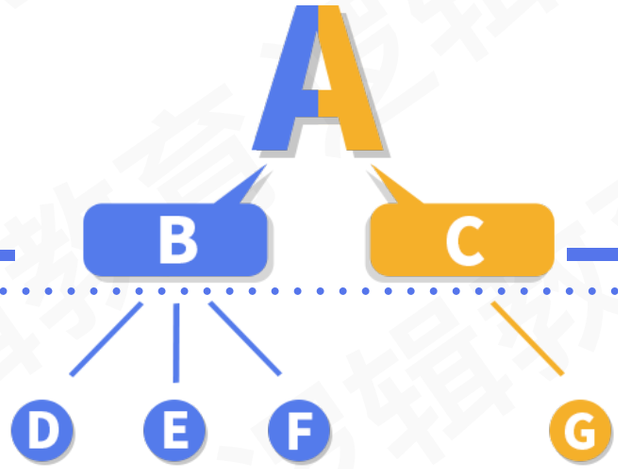
逻辑教育
Logic education

Hello 数据结构与算法



数据结构与算法—树和二叉树

数据结构与算法主题

@CC老师

全力以赴．非同凡"想"

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

树结构在客观世界中应用

赌王何鸿燊的族谱图(来自网络)

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

树结构在客观世界中应用

阿里巴巴集团CEO，董事局主席马云(风清扬)

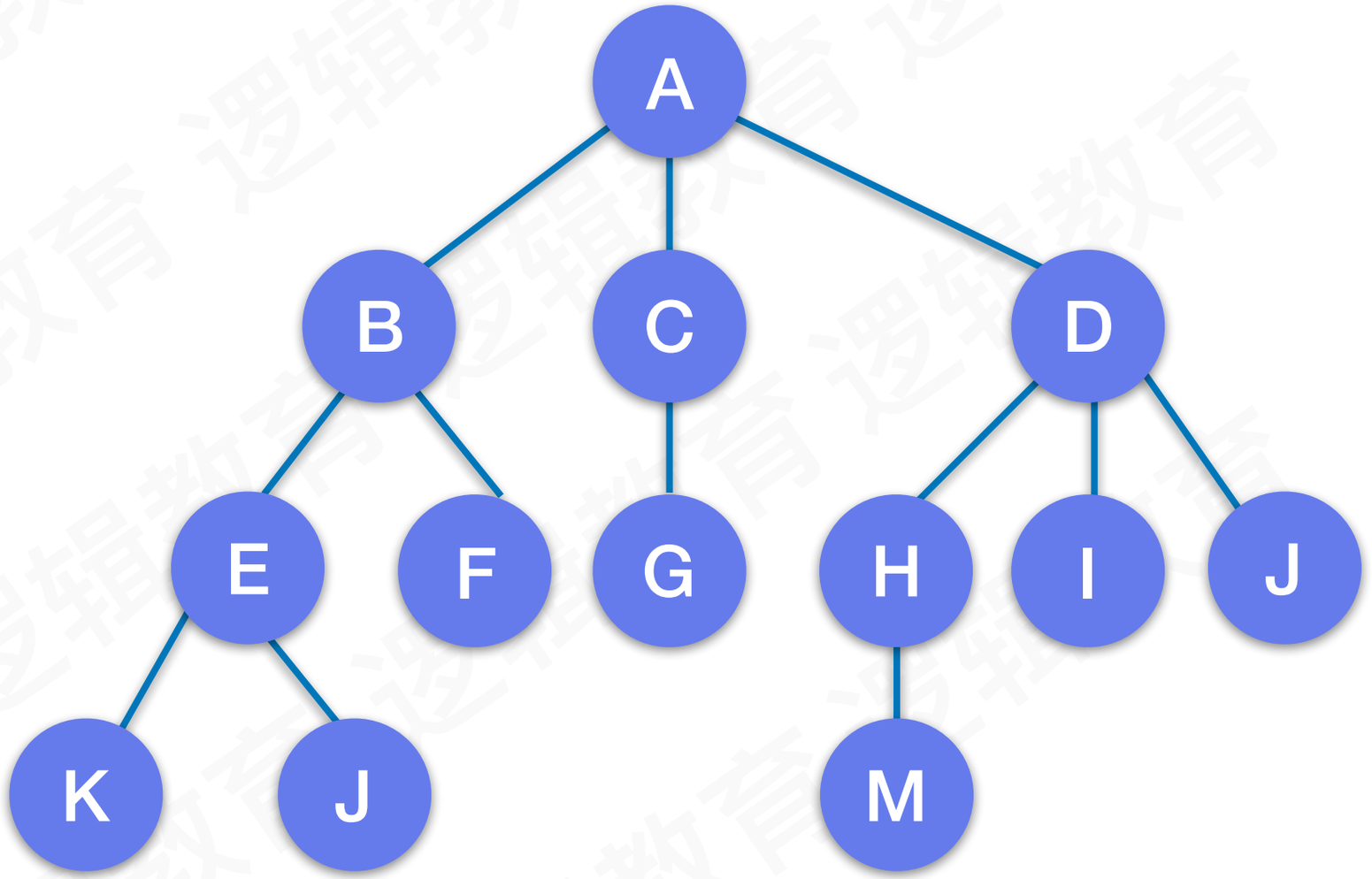
阿里组织架构图(来自网络)

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

树结构

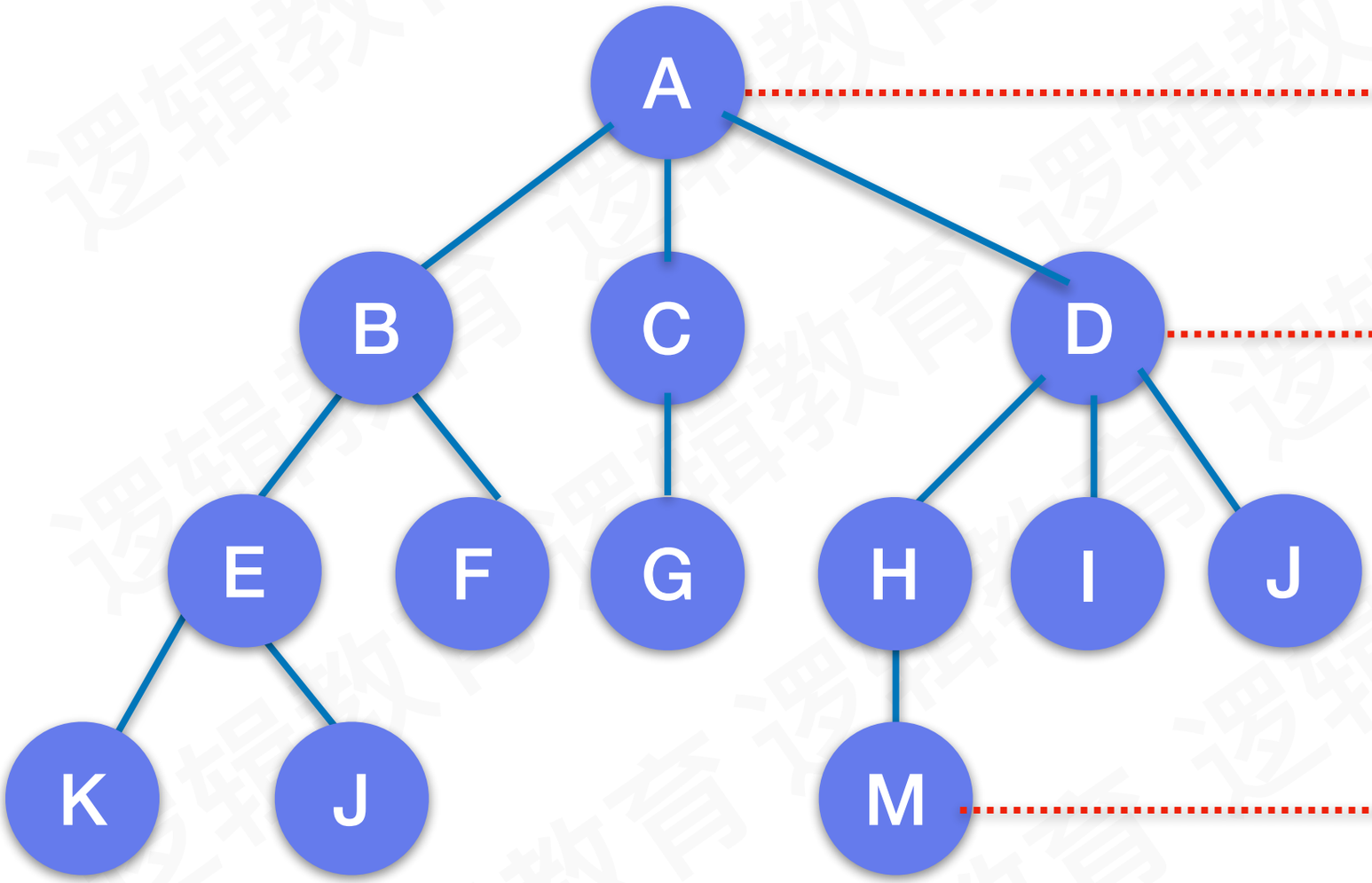
只有根结点的树



一般树

课程研发:CC老师
课程授课:CC老师

树结构中的高度,深度以及层数



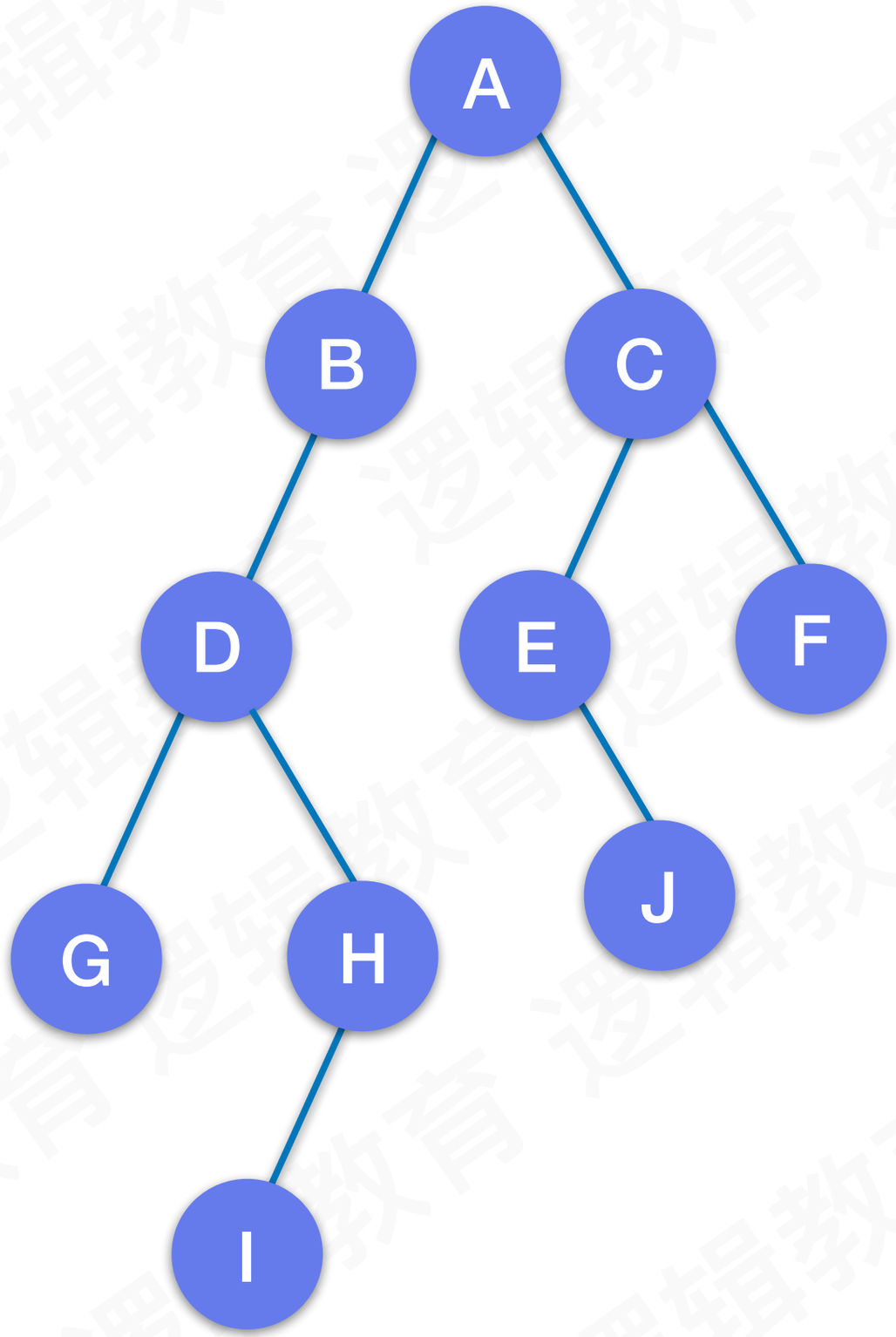
	高度	深度	层
A	3	0	1
B C D	2	1	2
E F G H I J	1	2	3
K J M	0	3	4

一般树

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

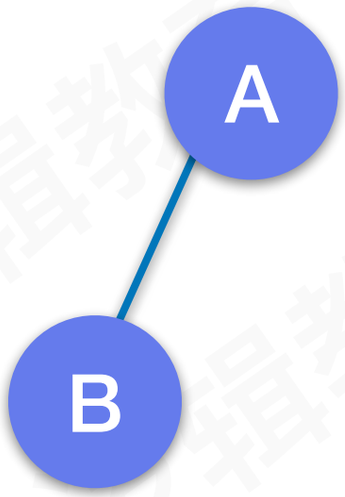
二叉树



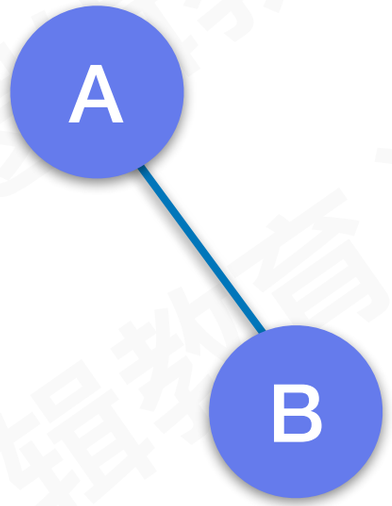
课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的特性



