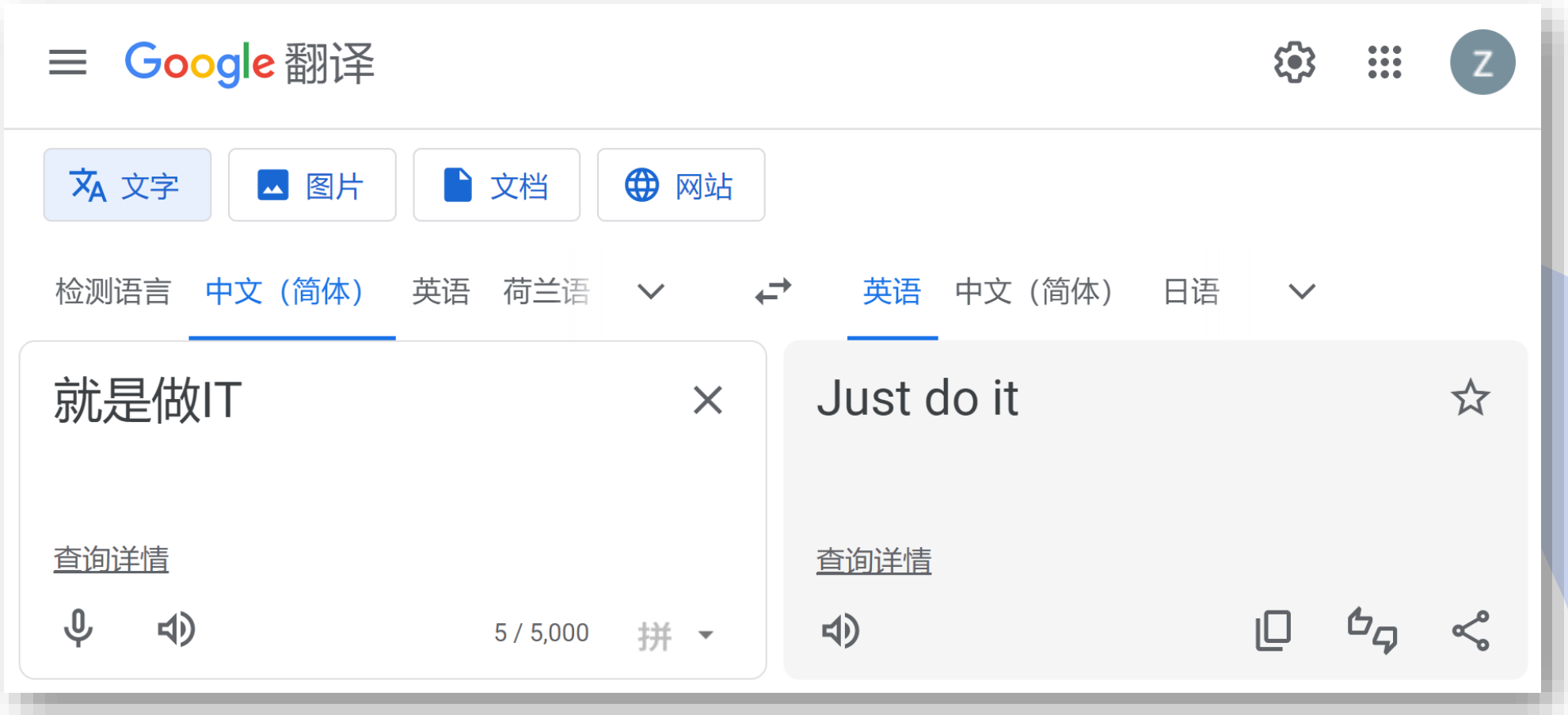
结构之美

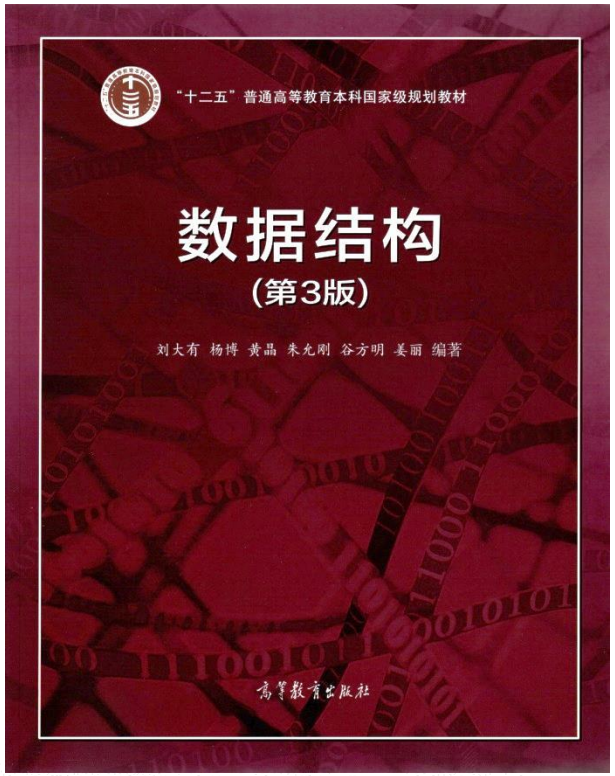
算法之道

zhuyungang@jlu.edu.cn

Last updated on 2024.10

谷歌翻译权威认证


Google 翻译
Z
文字
图片
文档
网站
检测语言
中文（简体）
英语
荷兰语
英语
中文（简体）
日语
就是做IT
查询详情
5 / 5,000
拼
Just do it
查询详情



线索二叉树

动机和基本概念

中序线索二叉树的基本操作

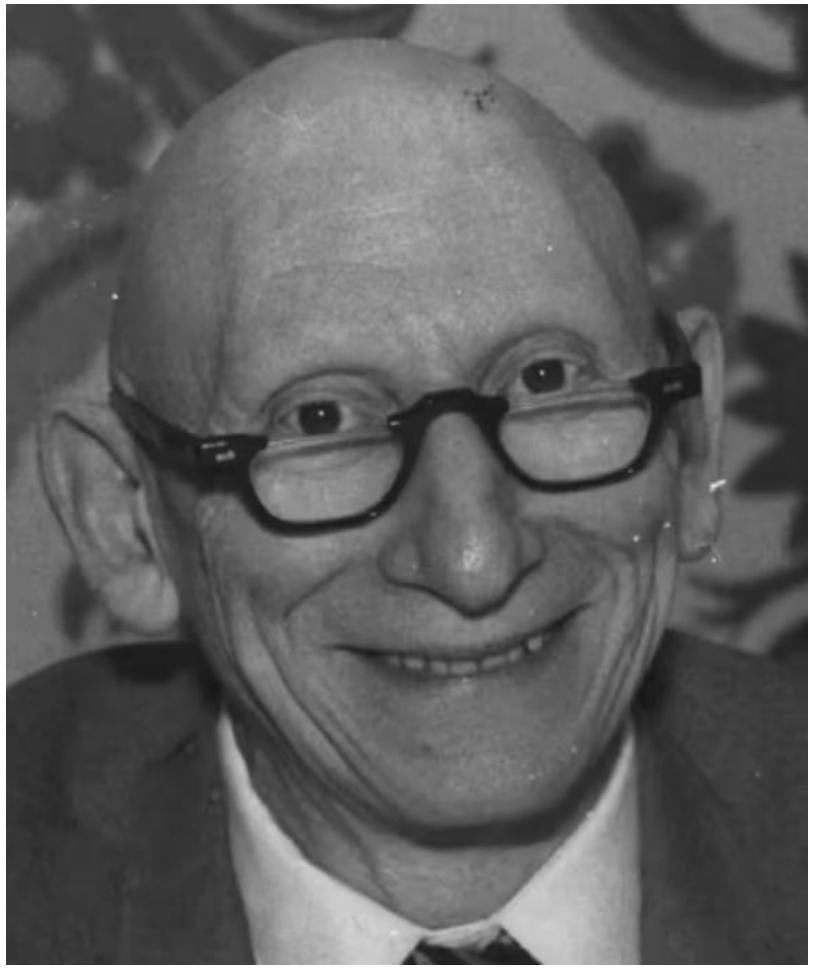
问题与拓展

数据之法
结构之美
算法之道

zhuyungang@jlu.edu.cn

Last updated on 2024.10

线索二叉树的提出者



Alan J. Perlis

(1922 - 1990)

首届图灵奖获得者

美国工程院院士

卡内基梅隆大学计算机系创始人

卡内基梅隆大学教授

耶鲁大学教授

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

1977年本科毕业于吉林大学计算机系

耶鲁大学博士（师从**Alan Perlis**）

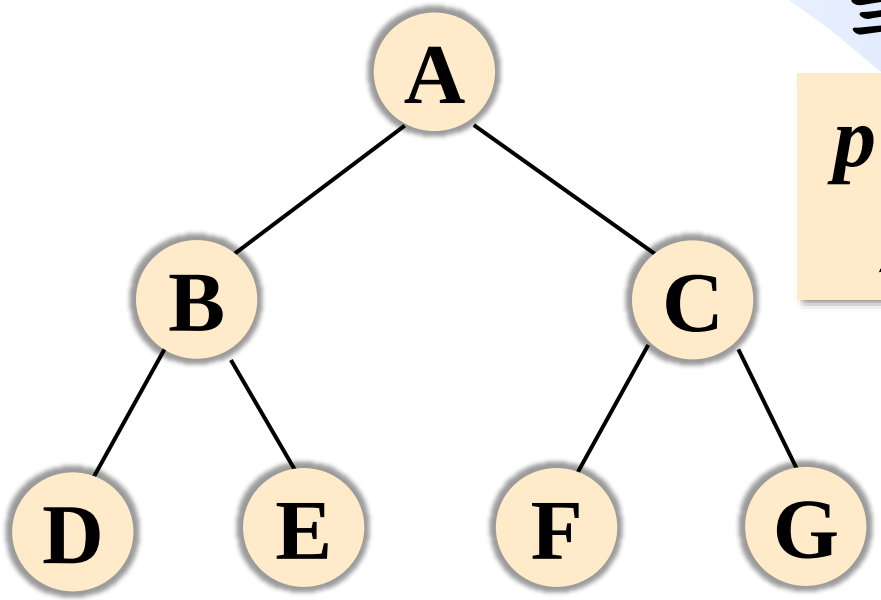
吉林大学计算机学科发展咨询委员会委员

练习

B

已知二叉树结点结构如下，给定二叉树和其中一个结点p，找出p的中根后继结点。【腾讯面试题】

```
struct TreeNode{
   int data;
   TreeNode *parent;
   TreeNode *left;
   TreeNode *right;
};
```



中根序列：D B E A F C G

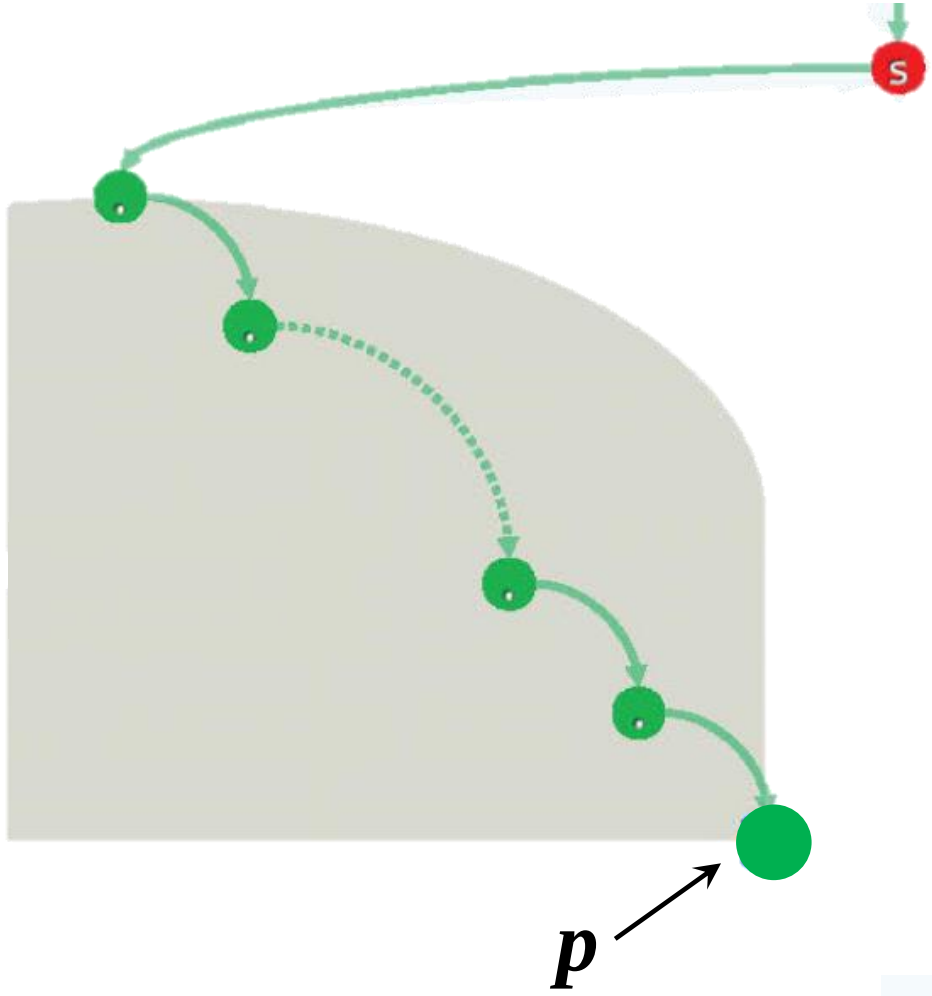
当p有右孩子时：

p的右子树的中根序列第1个结点

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

练习

B



当p无右孩子时：

将p包含于其左子树的最低祖先

时间复杂度

有父指针：$O(h)$

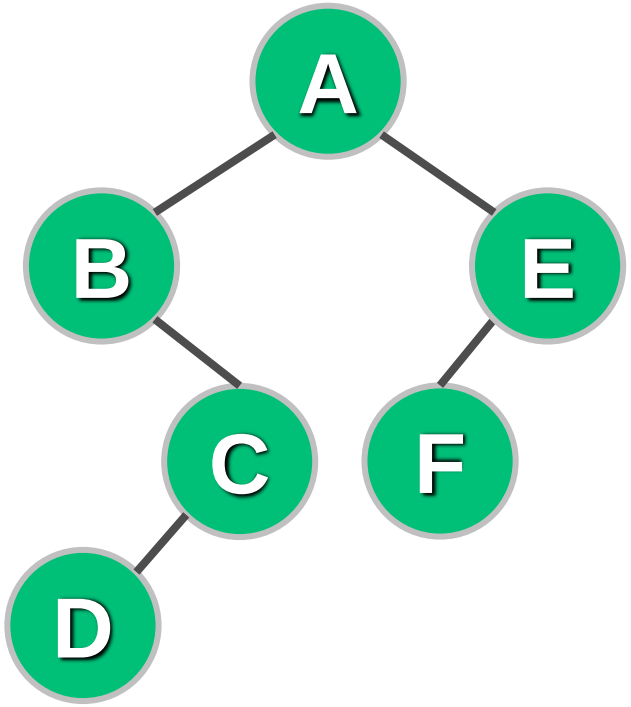
无父指针：中根遍历$O(n)$

对于完全二叉树$h=\lfloor \log n \rfloor$

线索二叉树——动机

A

- 在二叉树（结点无父指针域）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，并且不需要太多额外的空间？