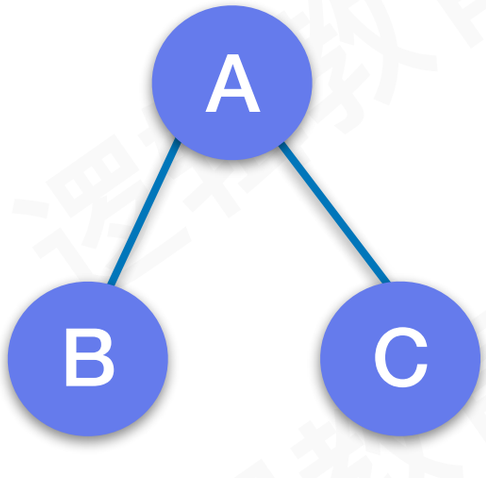
树1



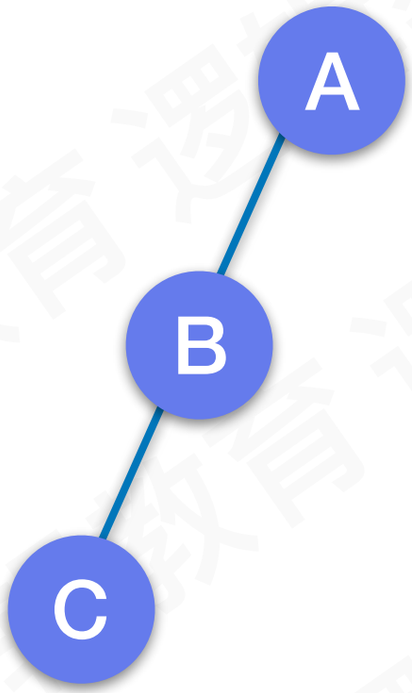
树2

课程研发:CC老师
课程授课:CC老师

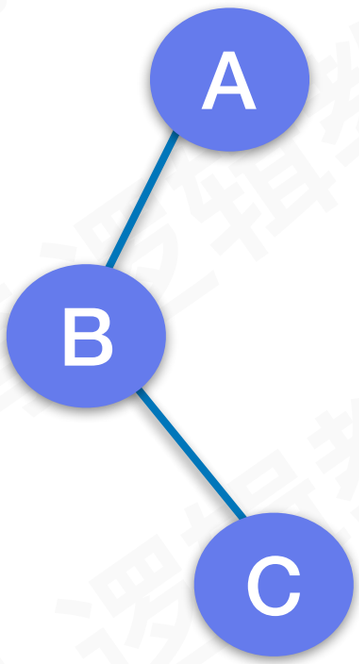
二叉树的五种形态



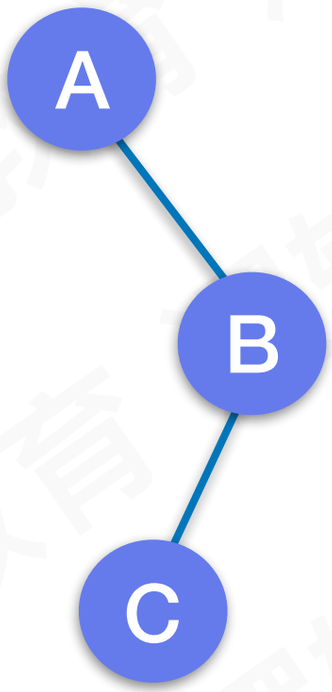
树1



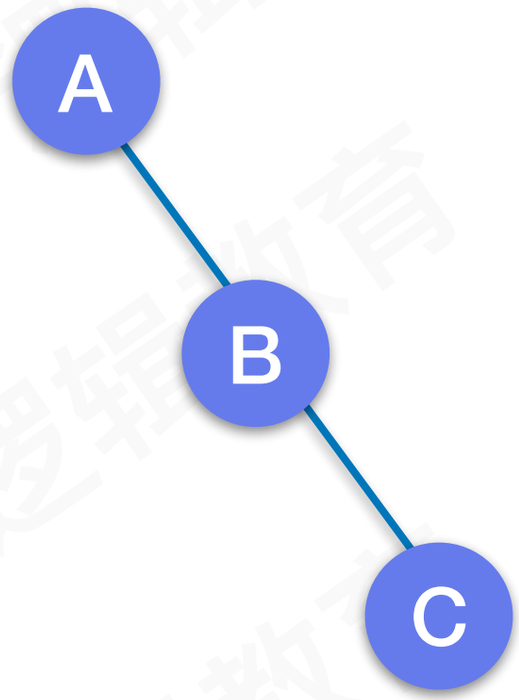
树2



树3



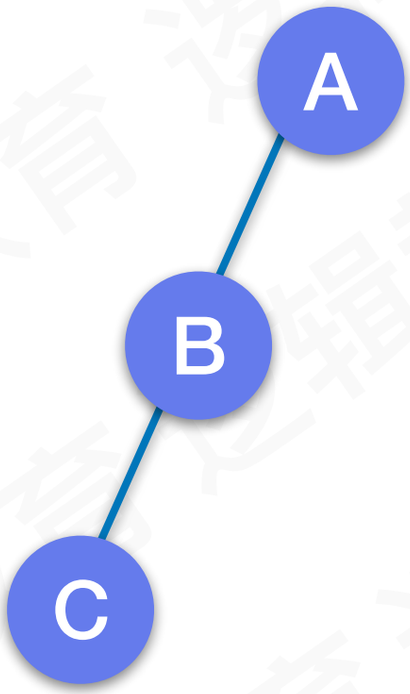
树4



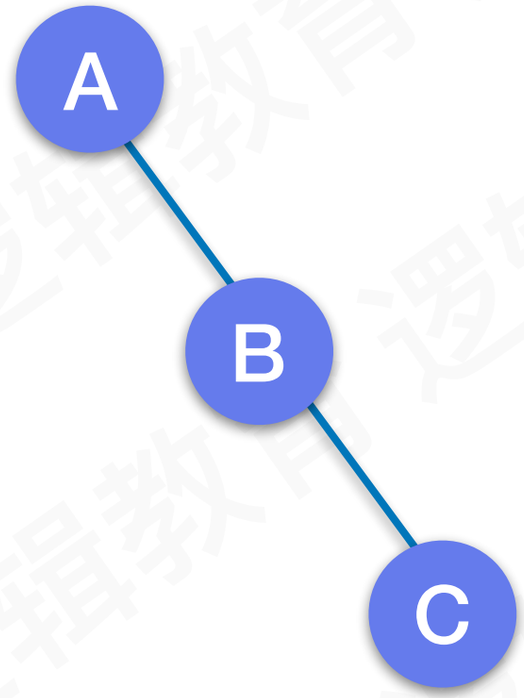
树5

课程研发:CC老师
课程授课:CC老师

特殊二叉树—斜树



树2

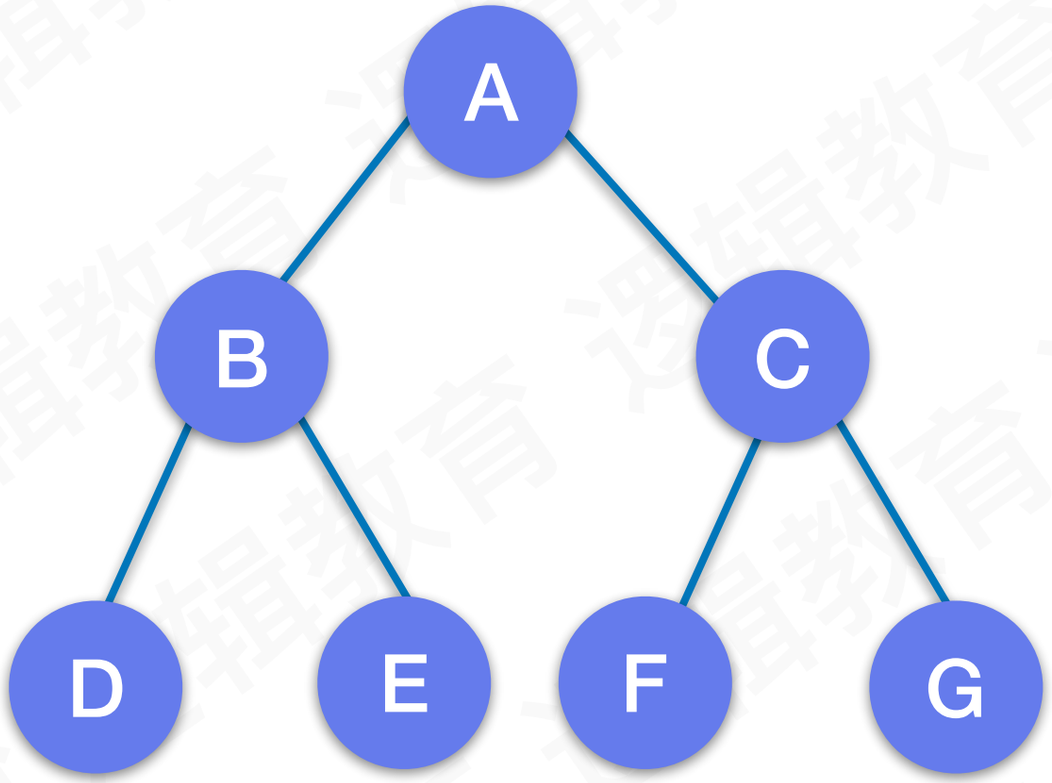


树5

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

特殊二叉树—满二叉树

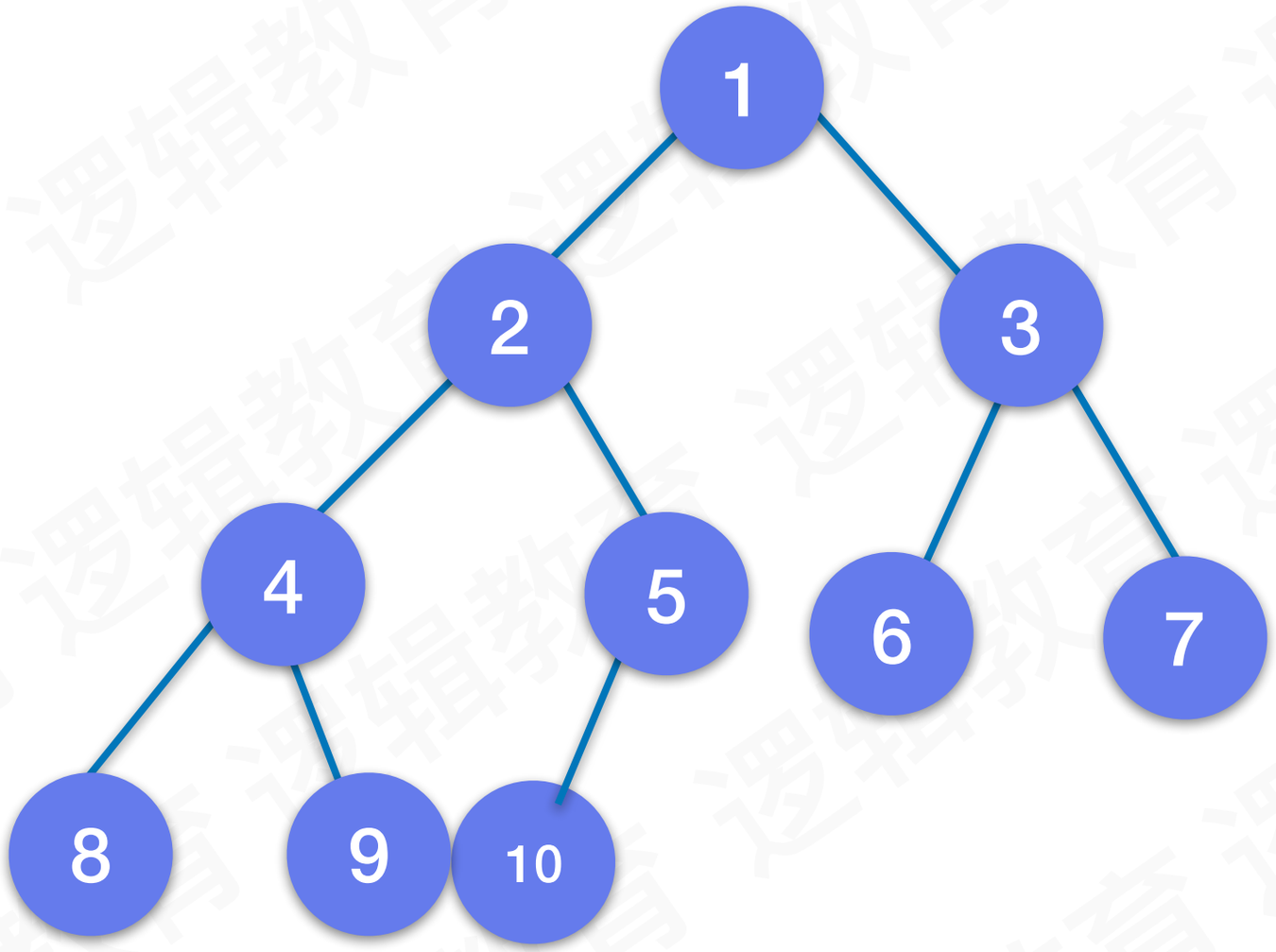


满二叉树

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

特殊二叉树—完全二叉树

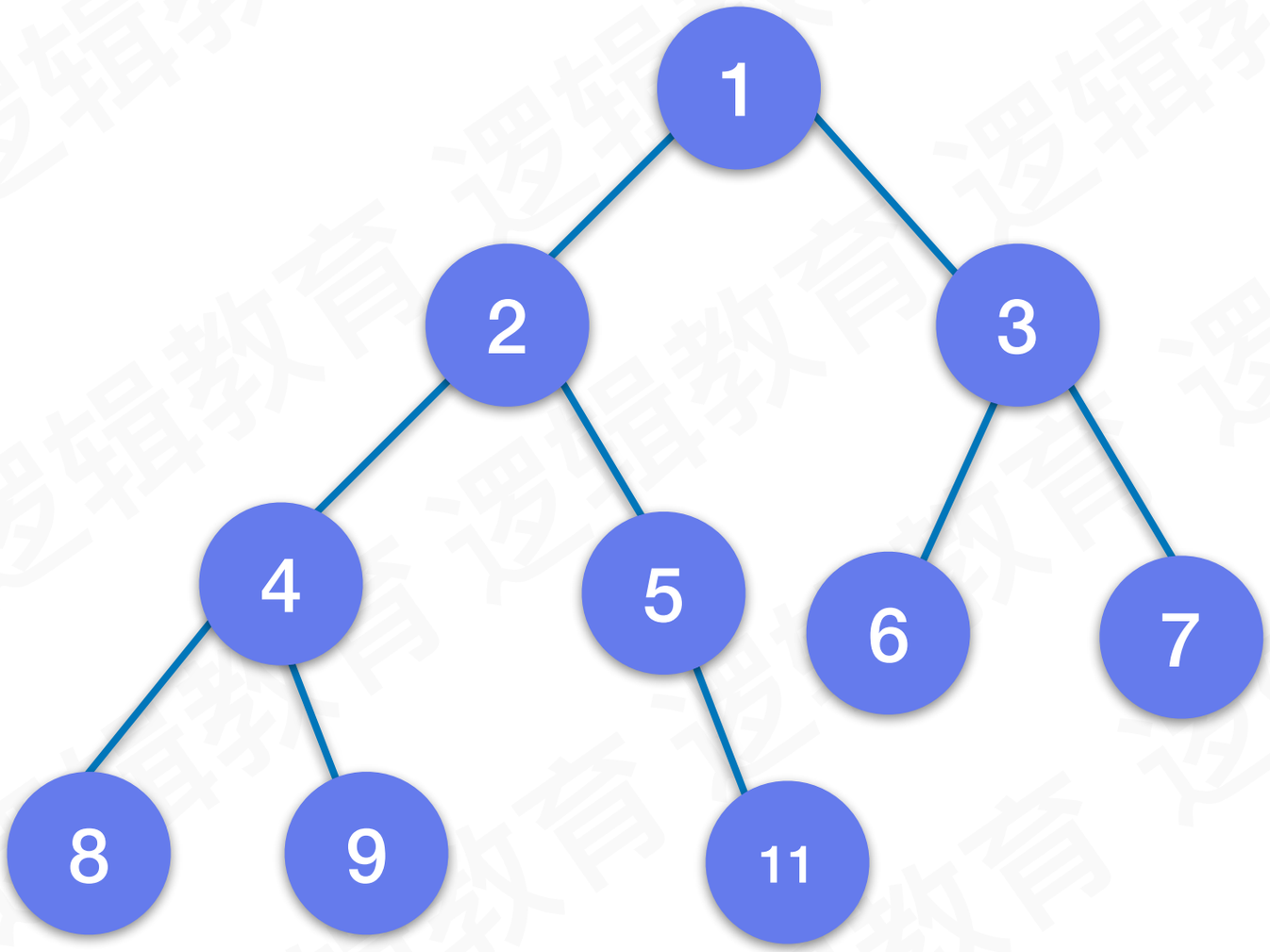


完全二叉树

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

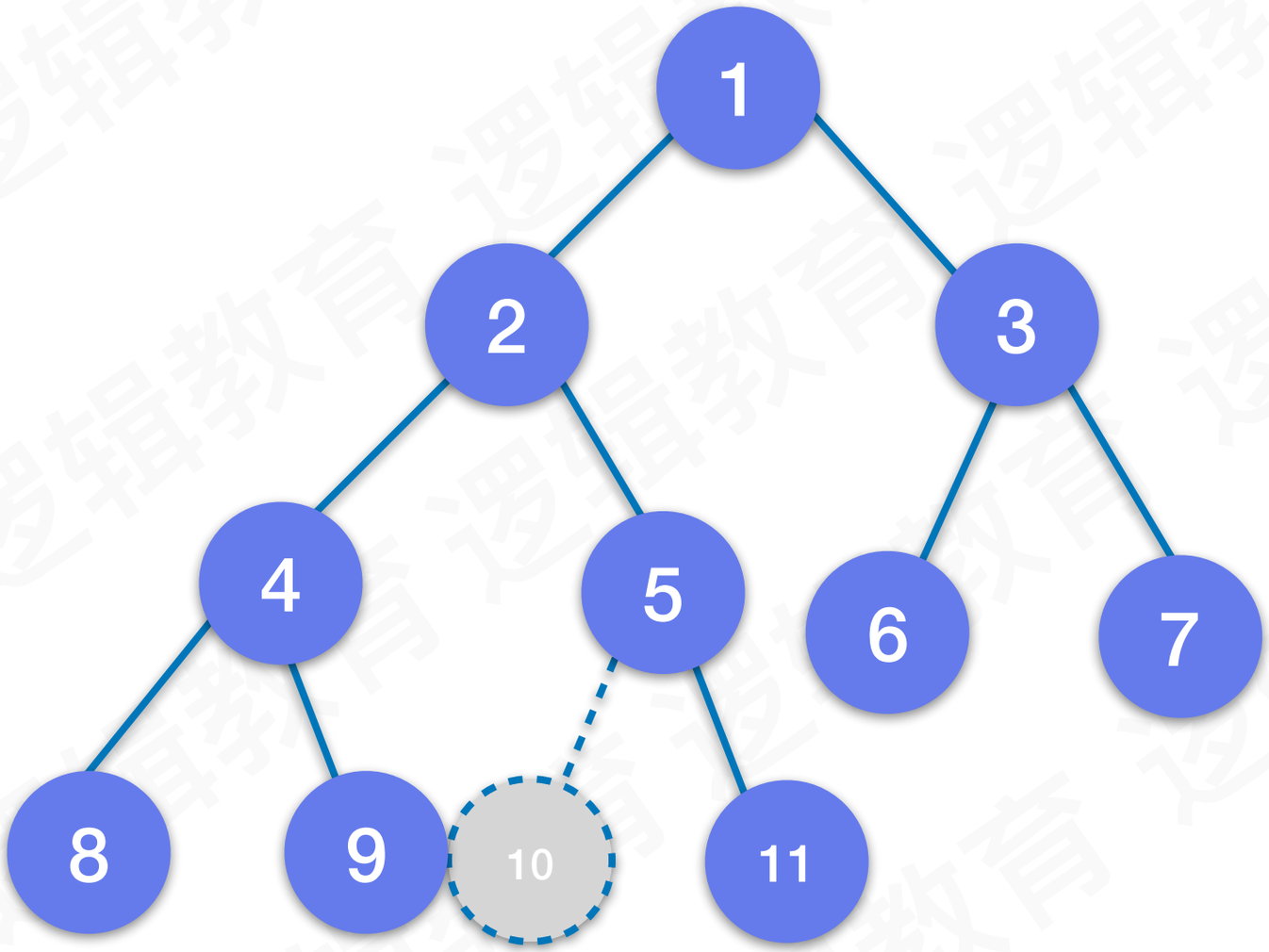
特殊二叉树—完全二叉树的判断



课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