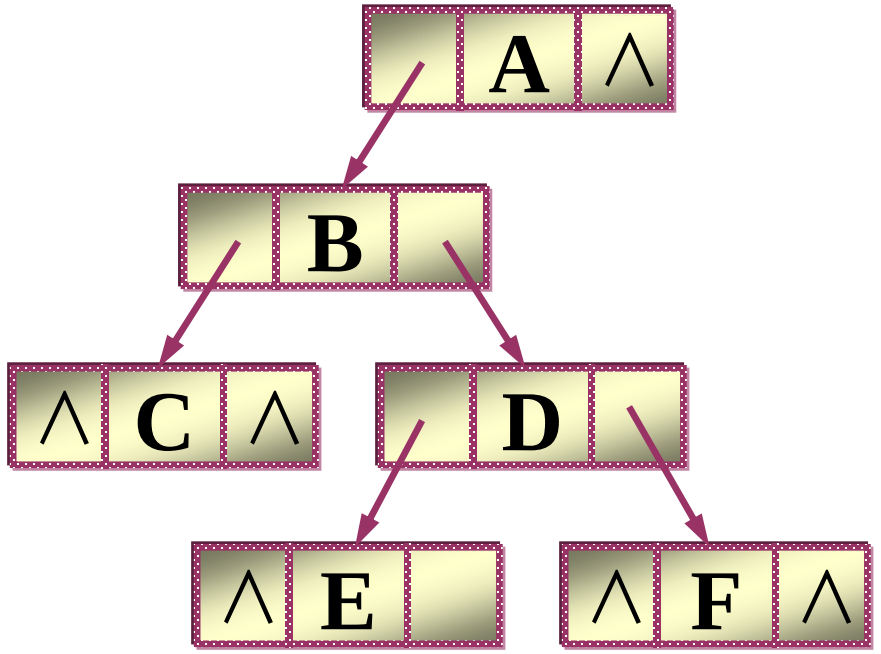
中根遍历序列：B D C A F E

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。

线索二叉树——动机

A

- 二叉树的结点中有很多空指针，造成存储空间的浪费。
- 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：指向结点的中根前驱或后继。



每个非空指针域
都对应一条边

线索二叉树

- 如果某结点
  - ✓有孩子，则其*Left/Right*指向孩子；
  - ✓无孩子，则其*Left/Right*指向其某种遍历序的前驱/后继，此时指针称为**线索**。
- 增加标志位*LThread* 和 *RThread*（为二进制位，占1比特）表示该结点的*Left*和*Right*指针到底是孩子指针（指向孩子）还是线索指针（指向某种遍历序的前驱/后继）。

线索二叉树

A

结点	LThread	Left	RThread	Right
有左孩子	0	指向左孩子		
无左孩子	1	指向前驱结点		
有右孩子			0	指向右孩子
无右孩子			1	指向后继结点

按中/先/后根遍历得到的线索二叉树称为中/先/后序线索二叉树

如何判断结点有无孩子?

如何判断叶结点?

以中序线索二叉树为例

A

LThread	Left	Data	Right	RThread

$$LThread = \begin{cases} 0, Left\ 指向该结点的左孩子 \\ 1, Left\ 指向该结点的中根前驱 \end{cases}$$

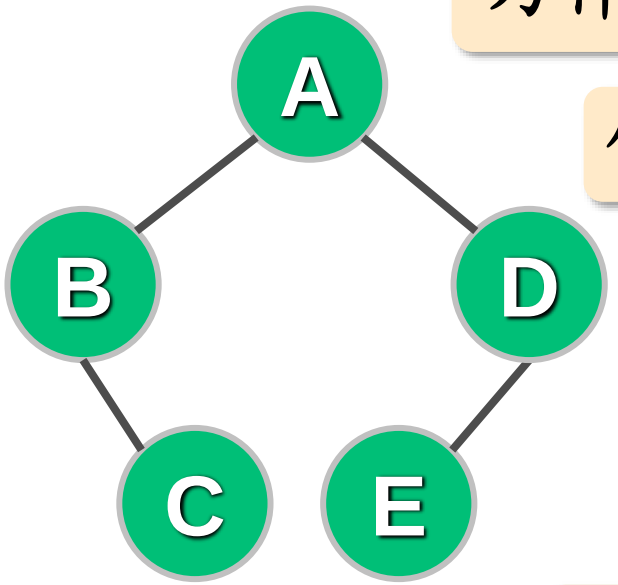
$$RThread = \begin{cases} 0, Right\ 指向该结点的右孩子 \\ 1, Right\ 指向该结点的中根后继 \end{cases}$$