特殊二叉树—完全二叉树的判断

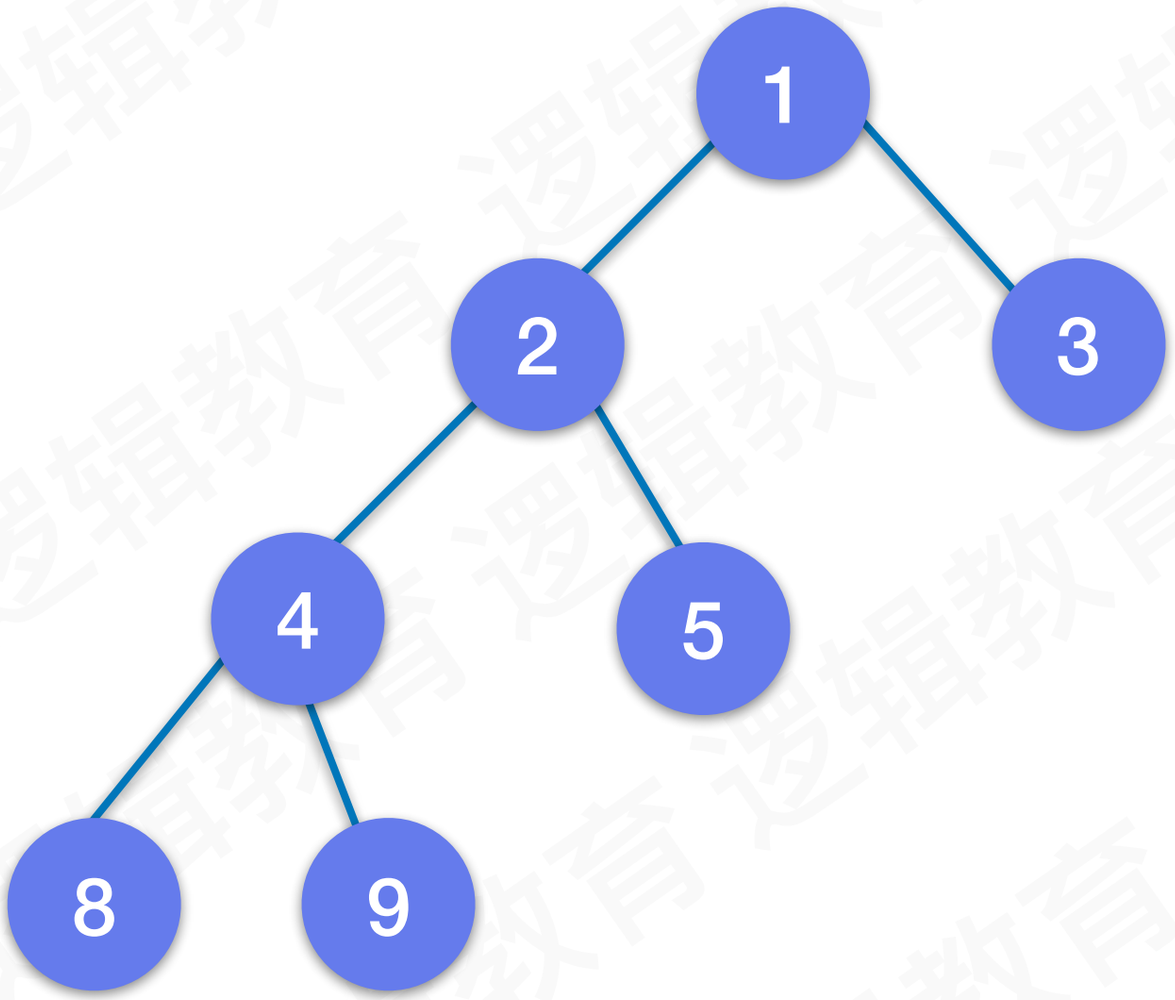


不是完全二叉树

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

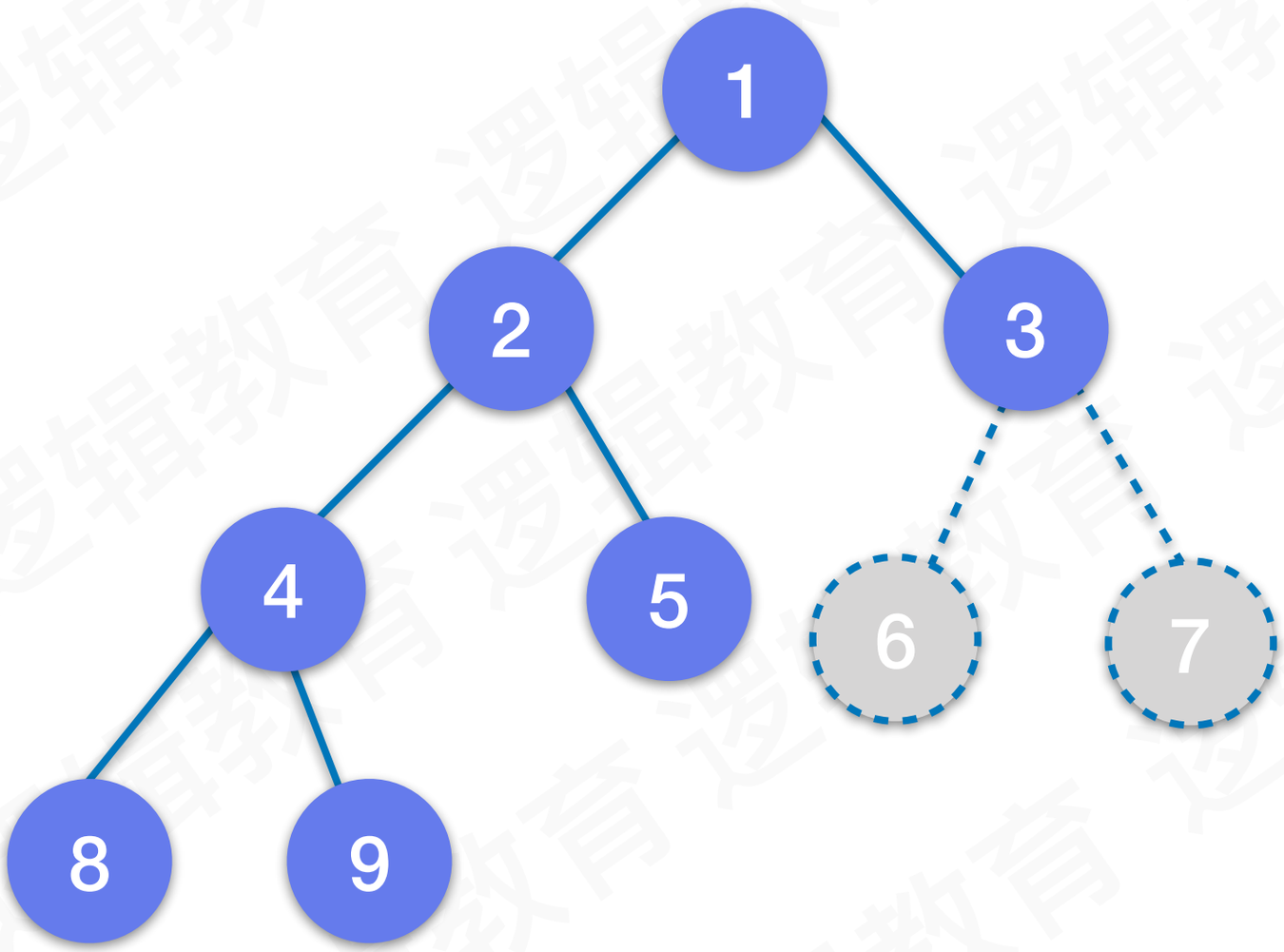
特殊二叉树—完全二叉树的判断



课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

特殊二叉树—完全二叉树的判断

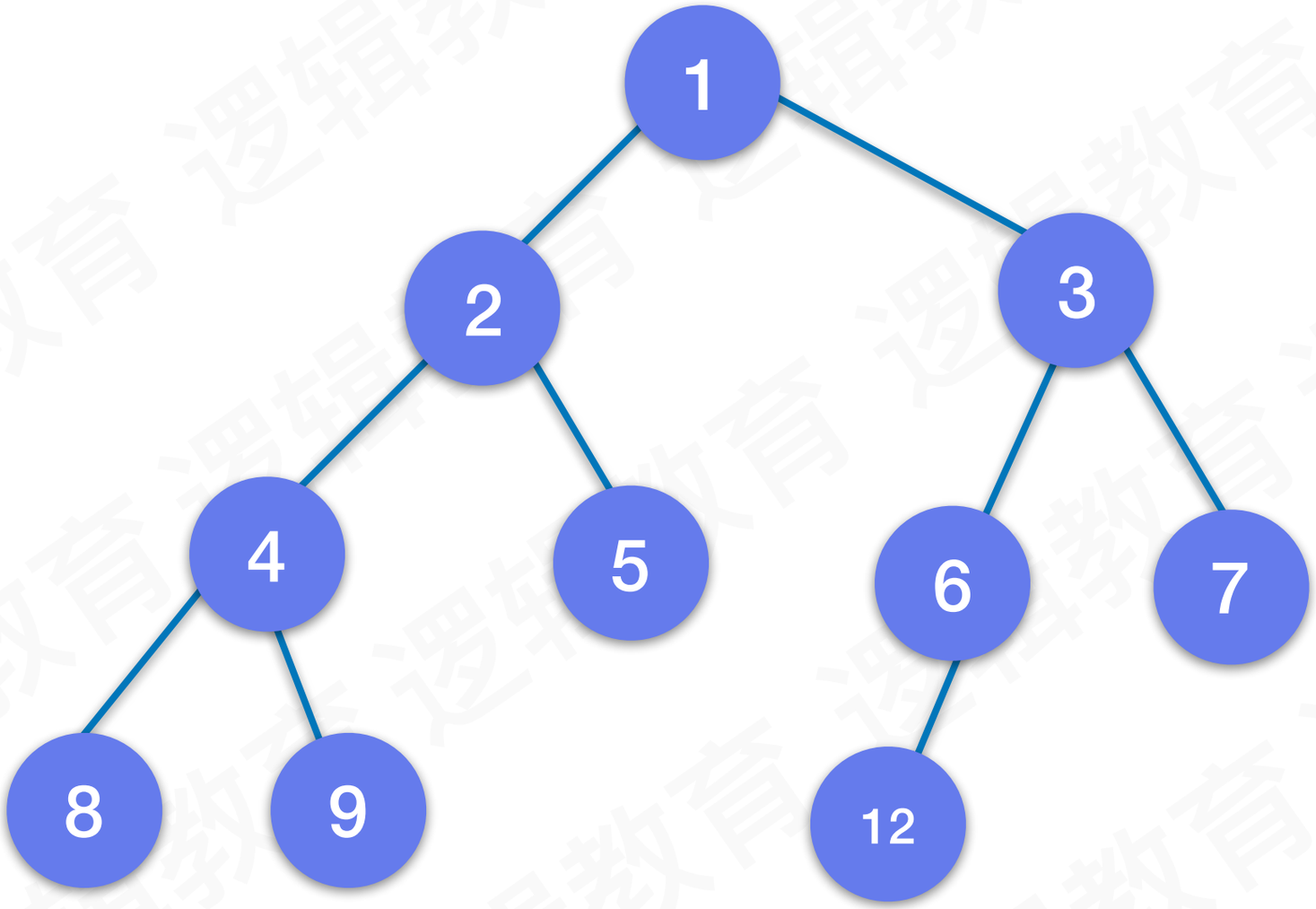


不是完全二叉树

课程研发:CC老师
课程授课:CC老师

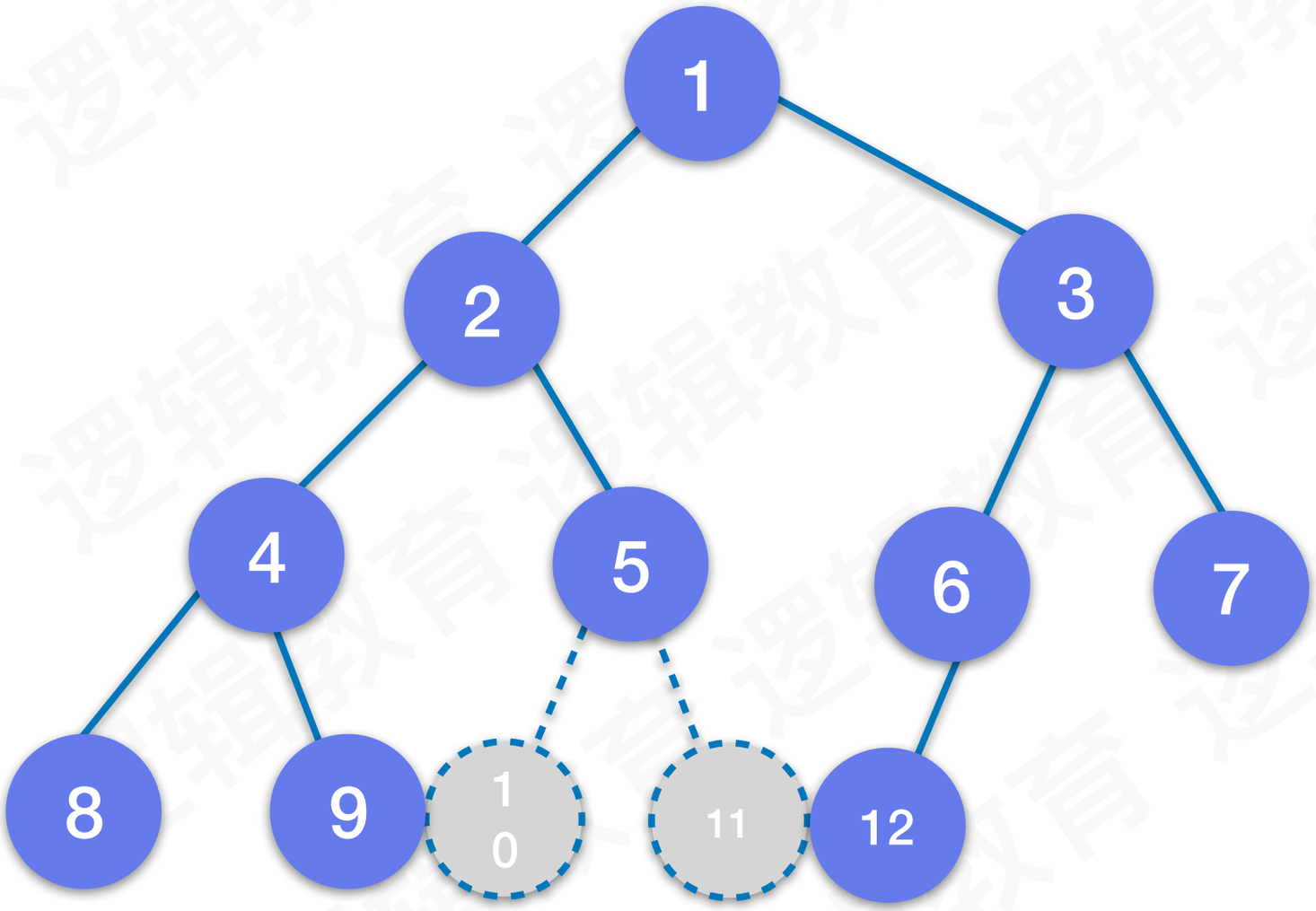
逻辑教育
Logic education

特殊二叉树—完全二叉树的判断



课程研发:CC老师
课程授课:CC老师

特殊二叉树—完全二叉树的判断

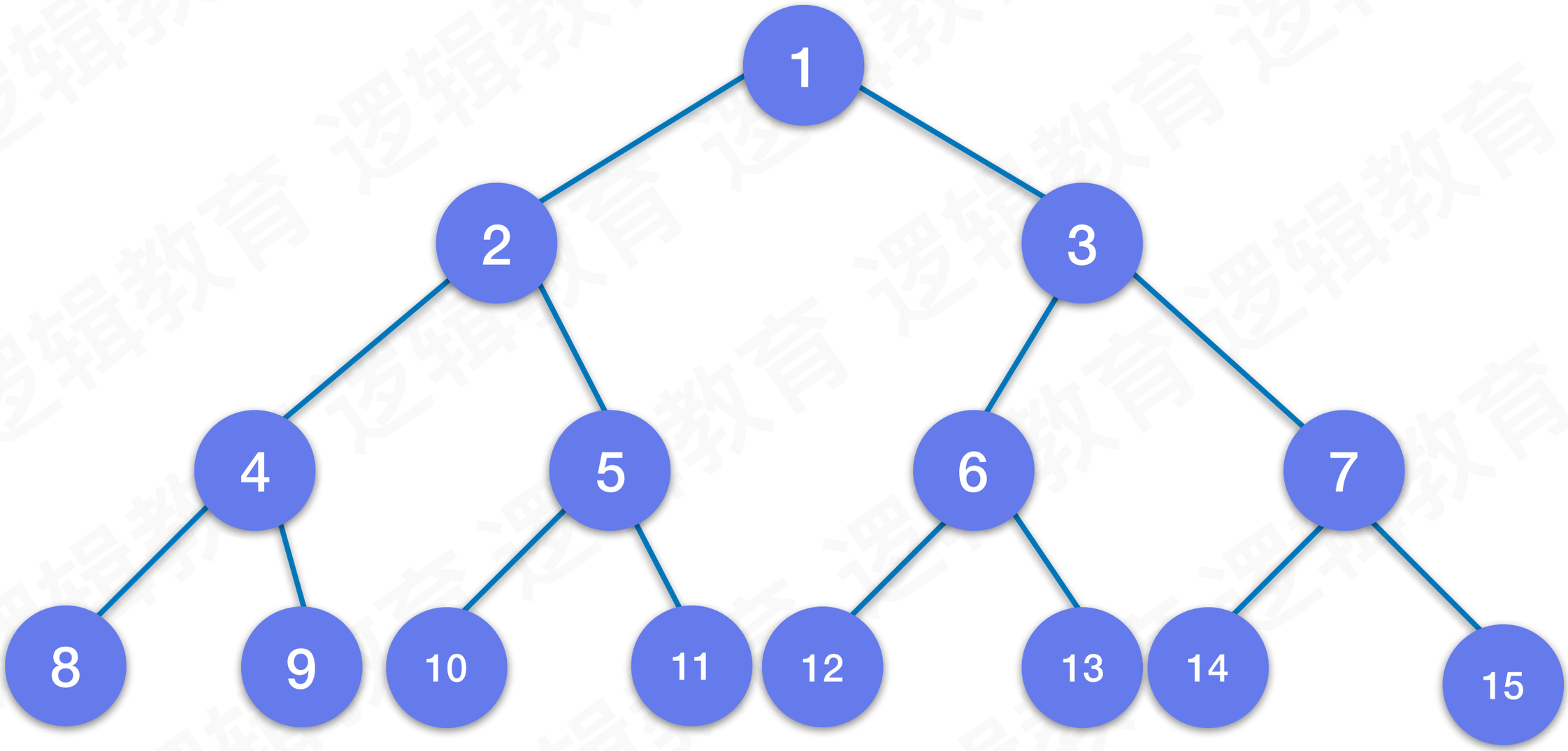


不是完全二叉树

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

特殊二叉树—完全二叉树的判断



课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的性质

1. 性质1: 在二叉树的第i层上最多有2^{i-1}个结点
2. 性质2: 深度为K的二叉树最多有$2^k - 1$ 个结点(K>=1)
3. 性质3: 对于任何一颗二叉树T,如果其终端结点数为n0,度为2的结点数为n2,则n0 = n2 + 1;
4. 性质4: 具有n个结点的完全二叉树深度为(log2(n))+1