一个结点是叶结点：Lthread==1 && Rthread==1

中序线索二叉树

A

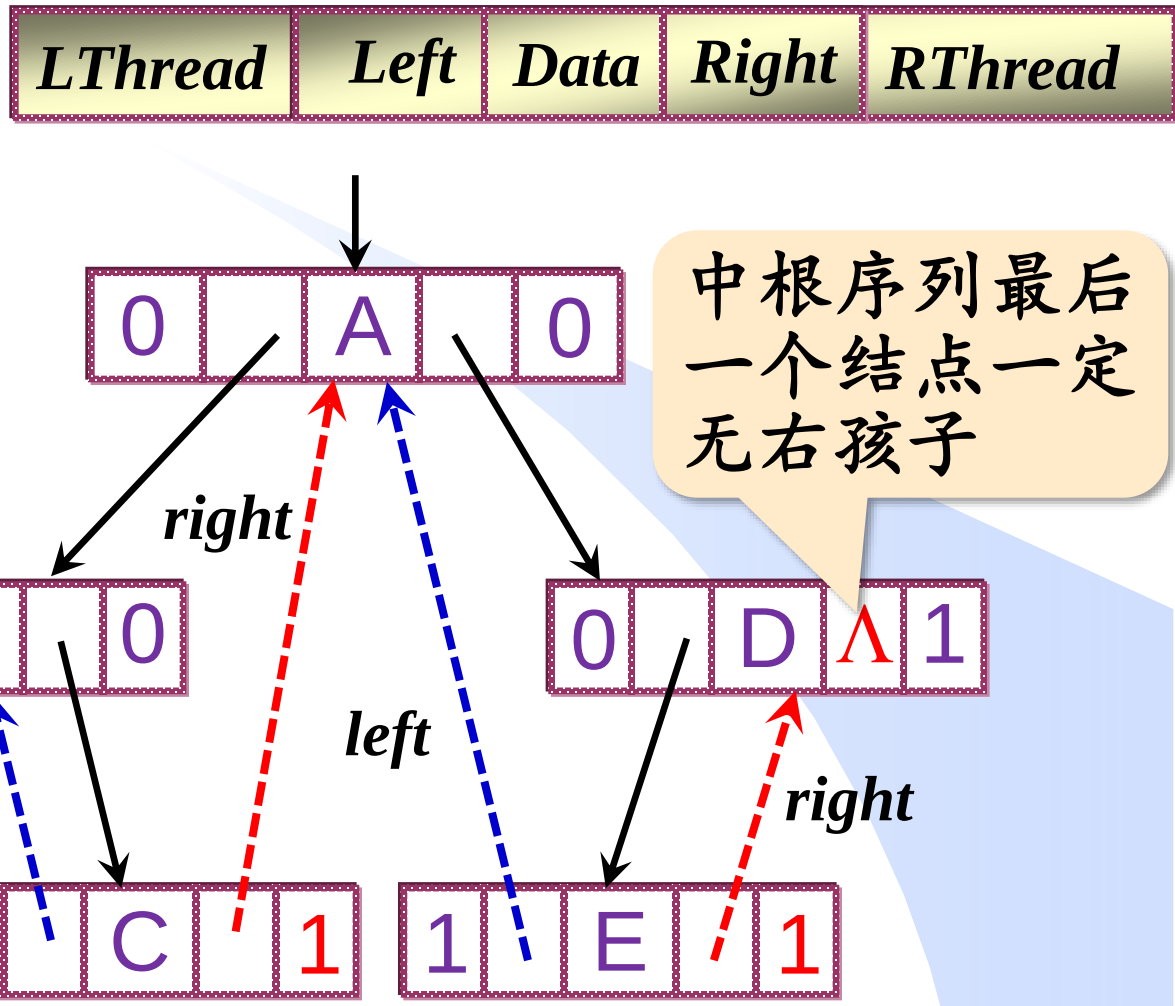
中根序列：B C A E D



为什么叫线索？

偶然还是必然？

中根序列第一个结点一定无左孩子



中根序列最后一个结点一定无右孩子

中序线索二叉树

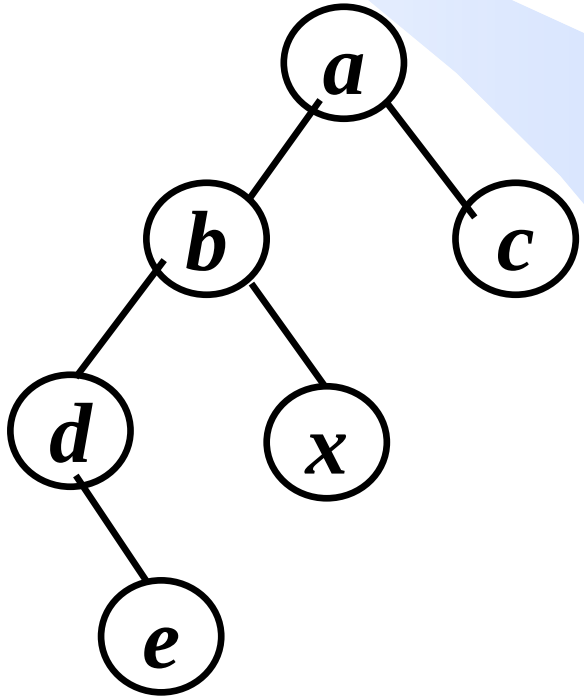
课下思考

A

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

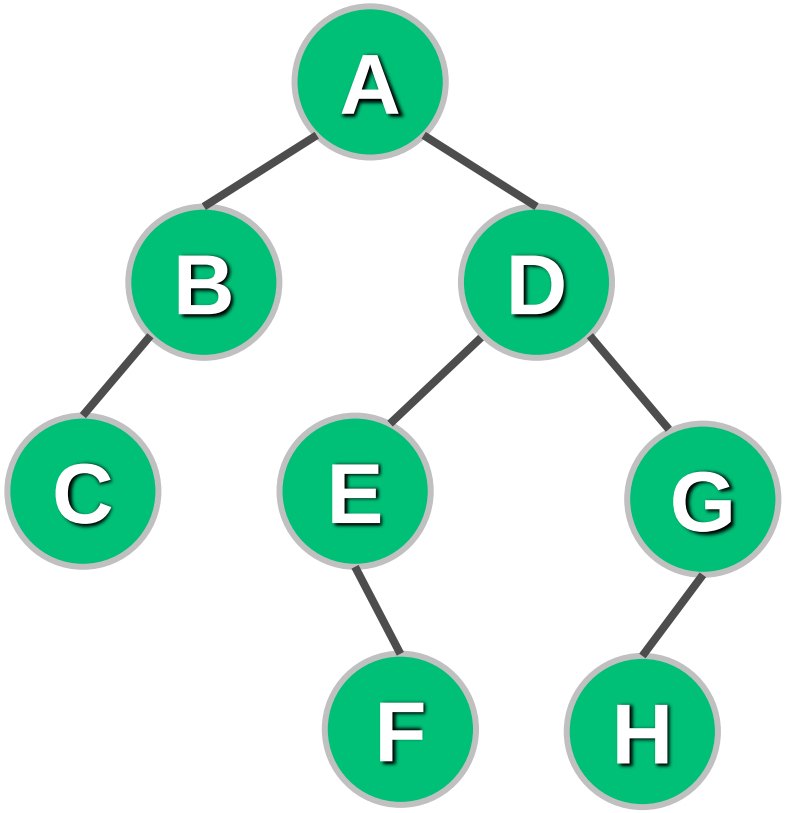
C. d, c　　D. b, a

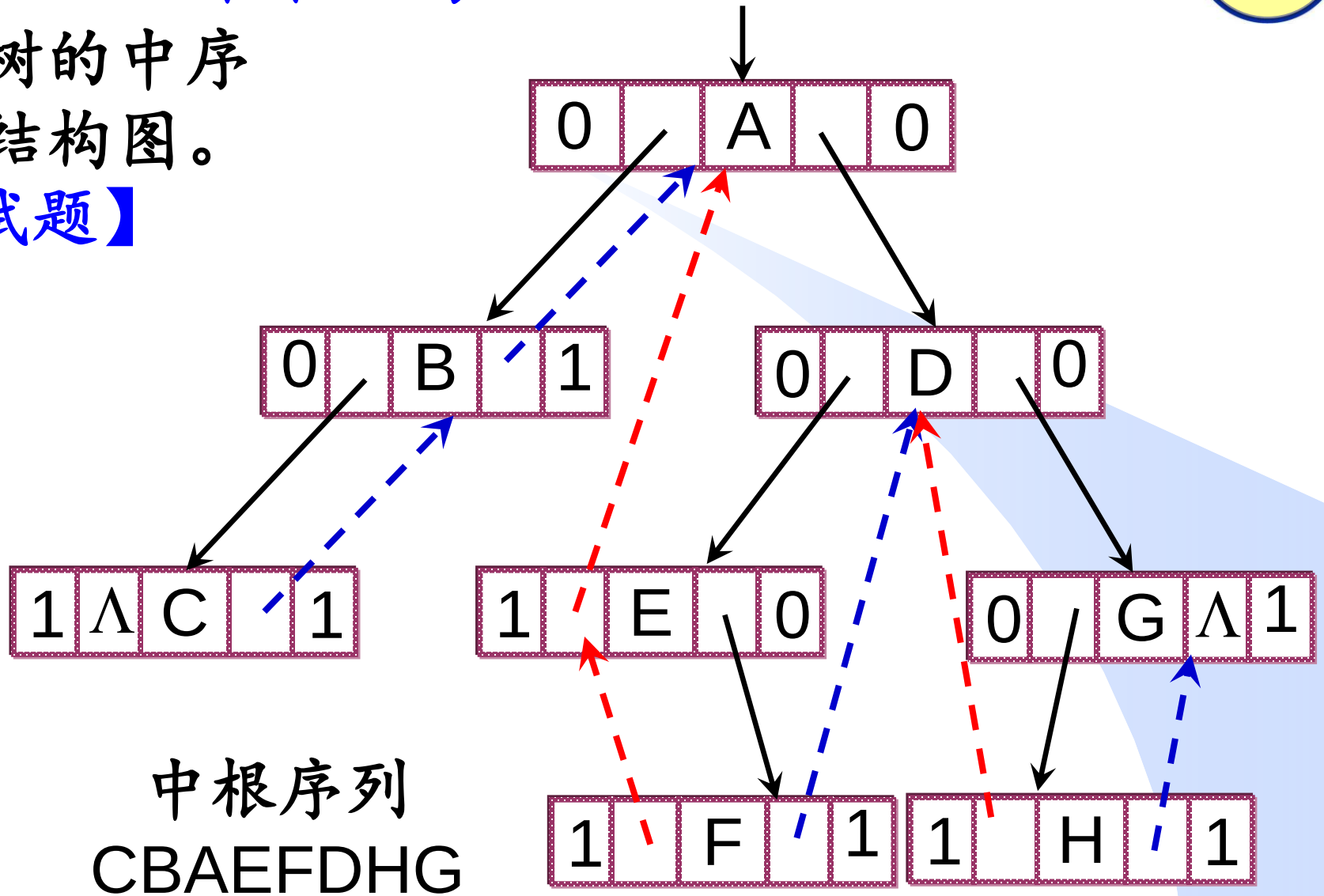


课下思考

画出左图所示二叉树的中序线索二叉树的链接结构图。

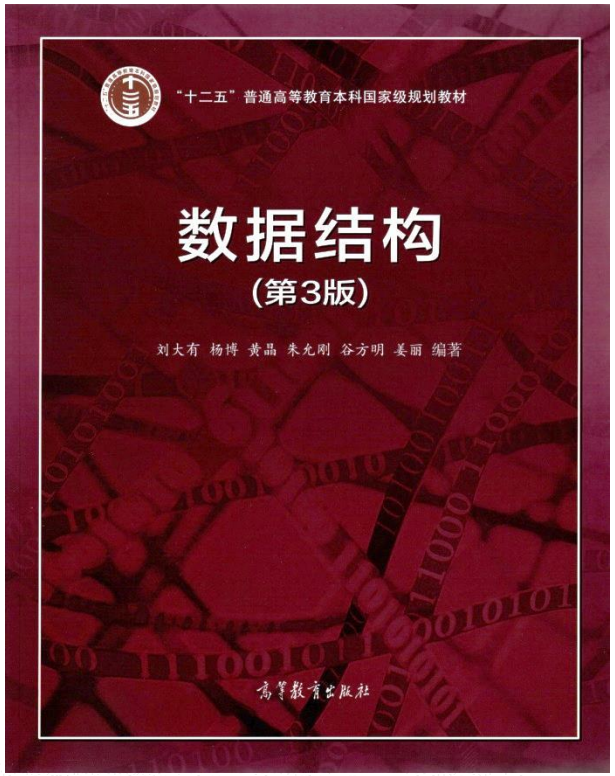
【吉林大学期末考试题】





中根序列

CBAEFDHG



线索二叉树

动机和基本概念

中序线索二叉树的基本操作

问题与拓展

数据之法