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的性质

5.性质5:对具有n个结点的完全二叉树，如果按照从上至下和从左至右的顺序对二叉树的所有结点从1开始编号，则对于任意的序号为i的结点有:

A.如果i>1，那么序号为i的结点的双亲结点序号为i/2；

B.如果i=1，那么序号为i的结点为根节点，无双亲结点；

C.如果2i<=n，那么序号为i的结点的左孩子结点序号为2i；

D.如果2i>n，那么序号为i的结点无左孩子；

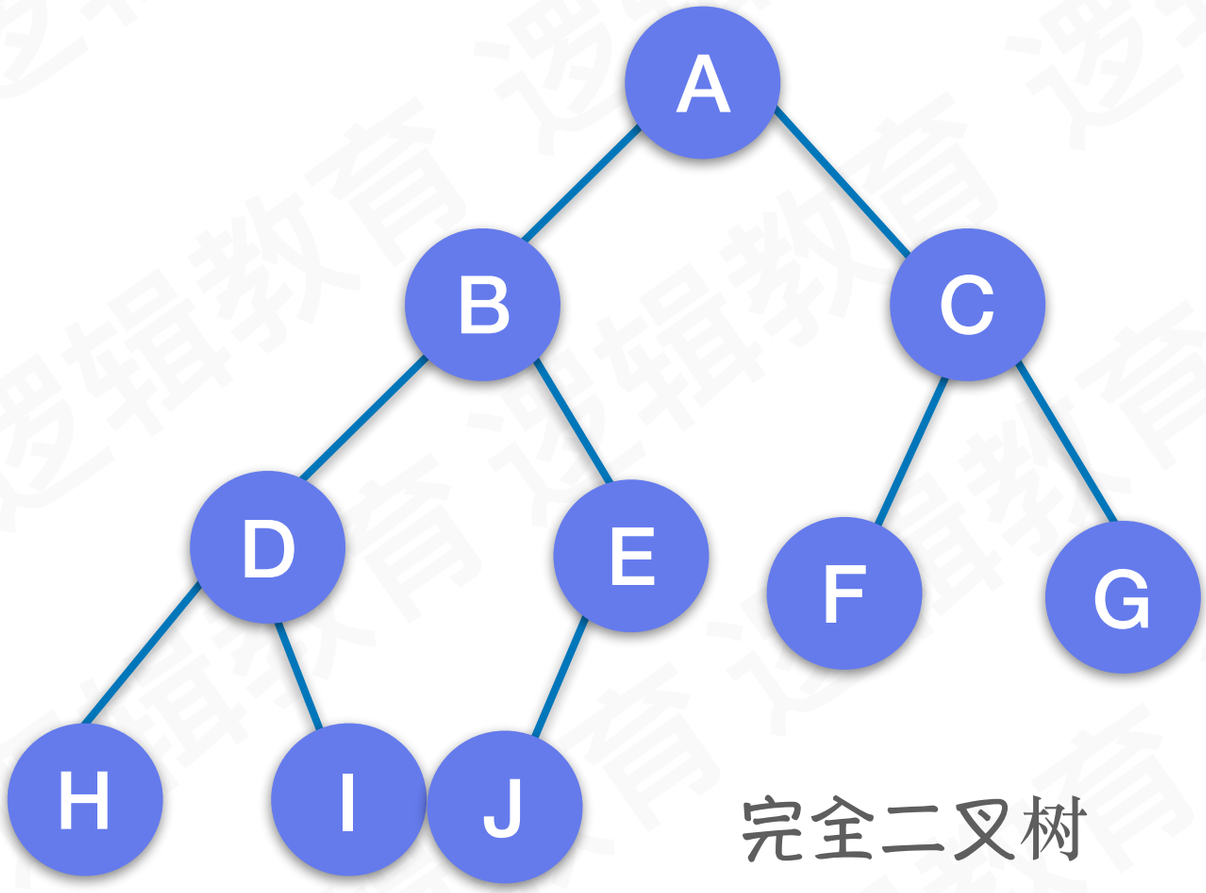
E.如果2i+1<=n，那么序号为i的结点右孩子序号为2i+1；

F.如果2i+1>n，那么序号为i的结点无右孩子。

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的顺序存储结构分析

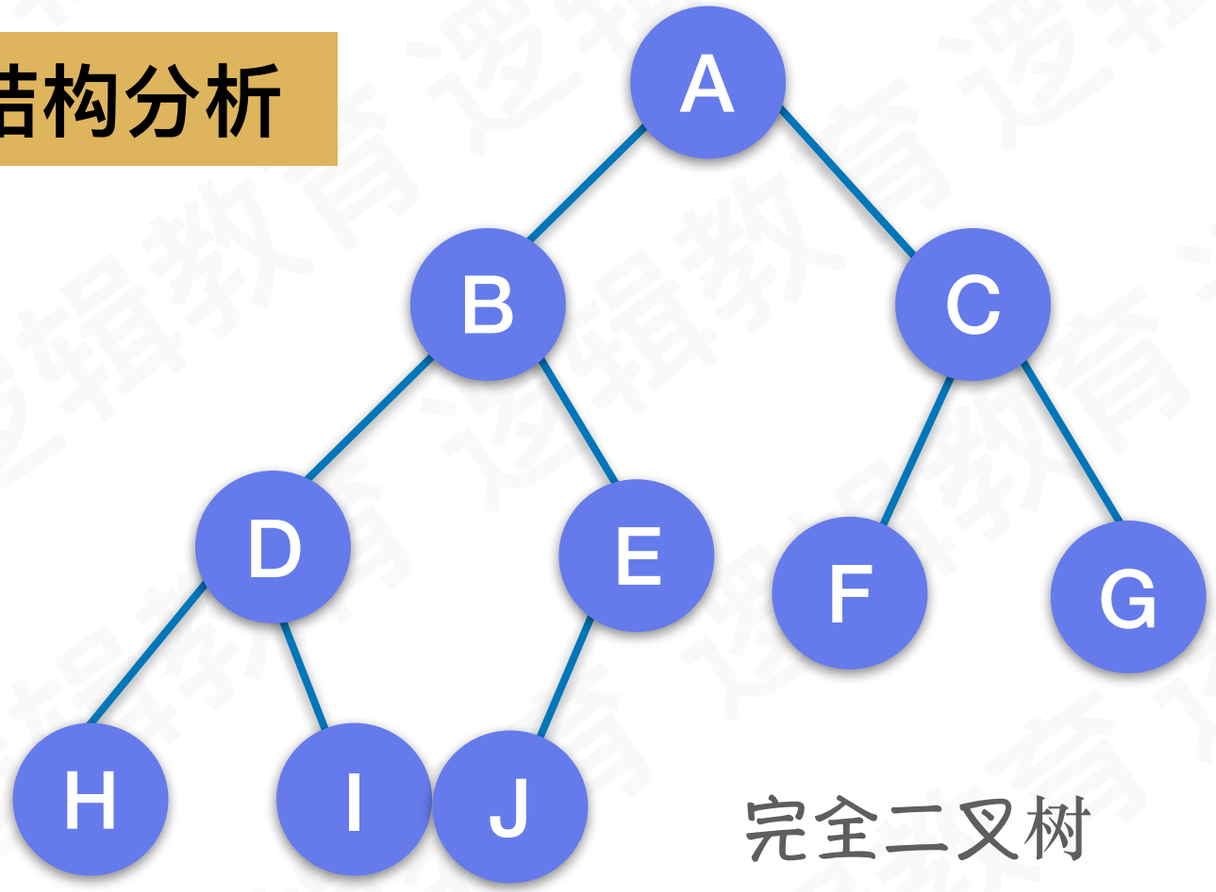


完全二叉树

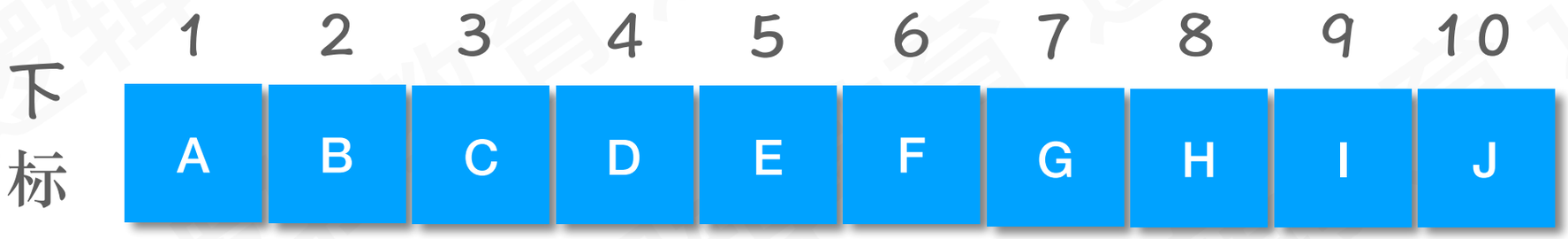
思考: 此完全二叉树应该如何存储到顺序存储结构中?

课程研发:CC老师
课程授课:CC老师

二叉树的顺序存储结构分析

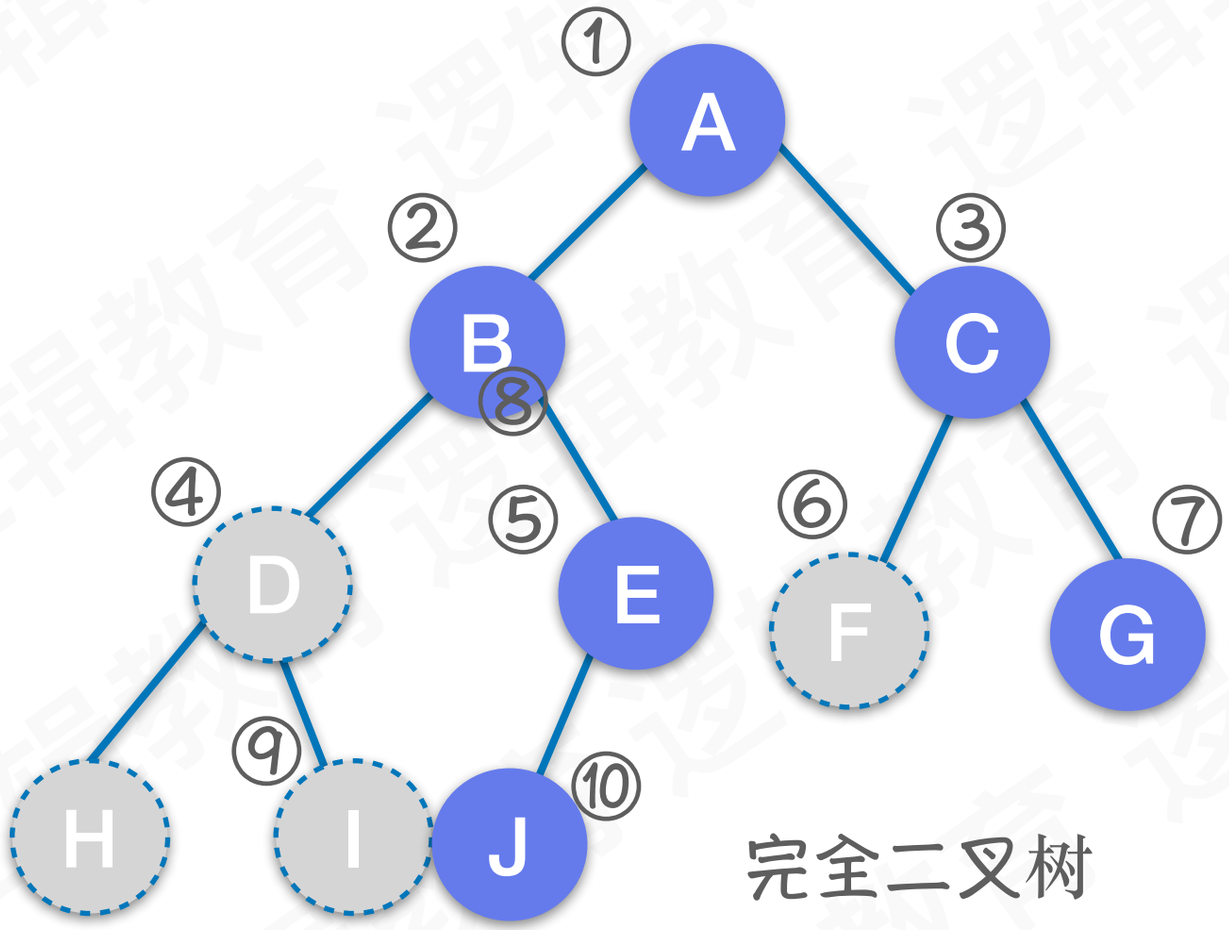


完全二叉树



课程研发:CC老师
课程授课:CC老师

二叉树的顺序存储结构分析



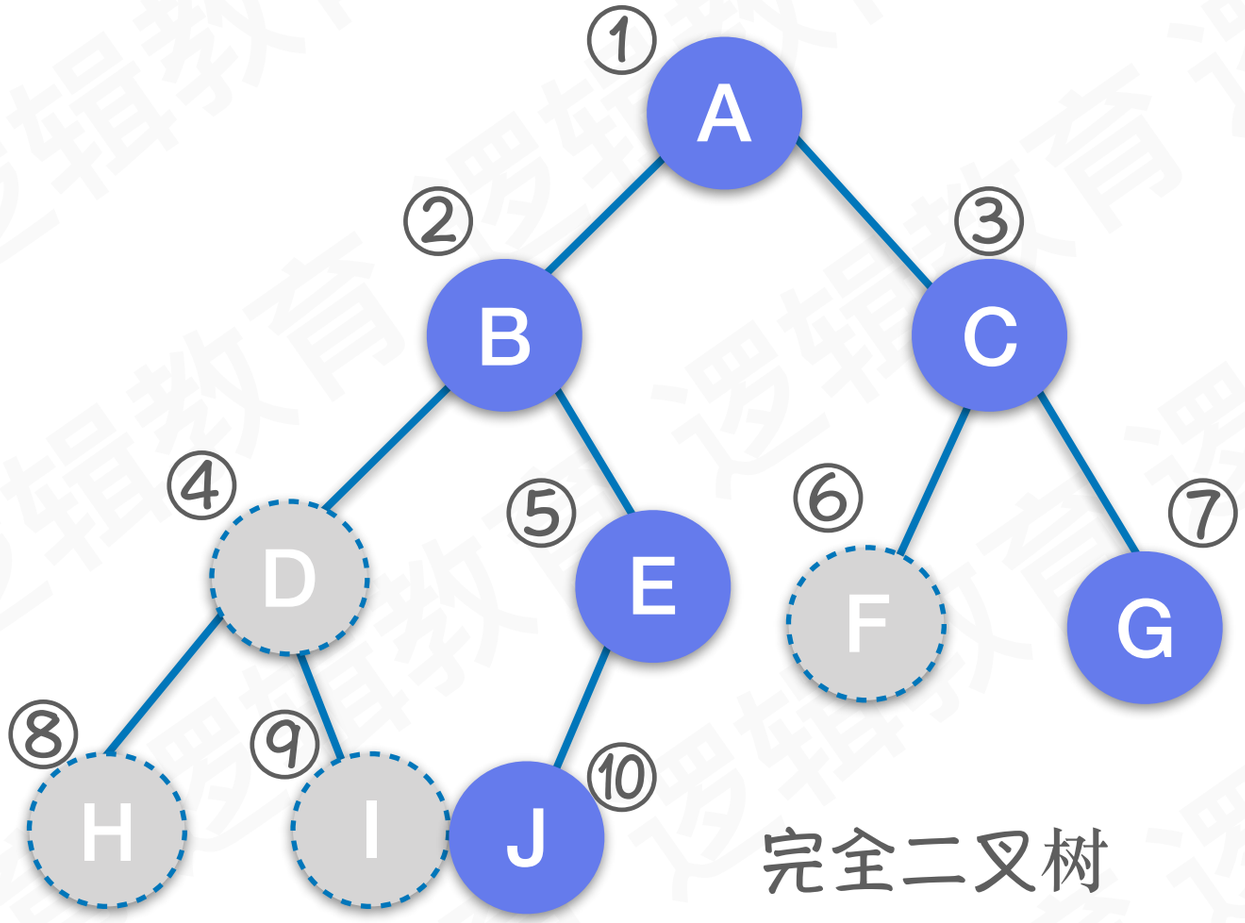
完全二叉树

思考: 此完全二叉树应该如何存储到顺序存储结构中?

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的顺序存储结构分析

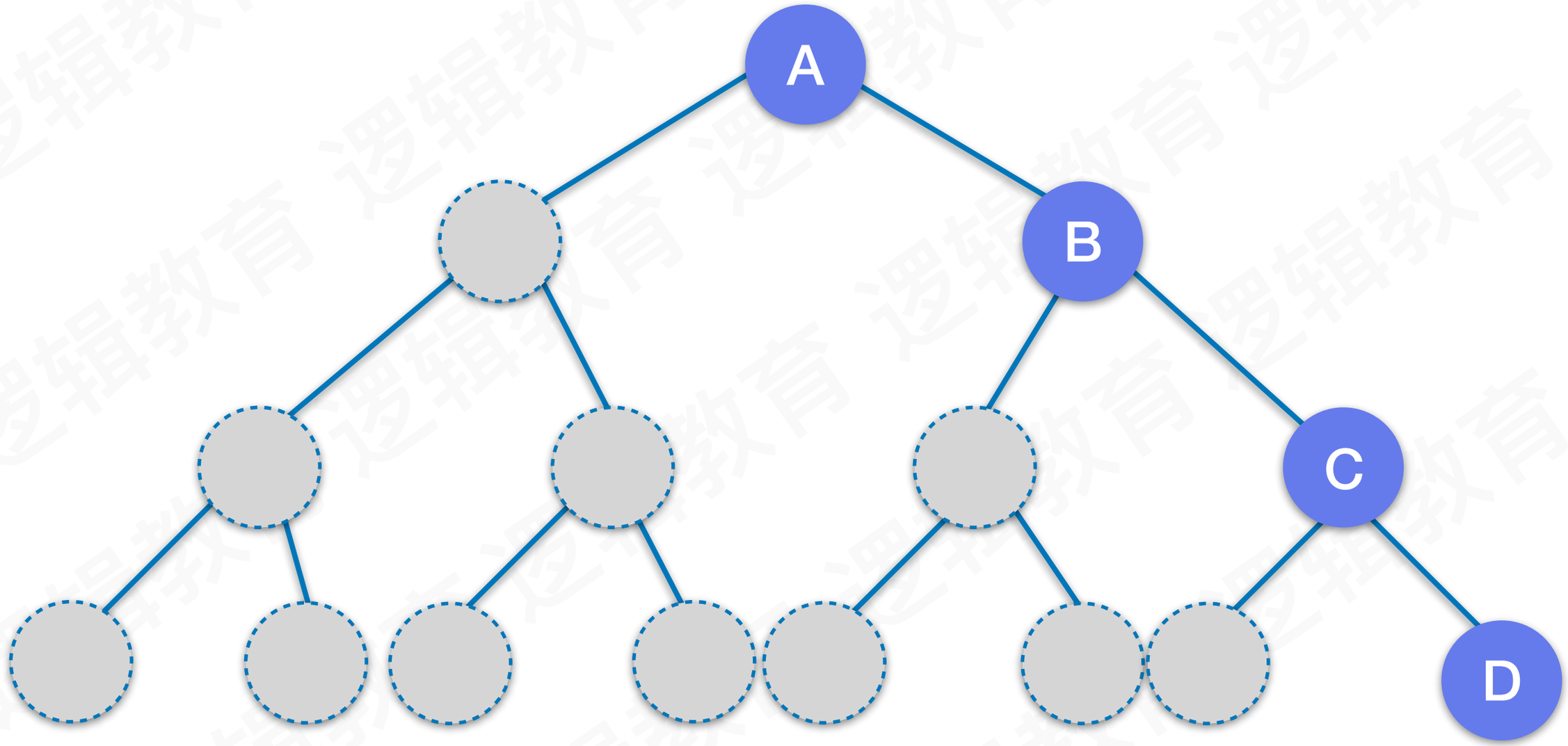


完全二叉树

下标	1	2	3	4	5	6	7	8	9	10
	A	B	C	^	E	^	G	^	^	J

课程研发:CC老师
课程授课:CC老师

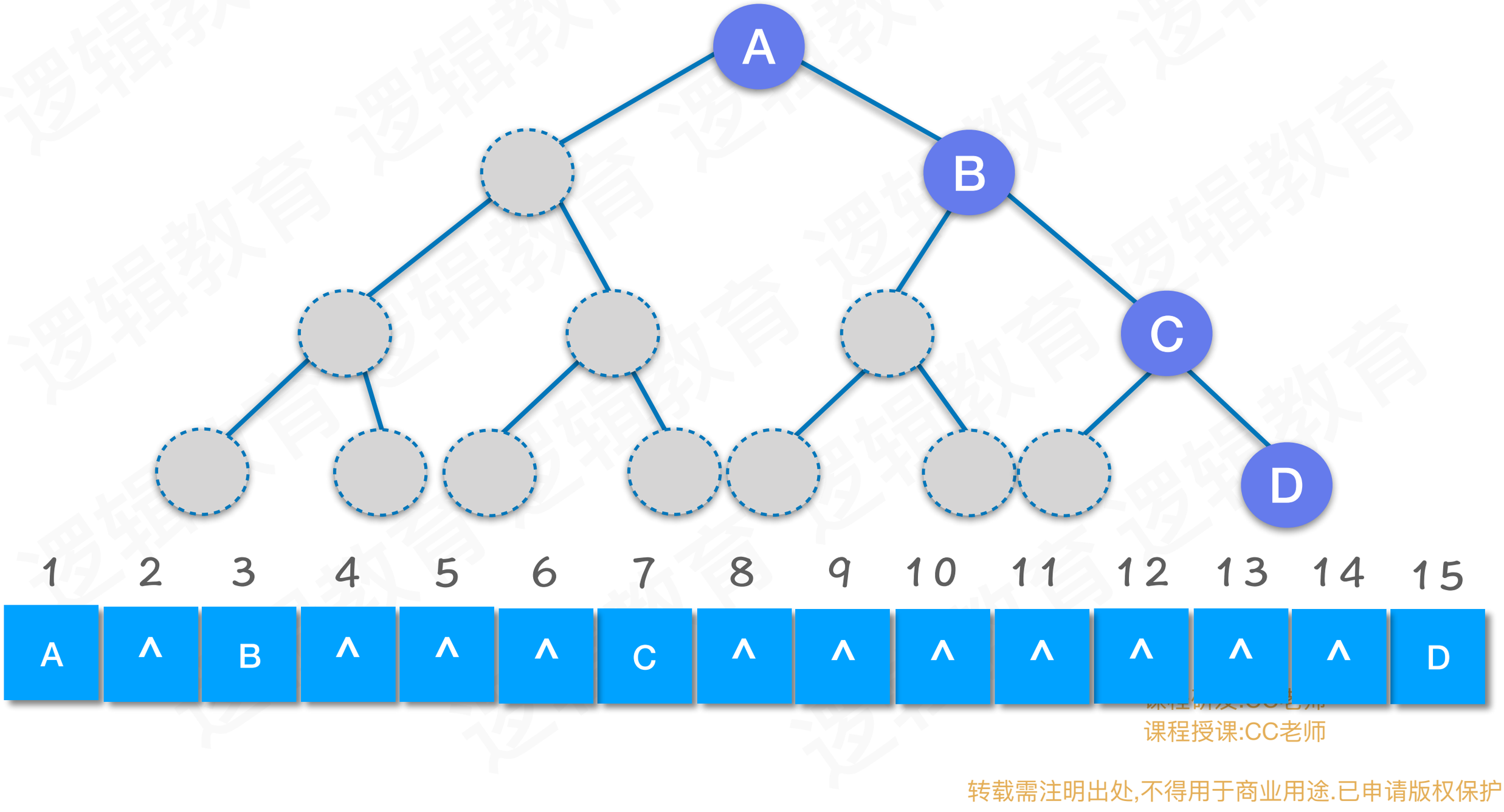
二叉树的顺序存储结构分析



思考: 此二叉树应该如何存储到顺序存储结构中?

课程研发:CC老师
课程授课:CC老师

二叉树的顺序存储结构分析



逻辑教育
Logic education

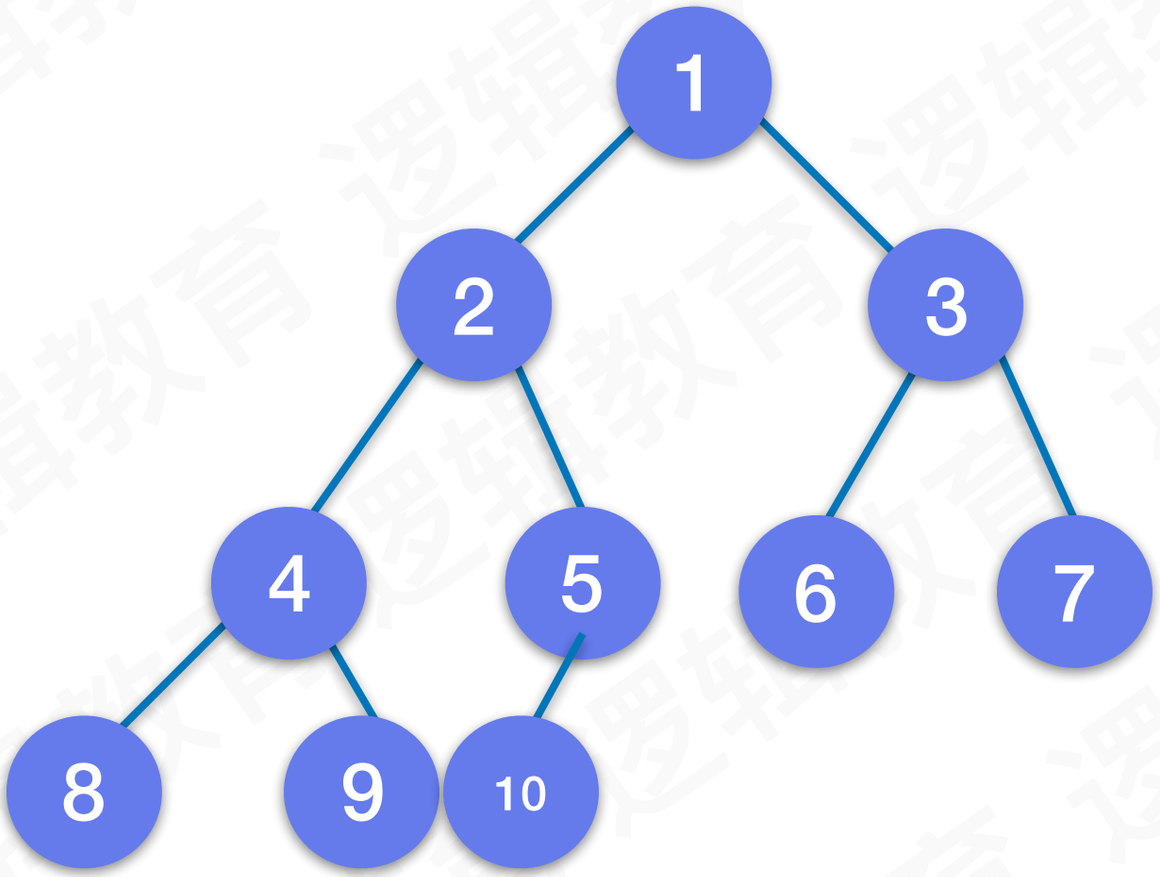
二叉树的实现

6.1 visit
6.2 构造空二叉树T,因为T是固定数组,不会改变.
6.3 按层序次序输入二叉树中的结点值(字符型或整型),构造顺序存储的二叉树T
6.4 判断二叉树是否为空
6.5 获取二叉树的深度
6.6 返回处于位置e(层,本层序号)的结点值
6.7 获取二叉树跟结点的值
6.8 给处于位置e的结点赋值
6.9 获取e的双亲;
6.10 获取某个结点的左孩子;
6.11 获取某个结点的右孩子;
6.12 获取结点的左兄弟
6.13 获取结点的右兄弟

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的实现



课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的遍历

二叉树的遍(Traversing binary tree) 是指的从根结点出发,按照某种次序依次访问二叉树中所有结点,使得每个结点被访问一次且仅被访问一次.

关键词: 访问 和 次序

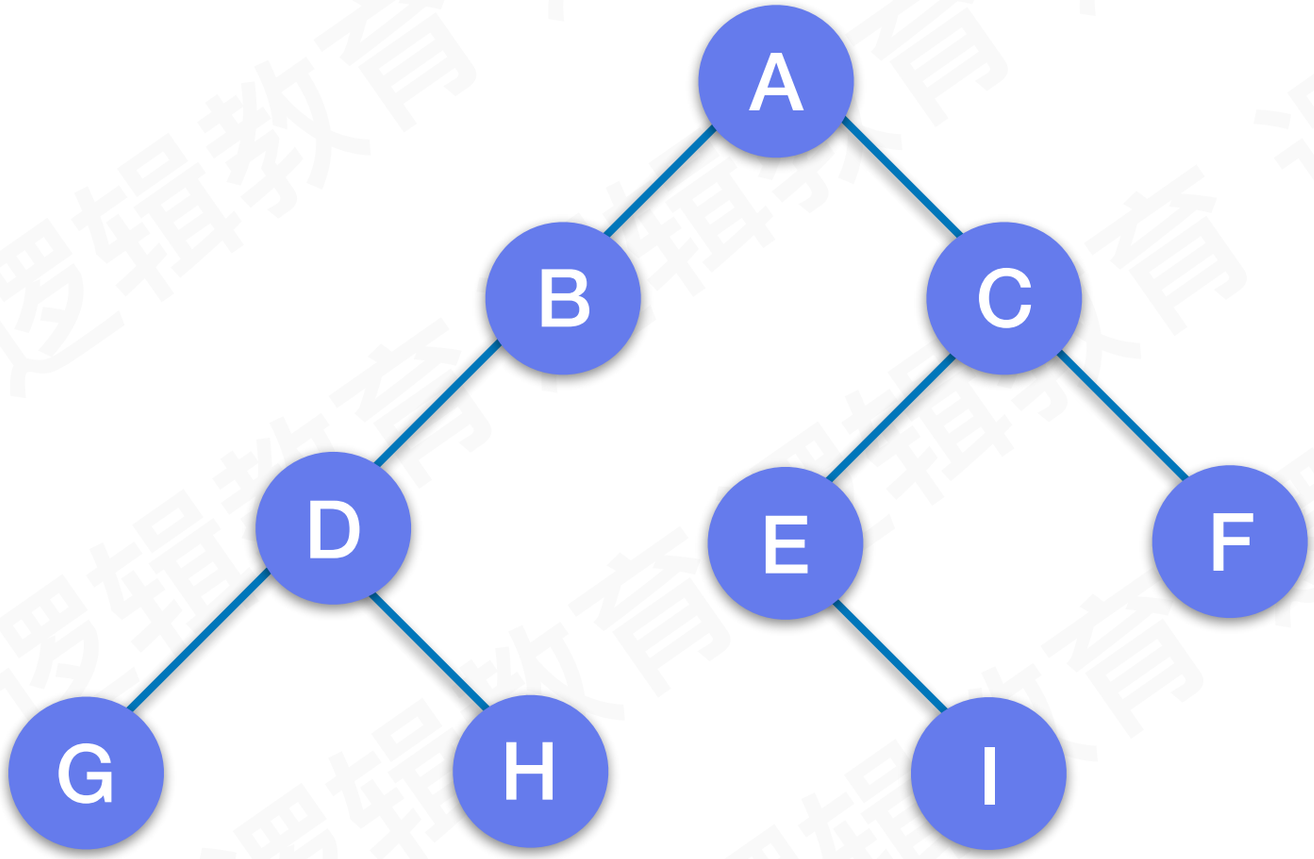
思考:
二叉树的遍历方法有哪些了?
是不是可以像线性表从头到尾进行访问?

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的遍历方法—前序遍历

规则: 若二叉树为空,则空操作返回; 否则先访问根结点,然后前序遍历左子树,在前序遍历右子树



思考:

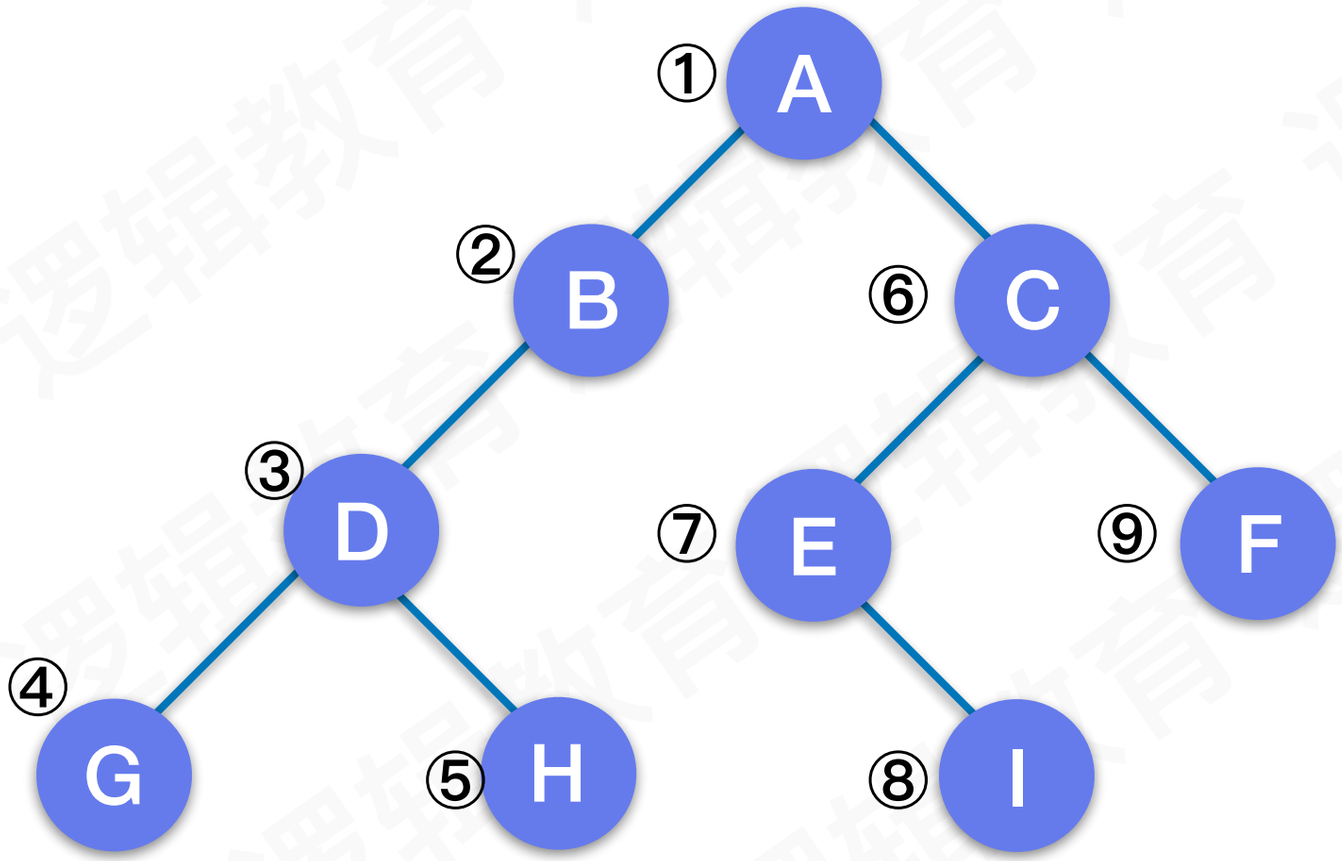
按照前序遍历. 二叉树读取来的数据应该是什么

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的遍历方法—前序遍历

规则: 若二叉树为空,则空操作返回; 否则先访问根结点,然后前序遍历左子树,在前序遍历右子树



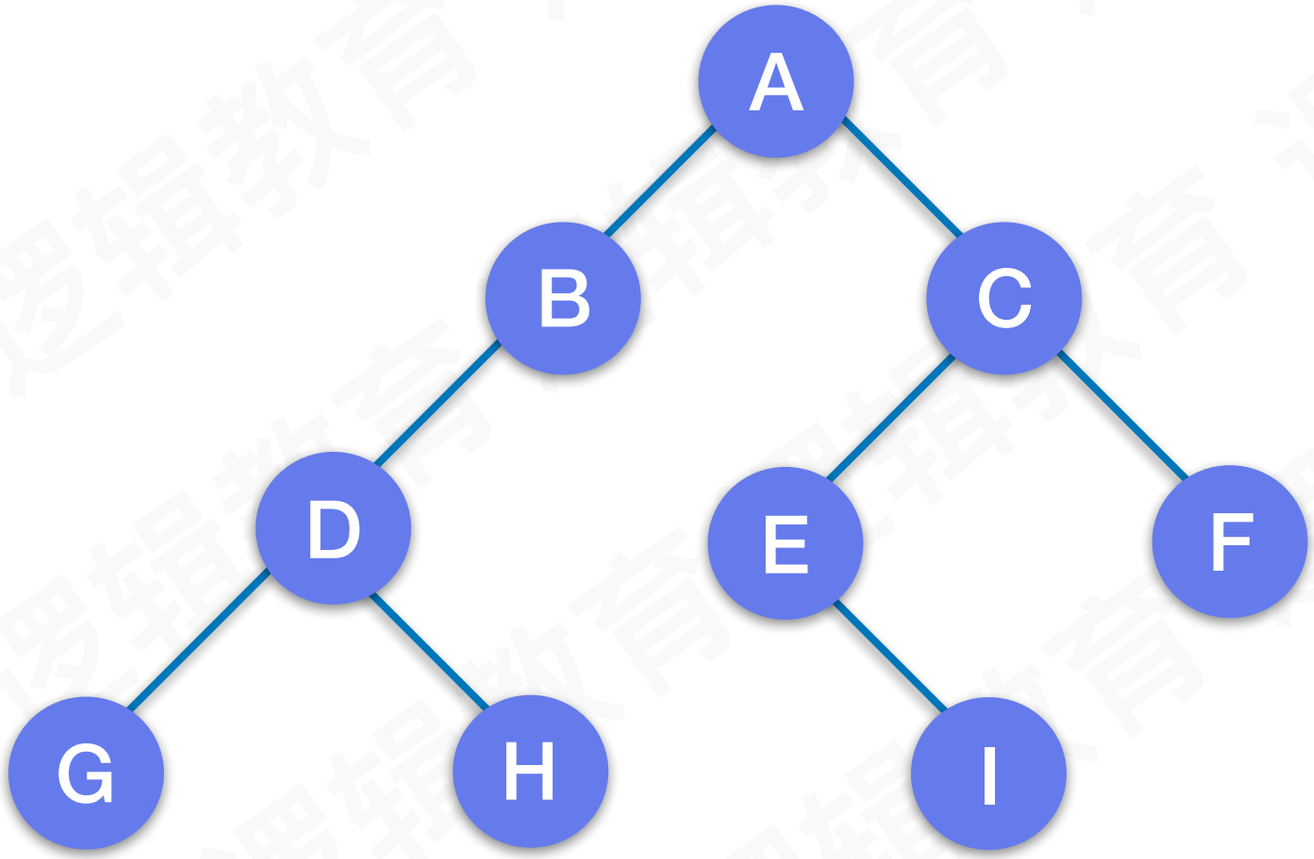
二叉树读取顺序: ABDGHCEIF

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的遍历方法—中序遍历

规则: 若二叉树为空,则空操作返回; 否则从根结点开始(注意并不是先访问根结点),中序遍历根结点的左子树,然后是访问根结点,最后中序遍历右子树.



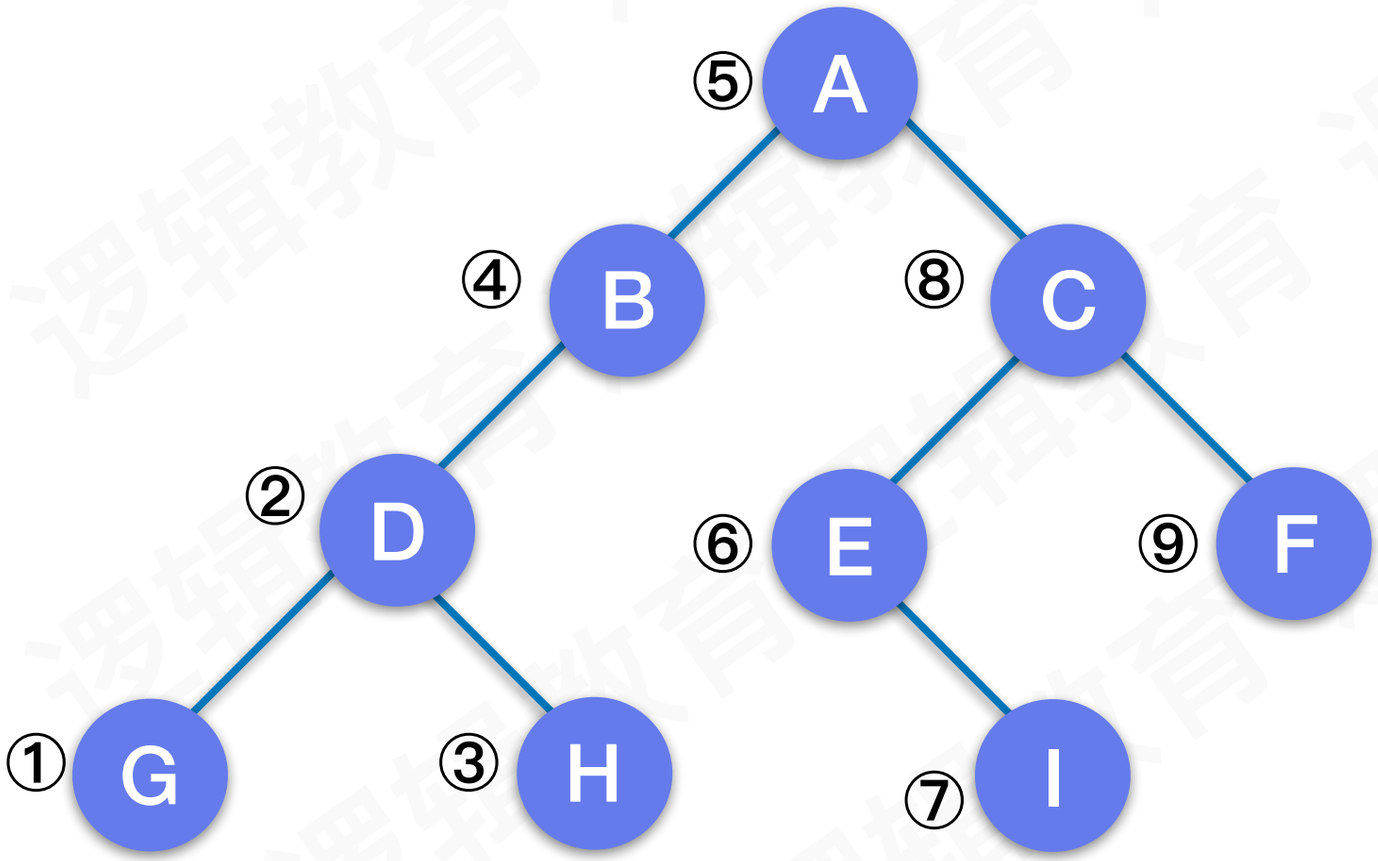
思考:
按照中序遍历. 二叉树读取来的数据应该是什么?

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的遍历方法—中序遍历

规则: 若二叉树为空,则空操作返回; 否则从根结点开始(注意并不是先访问根结点),中序遍历根结点的左子树,然后是访问根结点,最后中序遍历右子树.

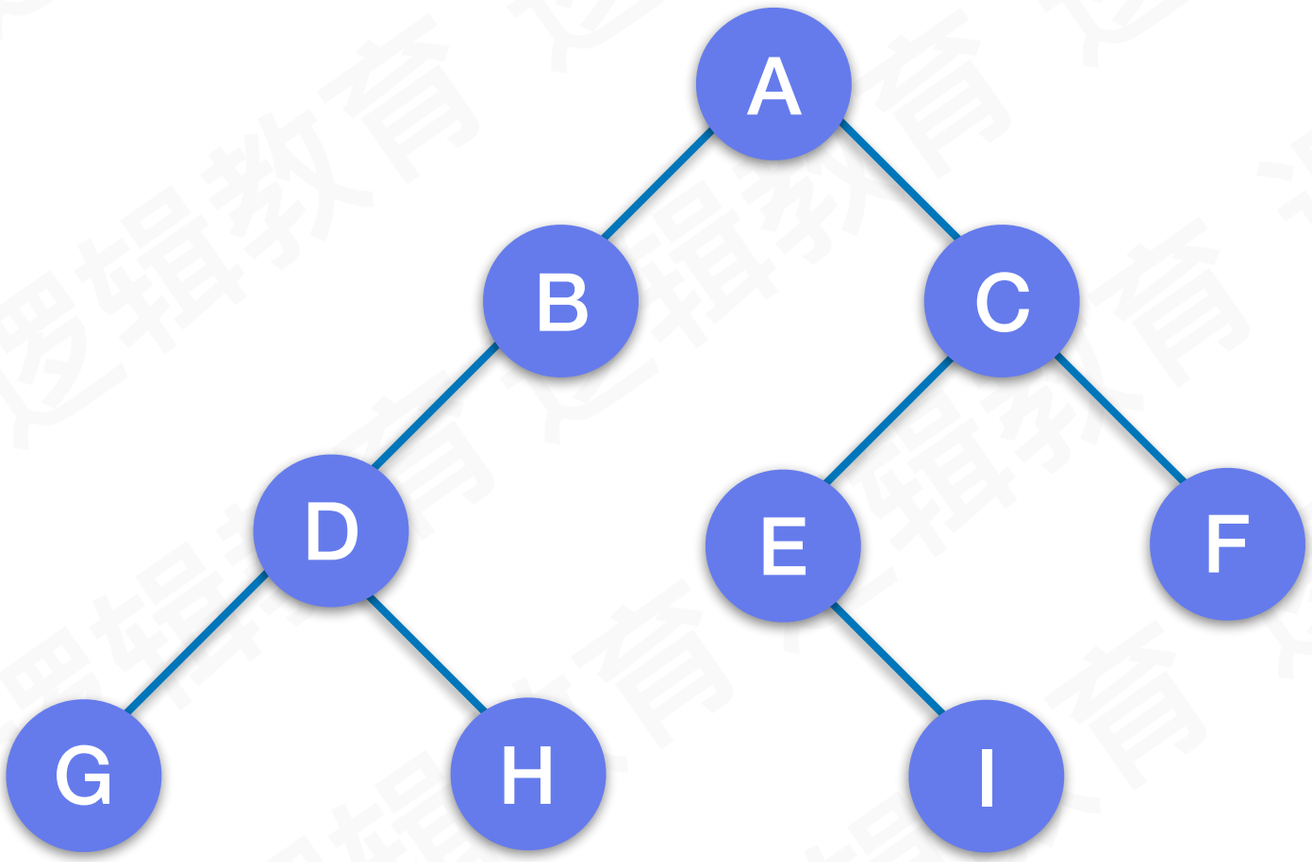


二叉树读取顺序:GDHBAEICF

课程研发:CC老师
课程授课:CC老师

二叉树的遍历方法—后序遍历

规则: 若二叉树为空,则空操作返回; 否则从左到右先叶子后结点的方式遍历左右子树,最后访问根结点



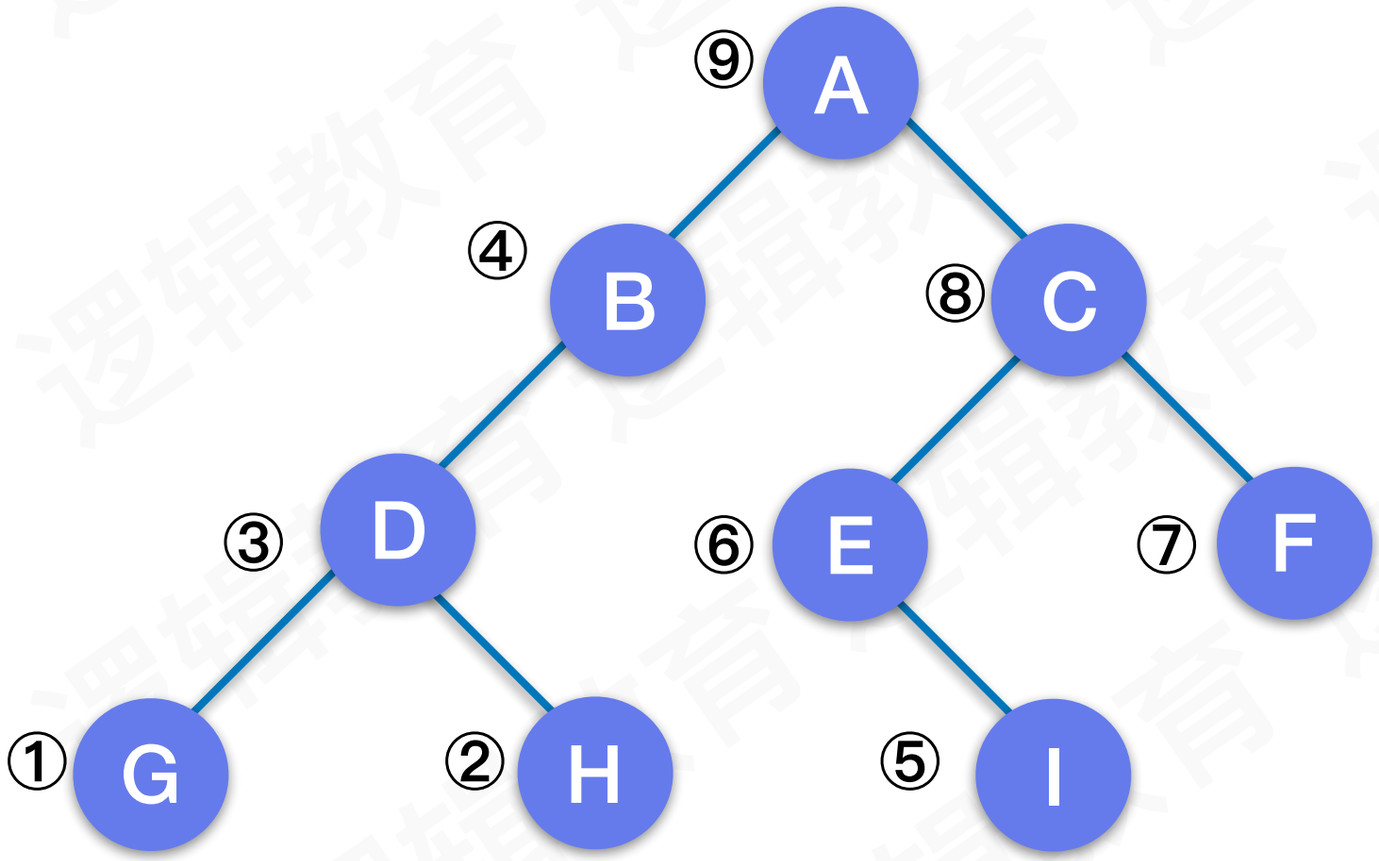
思考:

按照后序遍历. 二叉树读取来的数据应该是什么?