结构之美
算法之道

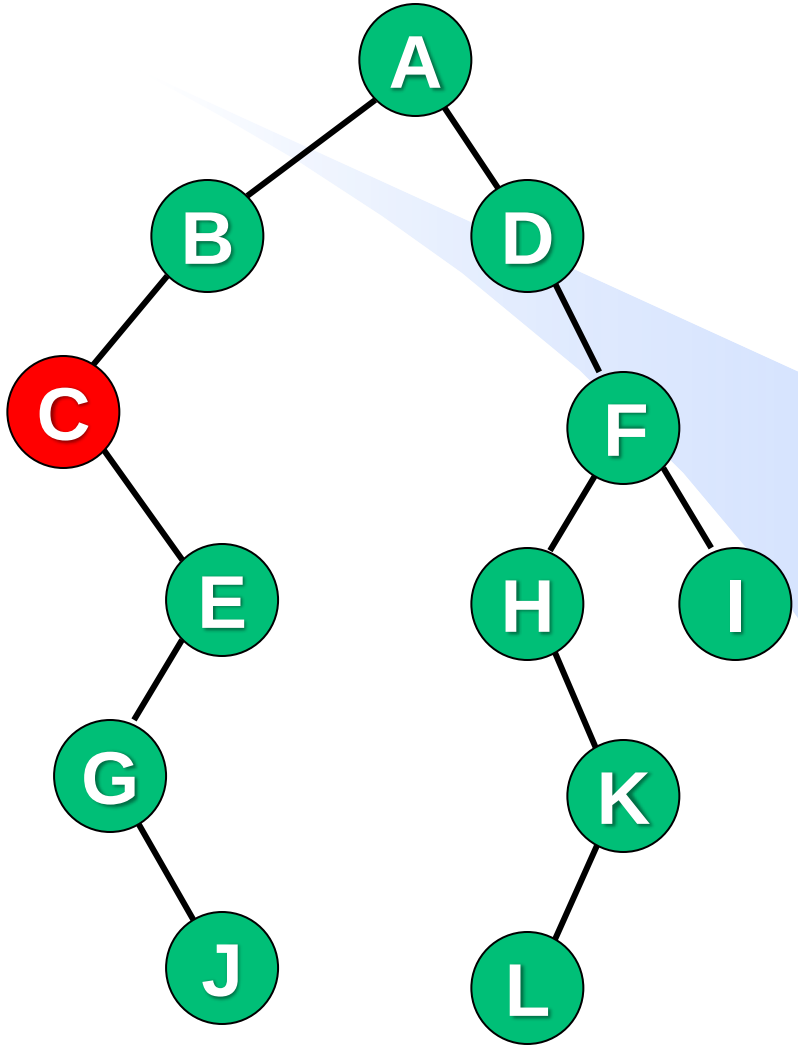
zhuyungang@jlu.edu.cn

Last updated on 2024.10

找线索二叉树的中根序列的第一个结点

B

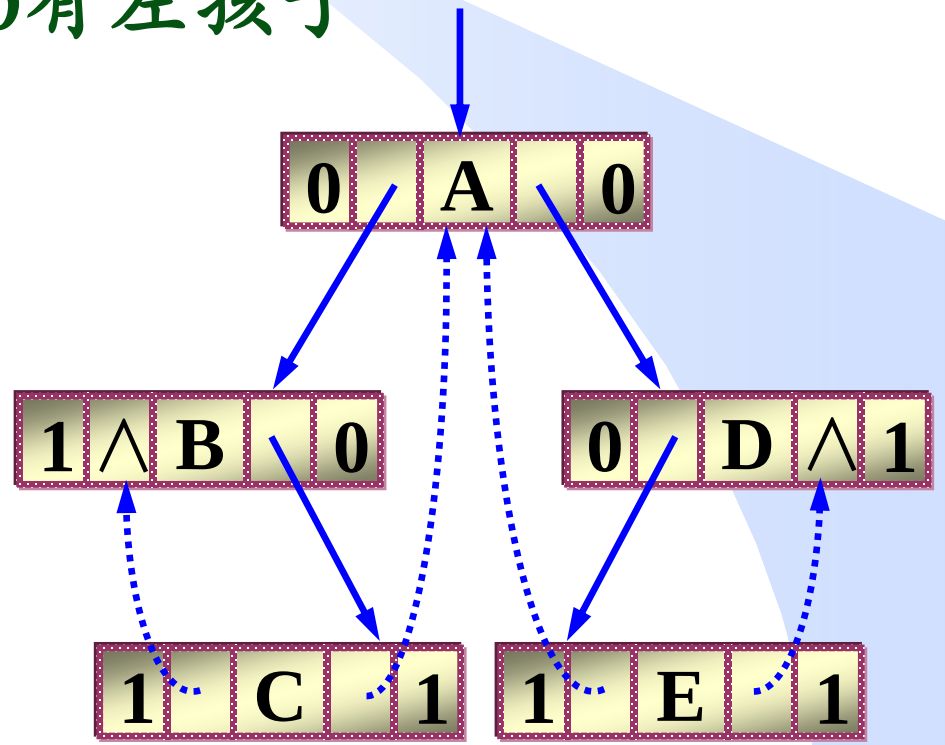
从根结点出发，沿左分支下行，直到最深的结点，该结点是中根序列第一个结点。



找中根序列的第一个结点

B

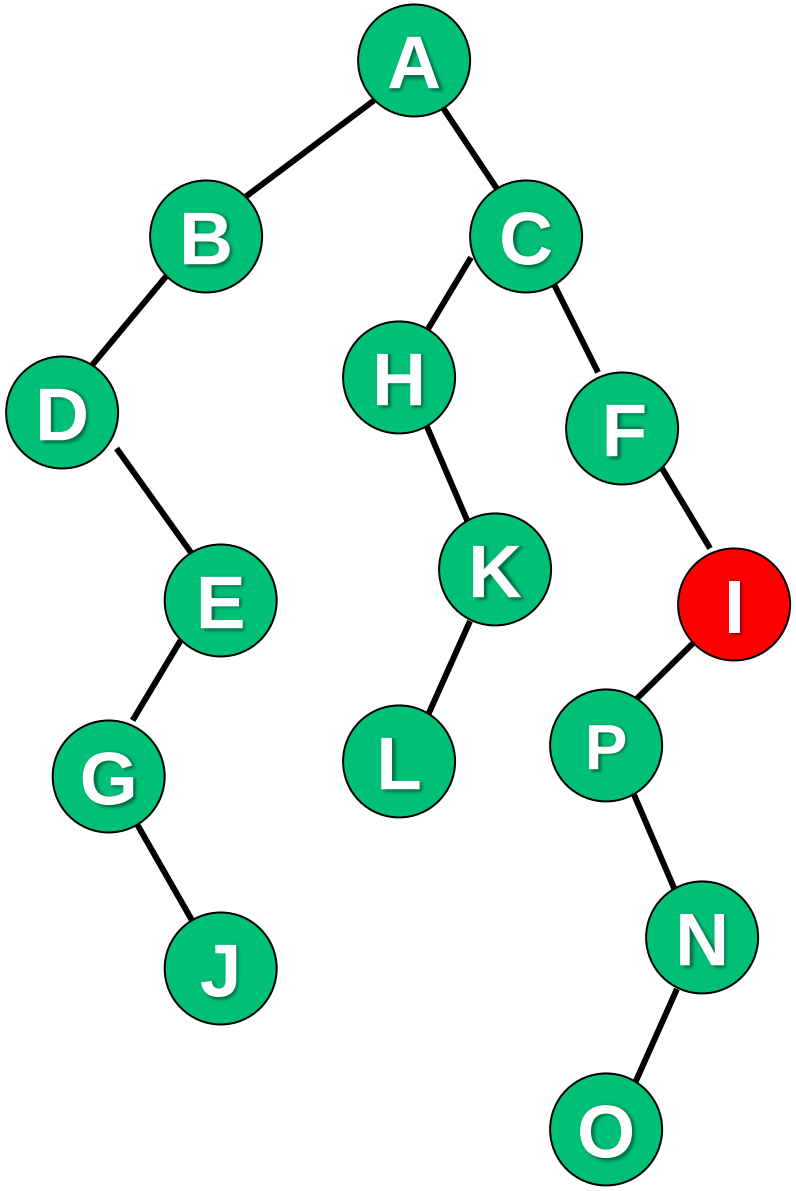
```
TreeNode* FirstInOrder(TreeNode* root){
    //在以root为根的中序线索二叉树中找中根序列首结点,返回指针
    TreeNode* p = root;
    while(p->LThread == 0)    //p有左孩子
        p = p->left;
    return p;
}
```



时间复杂度O(*h*)
***h*为二叉树高度**

找中根序列的最后一个结点

B

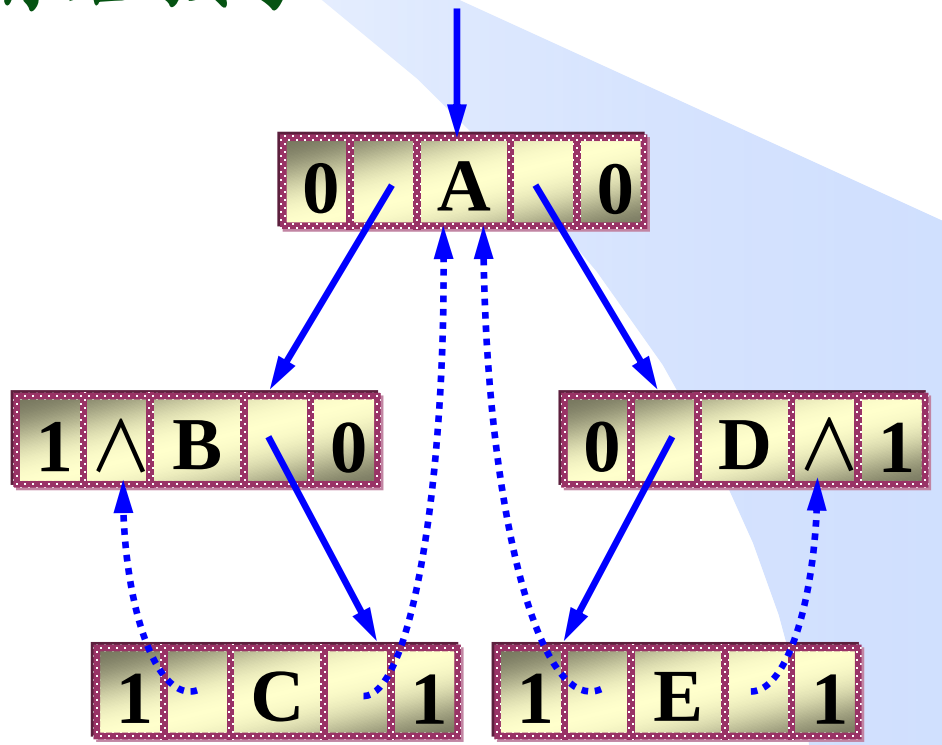