课程研发:CC老师
课程授课:CC老师

二叉树的遍历方法—后序遍历

规则: 若二叉树为空,则空操作返回; 否则从左到右先叶子后结点的方式遍历左右子树,最后访问根结点



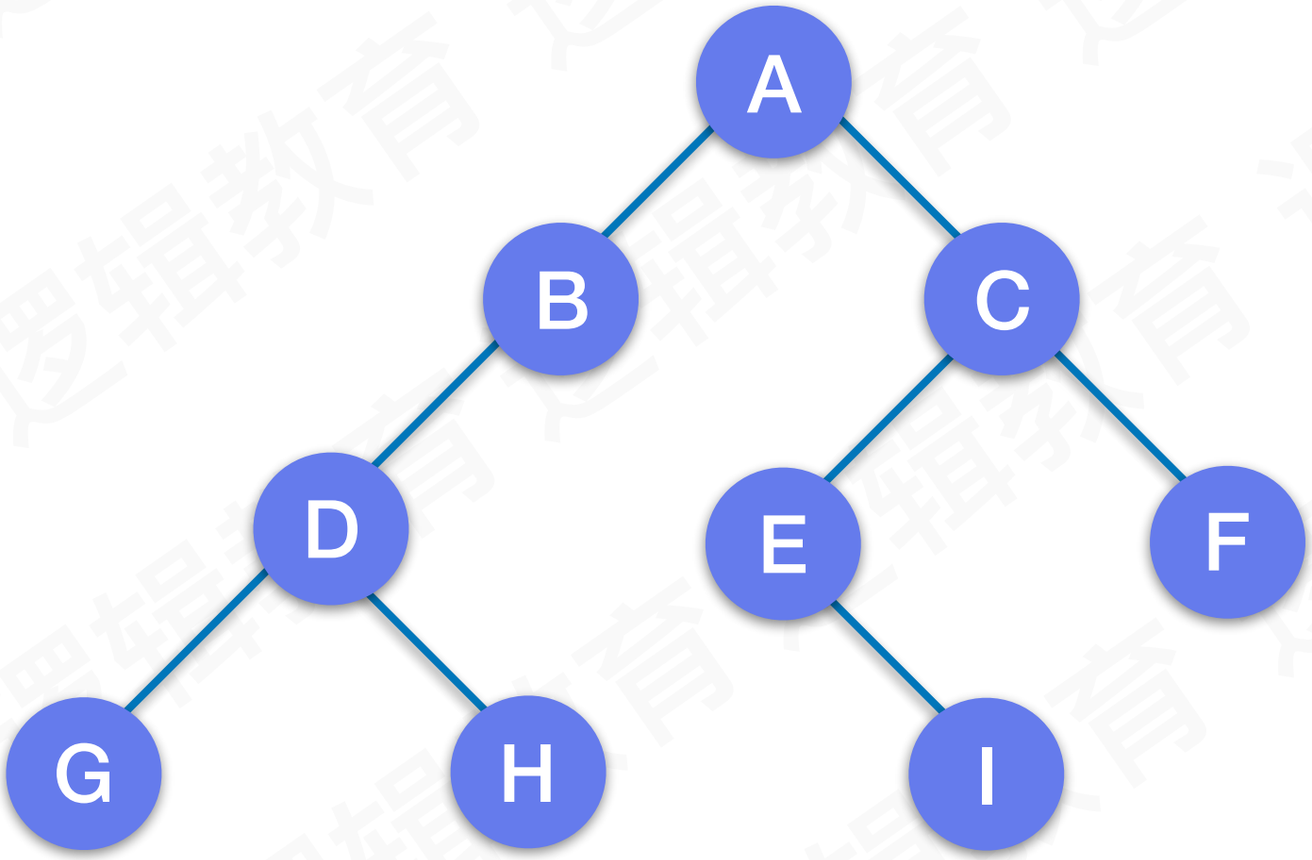
二叉树读取顺序:GHDBIEFCA

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的遍历方法—层序遍历

规则: 若二叉树为空,则空操作返回; 否则从树的第一层,也是就是根结点开始访问,从上而下逐层遍历,在同一层中, 按从左到右的顺序对结点逐个访问.



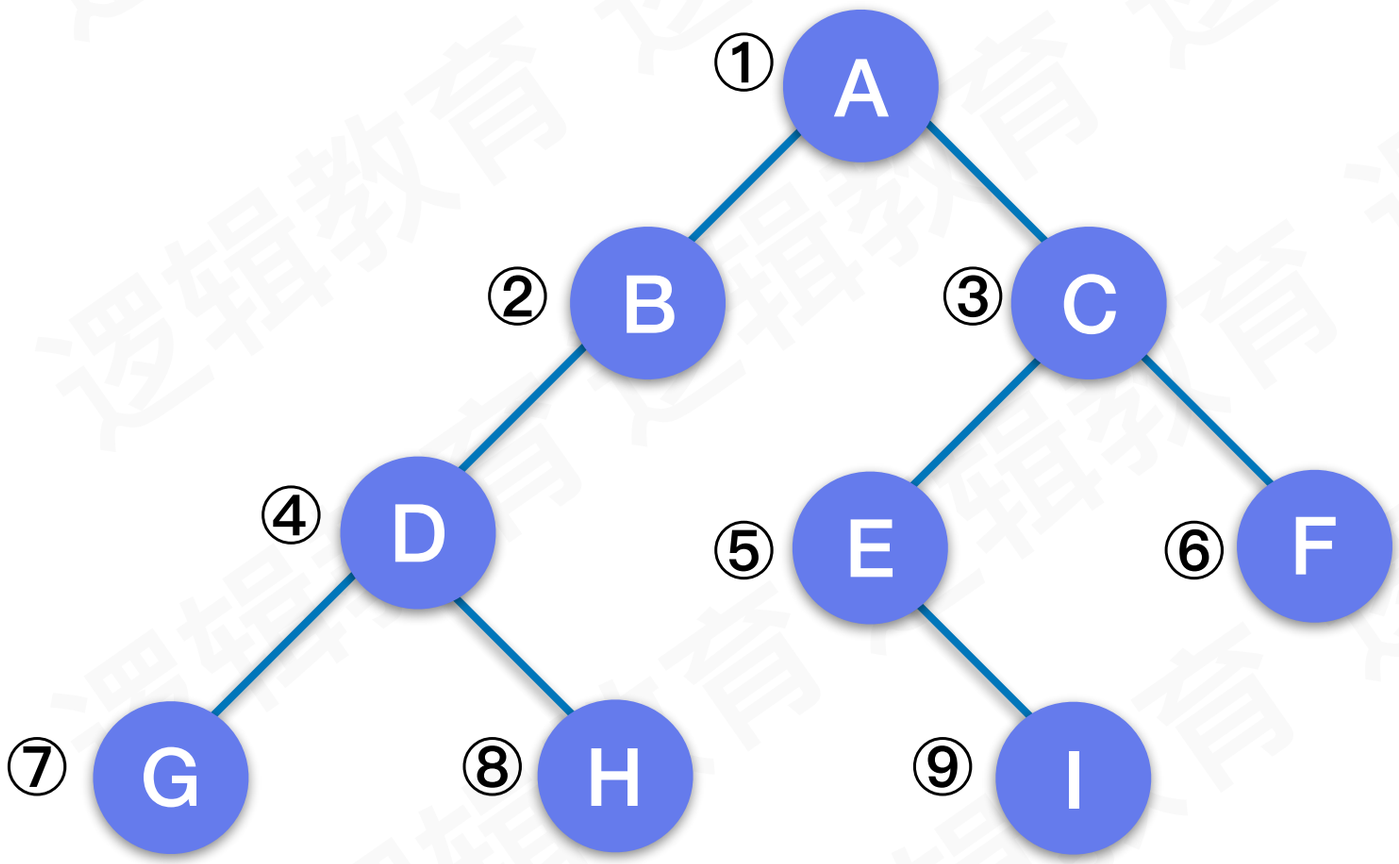
思考:

按照层序遍历. 二叉树读取来的数据应该是什么?

课程研发:CC老师
课程授课:CC老师

二叉树的遍历方法—层序遍历

规则: 若二叉树为空,则空操作返回; 否则从树的第一层,也是就是根结点开始访问,从上而下逐层遍历,在同一层中, 按从左到右的顺序对结点逐个访问.



二叉树读取顺序:ABCDEFGHI

课程研发:CC老师
课程授课:CC老师

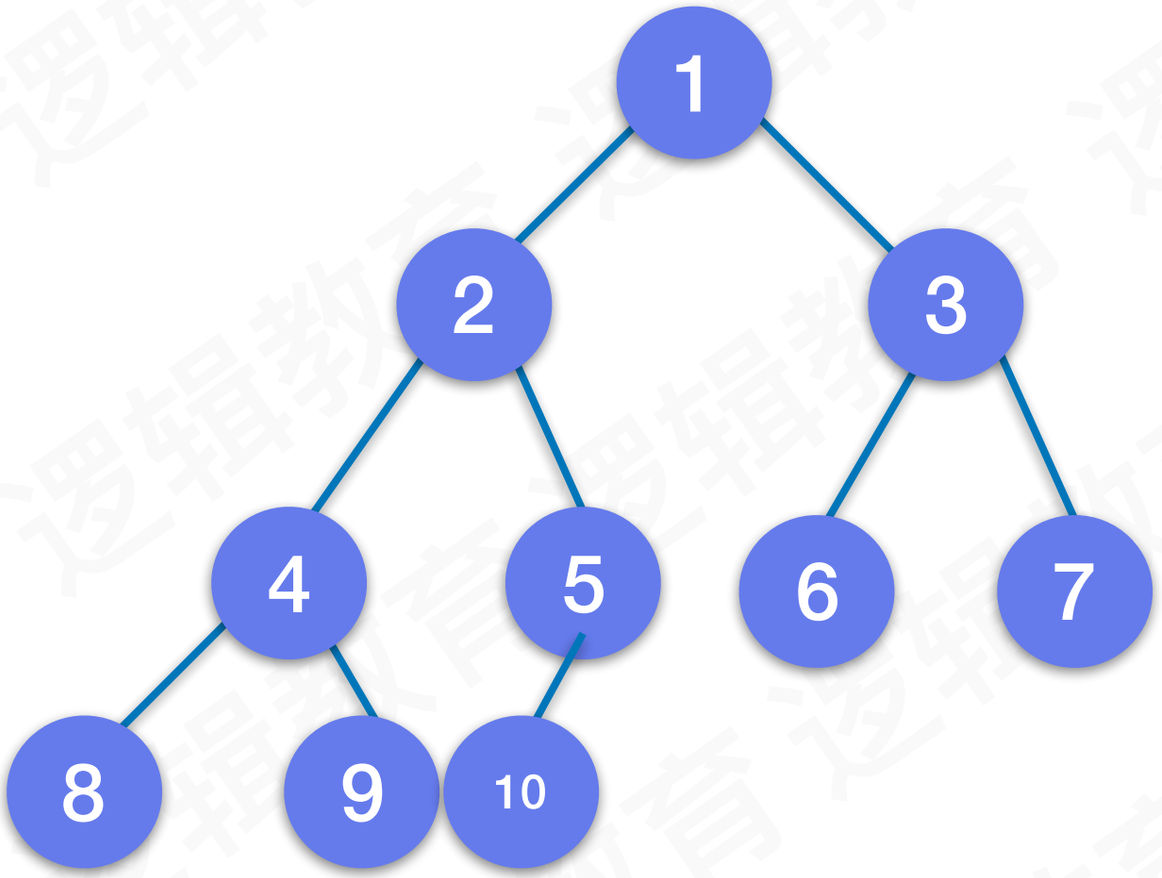
逻辑教育
Logic education

二叉树顺序存储结构遍历实现

6.14 层序遍历二叉树
6.15 前序遍历二叉树
6.16 中序遍历
6.17 后序遍历

课程研发:CC老师
课程授课:CC老师

二叉树的遍历方法—遍历



思考:

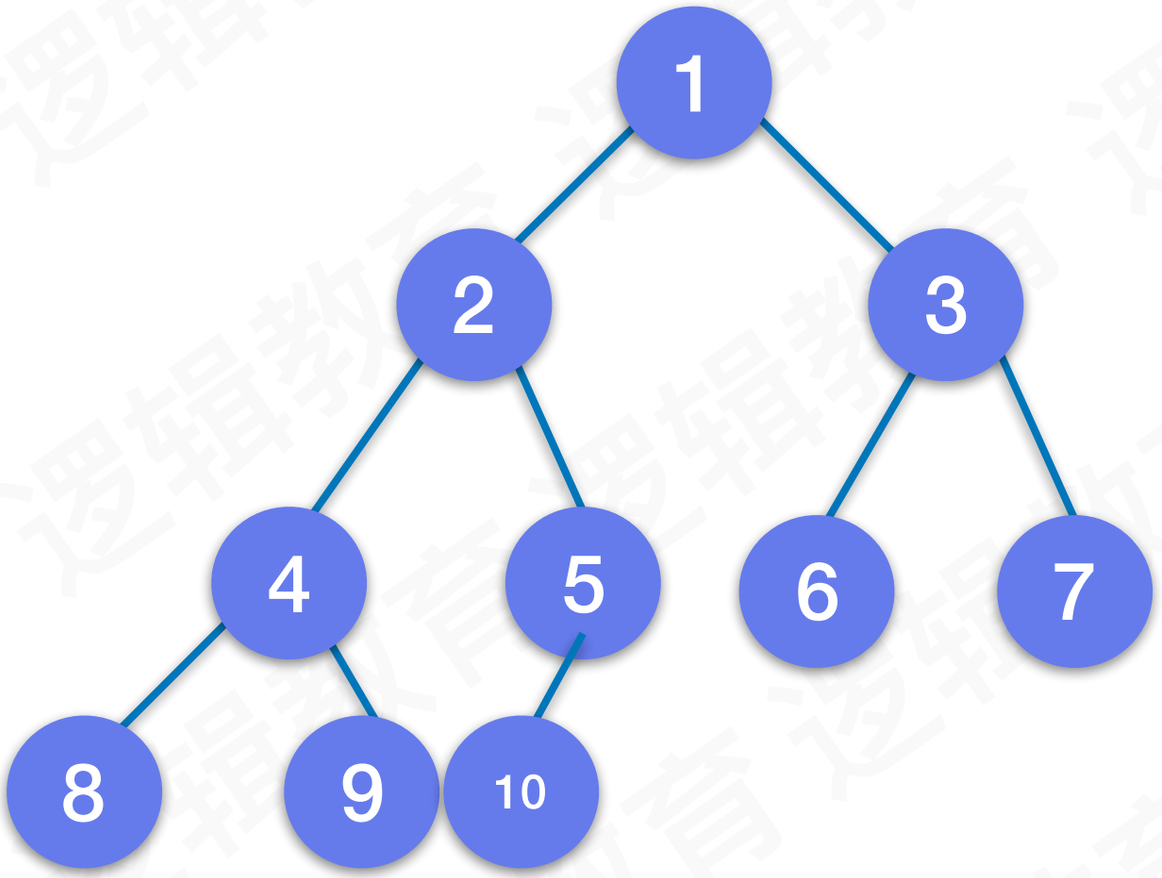
请问该二叉树按照层序, 前序,中序,后续遍历后输出的顺序是?

前序:

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树的遍历方法—遍历



二叉树层序遍历: 1 2 3 4 5 6 7 8 9 10

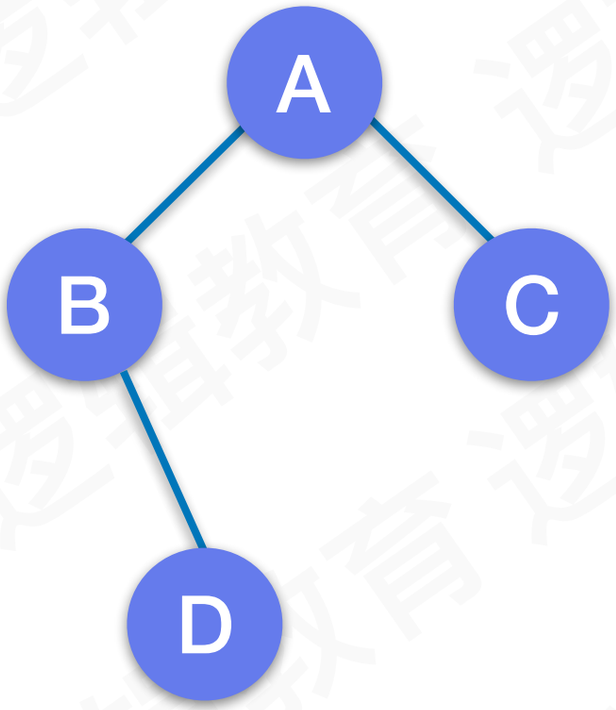
二叉树先序遍历: 1 2 4 8 9 5 10 3 6 7

二叉树中序遍历: 8 4 9 2 10 5 1 6 3 7

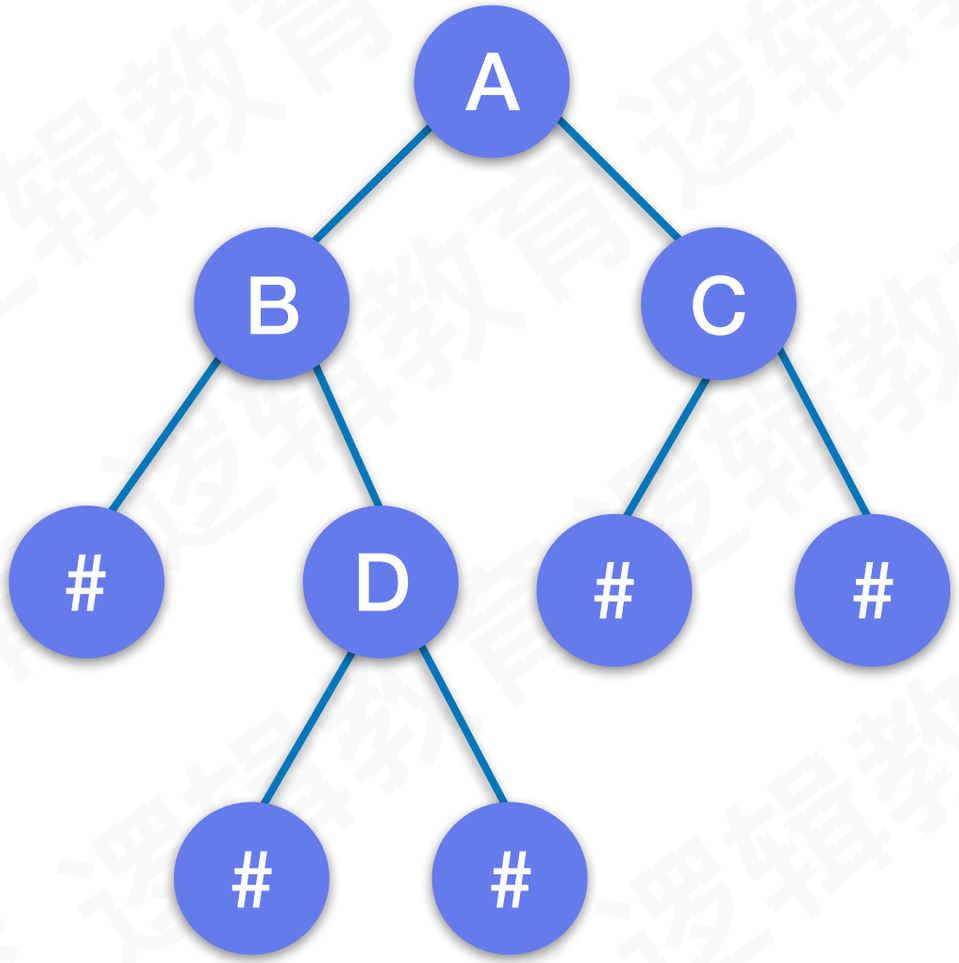
二叉树后序遍历: 8 9 4 10 5 2 6 7 3 1

课程研发:CC老师
课程授课:CC老师

二叉树的遍历方法—遍历



普通二叉树



扩展二叉树

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

二叉树链式存储基本操作实现

7.1 打印数据
7.2 构造空二叉树T
7.3 销毁二叉树
7.4 创建二叉树
7.5 二叉树T是否为空;
7.6 二叉树T的深度
7.7 二叉树T的根
7.8 返回p所指向的结点值
7.8 给p所指结点赋值为value