从根结点开始沿右分支下行，找第一个无右孩子的结点。

找中根序列的最后一个结点

B

```
TreeNode* LastInOrder(TreeNode* root){
    //在以root为根的中序线索二叉树中找中根序列末结点,返回指针
    TreeNode* p = root;
    while(p->RThread == 0) //p有右孩子
        p = p->right;
    return p;
}
```



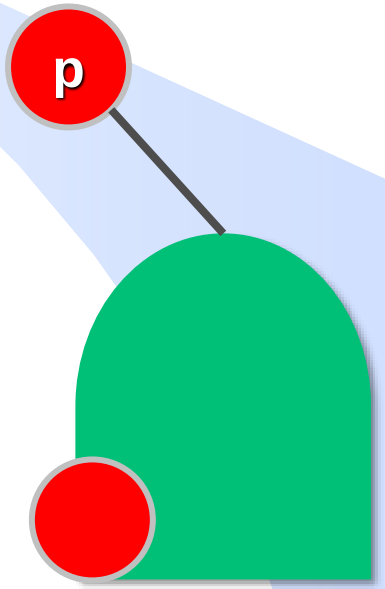
时间复杂度O(*h*)
***h*为二叉树高度**

在中序线索二叉树中，查找结点p的中根后继结点

B

- 若p->RThread为1，则p->right指向p的中根后继；
- 若p->RThread为0，则p的中根后继为p的右子树的中根序列的首结点。

```
TreeNode* NextInOrder(TreeNode* p){
    if(p->RThread==1) return p->right;
    return FirstInOrder(p->right);
}
```



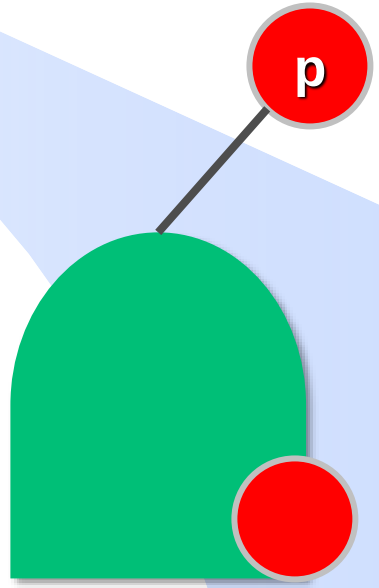
时间复杂度O(h)
h为二叉树高度

在中序线索二叉树中，查找结点p的中根前驱结点

B

- 若p->LThread为1，则p->left指向p的中根前驱；
- 若p->LThread为0，p的中根前驱结点是p的左子树的中根序列的最后一个结点。

```
TreeNode* PreInOrder(TreeNode* p){
    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

中序线索二叉树的中根遍历

B

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，直至其后继为空为止。

```
void InOrder(TreeNode *root) {
    for(TreeNode *p=FirstInOrder(root); p!=NULL; p=NextInOrder(p))
        visit(p->data);
}
```

时间复杂度O(*n*)

无需递归或栈
空间复杂度O(1)

二叉树的中序线索化

B

- 使二叉树变为线索二叉树的过程称为线索化。【大厂笔试题】
- 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”。
- 算法执行过程中，令指针*p*指向当前正在访问的结点，同时设置一个指针*pre*（作为全局变量）指向*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

B

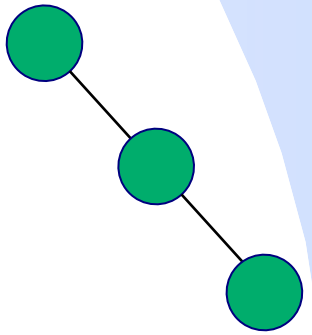
```
TreeNode *pre=NULL;
void Inorder_threading(TreeNode *p){ //中序线索化以p为根的二叉树
    if (p==NULL) return;
    Inorder_threading(p->left); //中序线索化p的左子树
    if(p->left==NULL){p->LThread=1; p->left=pre;}
    else p->LThread=0;
    if(pre!=NULL && pre->right==NULL){pre->RThread=1;pre->right=p;}
    else if(pre!=NULL) pre->RThread=0;
    pre=p;    //pre指向刚访问完的点
    Inorder_threading(p->right); //中序线索化p的右子树
}
```

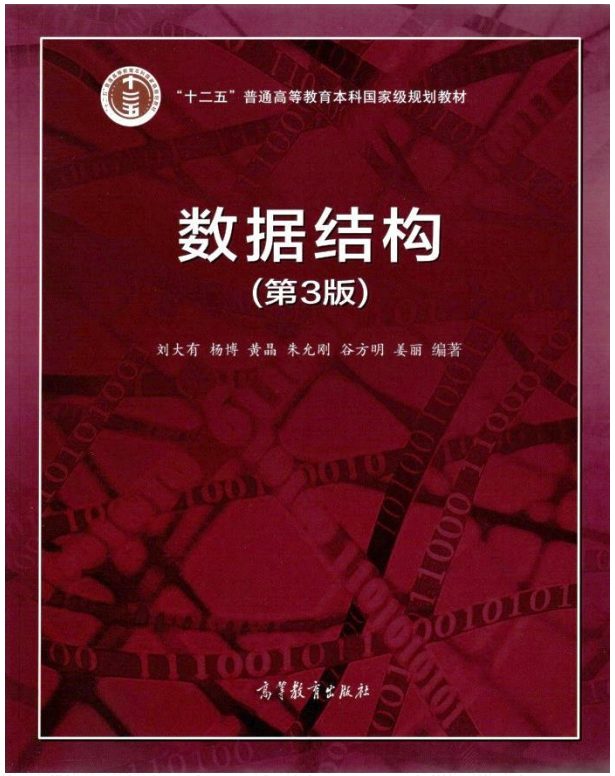
如果p无左孩子，p的左指针是线索，指向中根前驱pre

如果pre无右孩子，pre的右指针是线索，指向其中根后继p

算法结束后要调整中根序列末结点（pre指向）的RThread:

```
pre->RThread=1;
```





线索二叉树

动机和基本概念

中序线索二叉树的基本操作

问题与拓展

数据之法

结构之美

算法之道

zhuyungang@jlu.edu.cn

Last updated on 2024.10

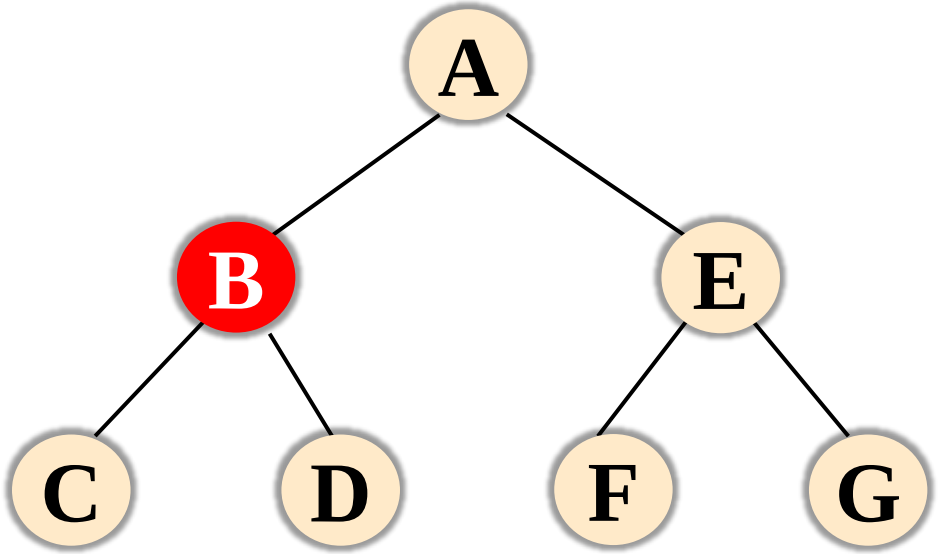
关于前序/后序线索二叉树的问题

B

LThread	*Left*	*Data*	*Right*	*RThread*

若结点结构中没有父指针：

✓前序线索二叉树不能解决高效查找结点的先根前驱的问题。



✓ B的先根前驱：*A*

✓ 无法通过*B*的*Left*/*Right*指针找到*A*

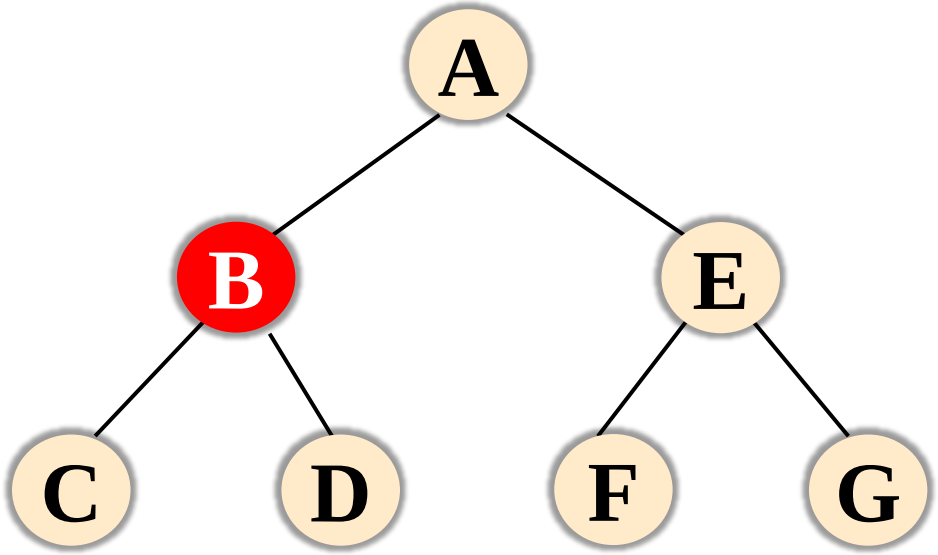
关于前序/后序线索二叉树的问题

B

LThread	Left	Data	Right	RThread

若结点结构中没有父指针：

✓后序线索二叉树不能解决高效查找结点的后根后继的问题。



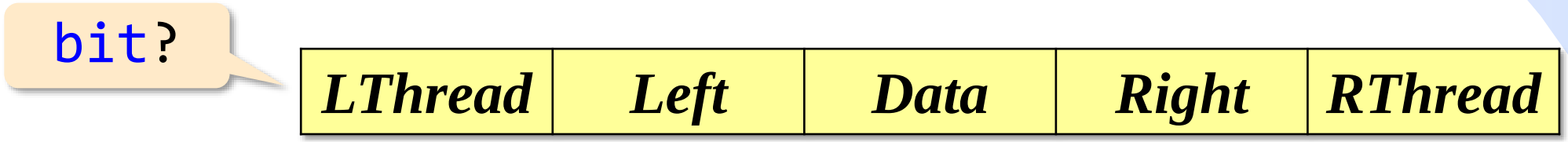
✓ B的后根后继：*F*

✓ 无法通过*B*的*Left*/*Right*指针找到*F*

线索域编程实现的问题

C

- *LThread*和*RThread*理论上为1比特（bit），但是包括C/C++语言在内的众多程序设计语言无法定义1 bit的变量。



一种替代方案——扩展的线索二叉树

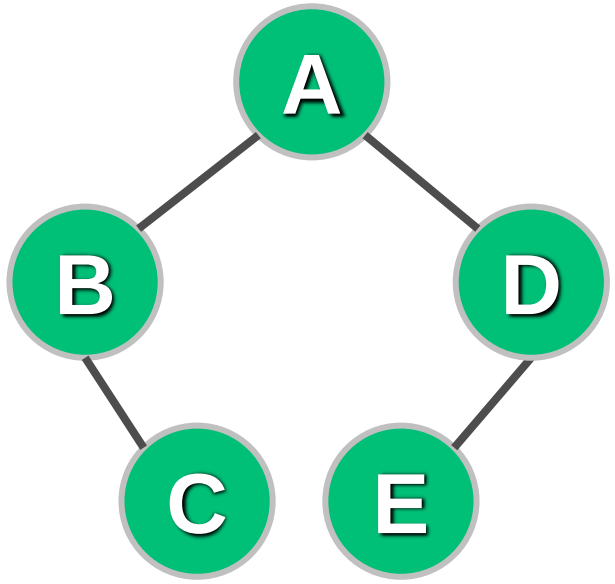
B

- 把*LThread* 和 *Rthread*换成指针域，*Pred*指向某种遍历序的前驱结点，*Succ*指向后继结点。
- 空间换时间：虽然存储空间增加，但使找某种遍历序的前驱/后继的最坏时间复杂度降为O(1)。

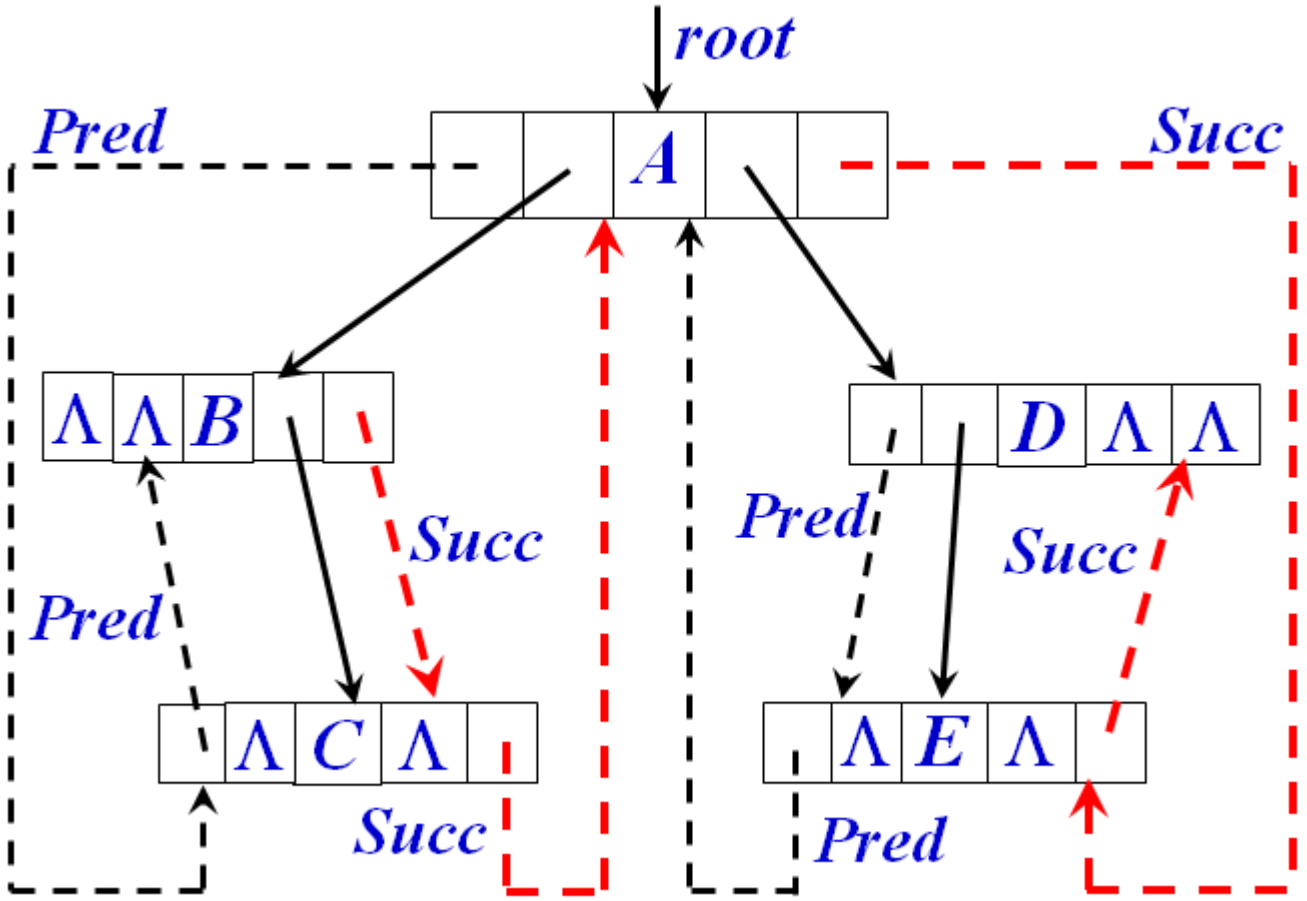
Pred	*Left*	*Data*	*Right*	*Succ*

中序扩展线索二叉树

中根序列：BCAED



Pred	Left	Data	Right	Succ



图中虚线箭头表示线索