课程研发:CC老师
课程授课:CC老师

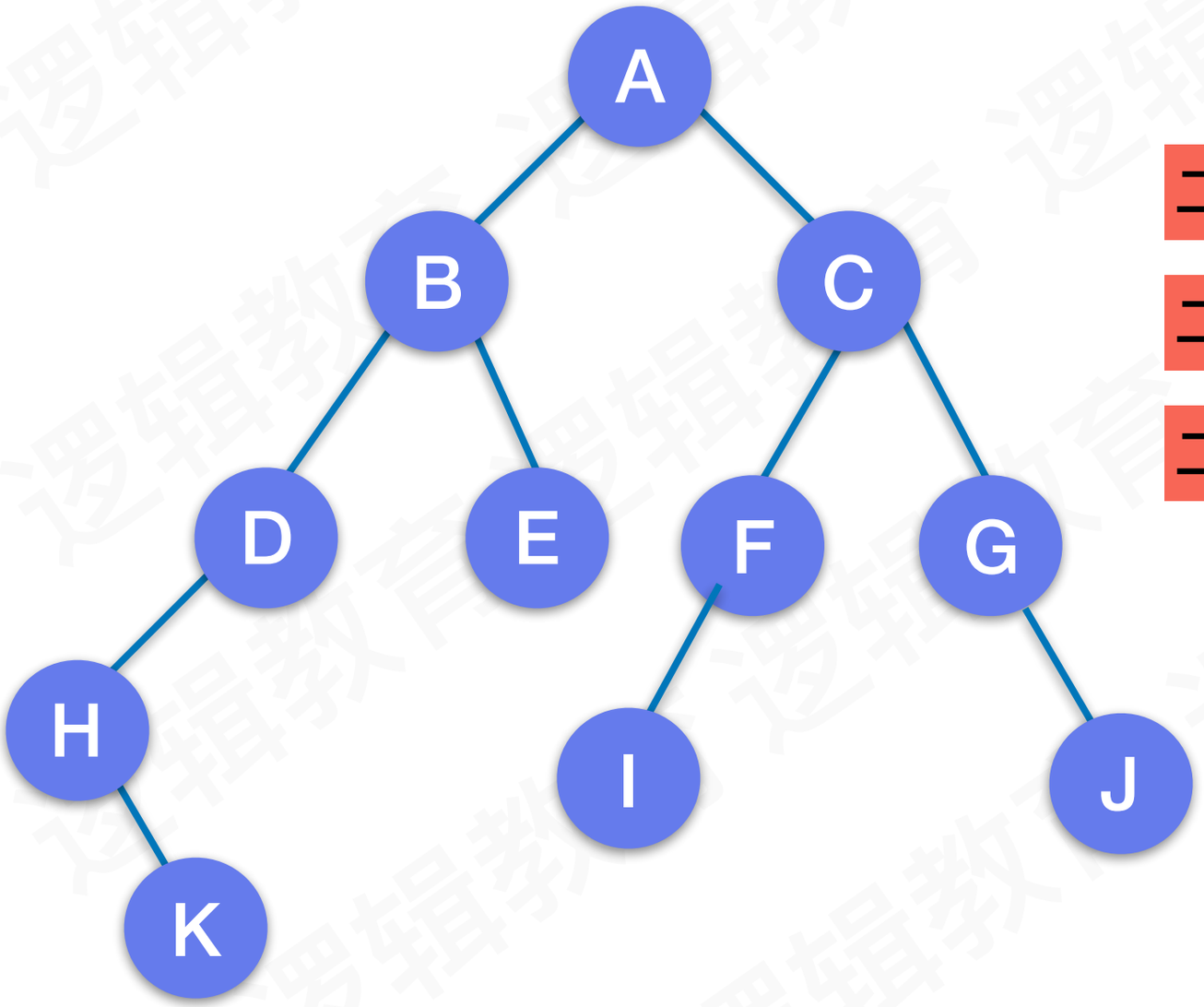
逻辑教育
Logic education

二叉树链式存储遍历实现

7.8 前序递归遍历T
7.9 中序递归遍历T
7.10 后序递归遍历T

课程研发:CC老师
课程授课:CC老师

二叉树的遍历方法—遍历



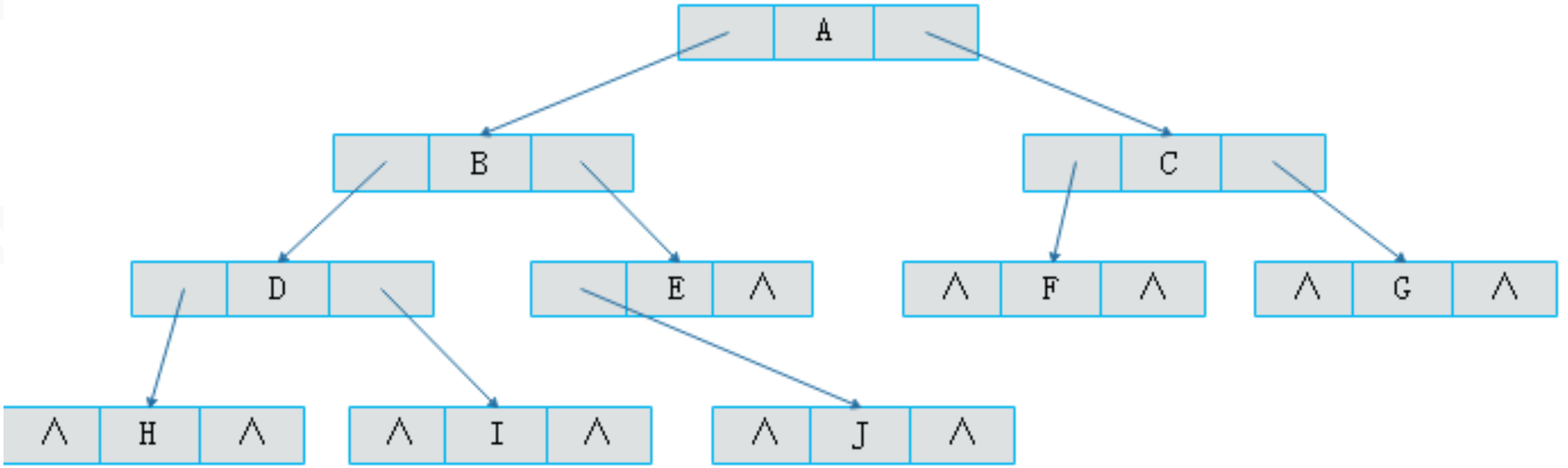
二叉树先序遍历: A B D H K E C F I G J

二叉树中序遍历: H K D B E A I F C G J

二叉树后序遍历: K H D E B I F J G C A

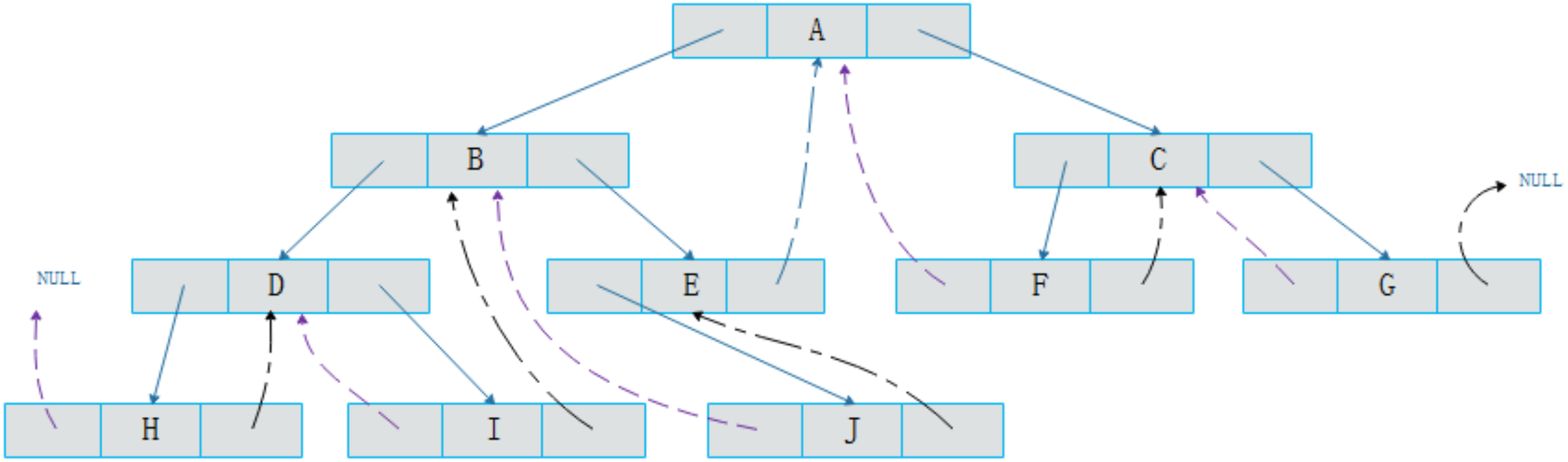
课程研发:CC老师
课程授课:CC老师

线索二叉树—产生背景



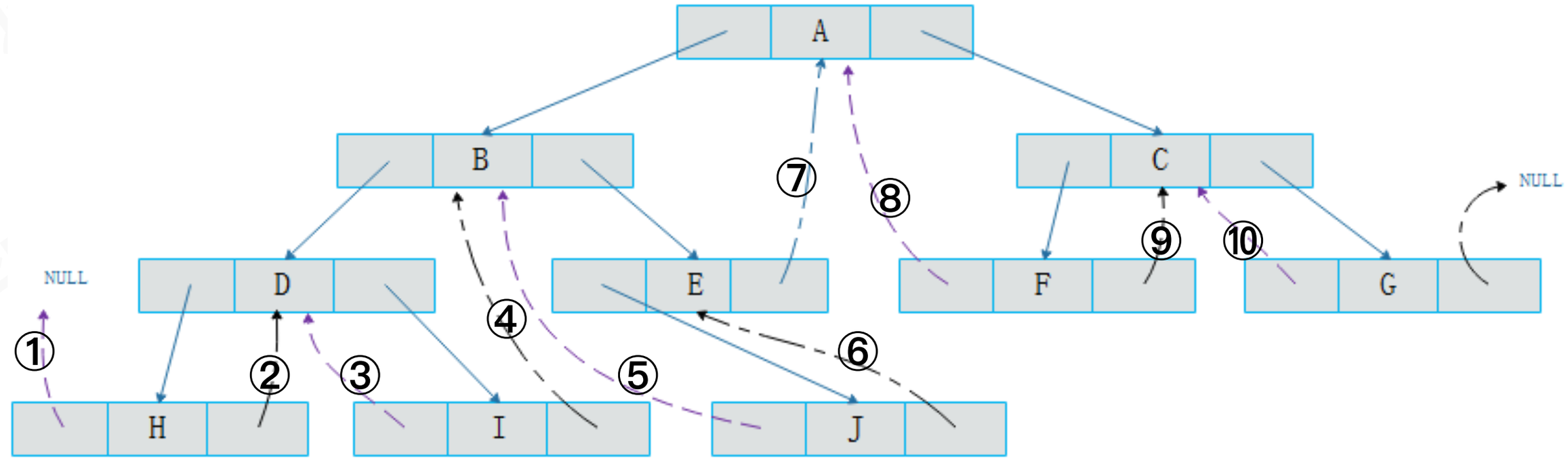
课程研发:CC老师
课程授课:CC老师

线索二叉树—线索二叉树



课程研发:CC老师
课程授课:CC老师

线索二叉树—线索二叉树

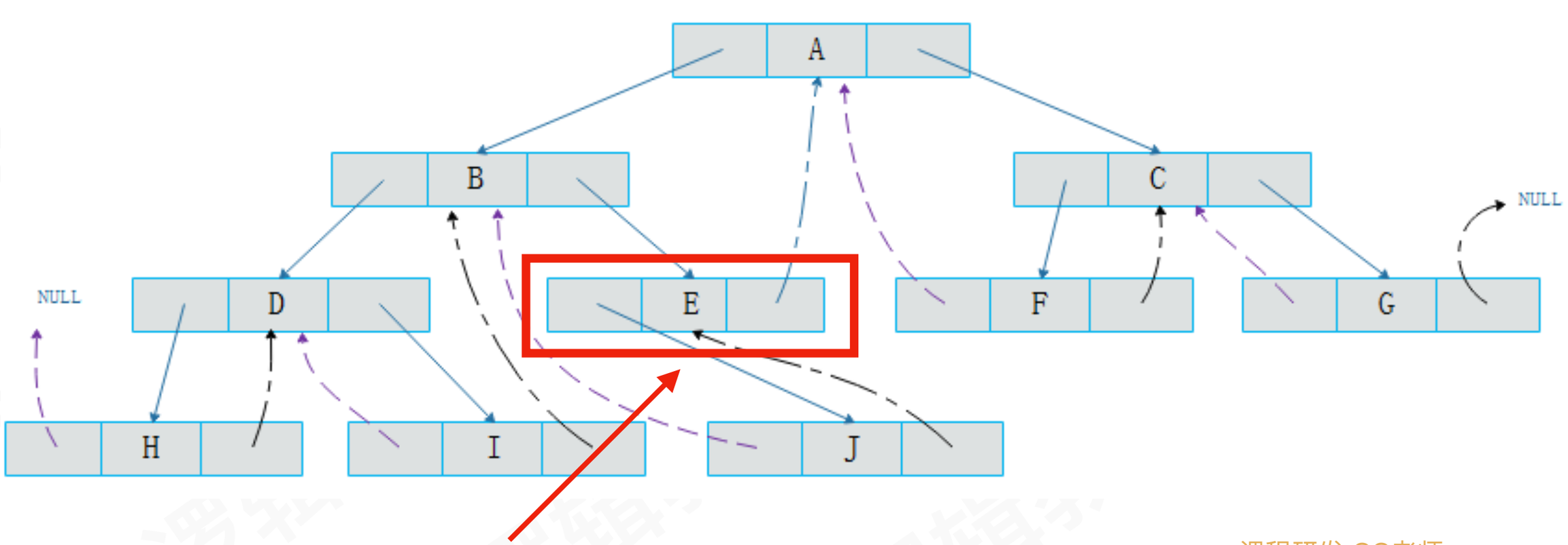


中序遍历: HDIBJEAFCG

课程研发:CC老师
课程授课:CC老师

线索二叉树—线索化带来的问题

思考: 我们如何区分一个结点的左孩子指针指向的是左孩子还是前驱结点了



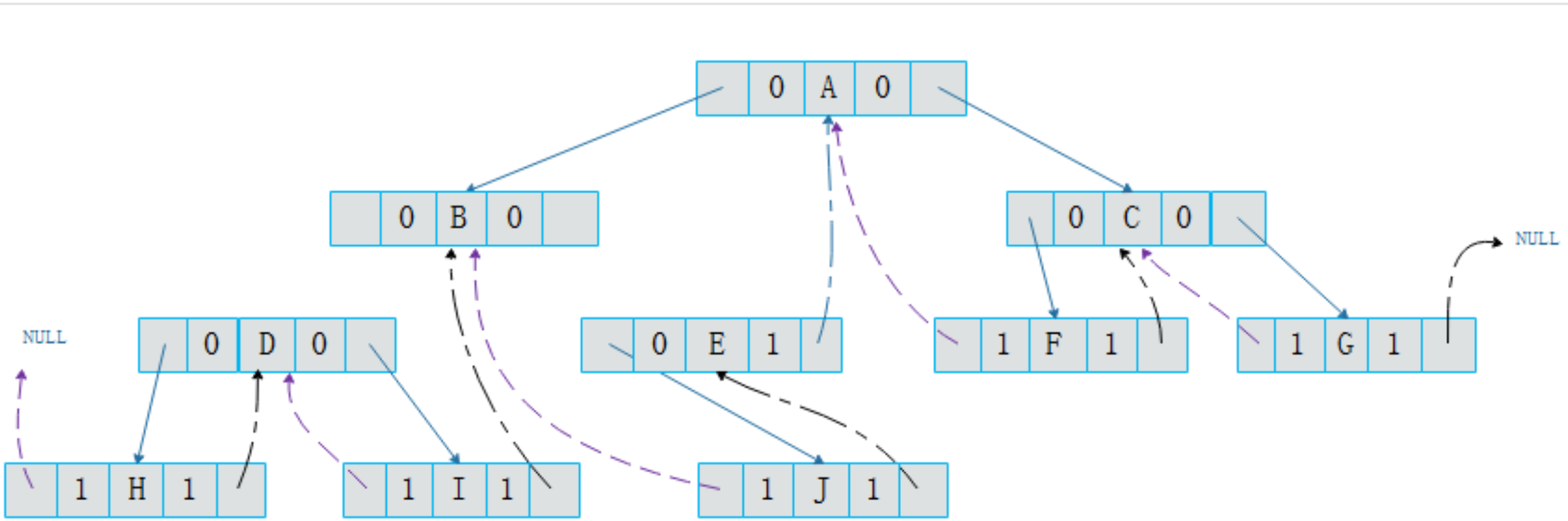
课程研发:CC老师
课程授课:CC老师

线索二叉树—线索化带来标识位

lchild	ltag	data	rtag	rchild

课程研发:CC老师
课程授课:CC老师

线索二叉树—带有标识位的线索二叉树



课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

线索二叉树—代码解读

代码解读: 除了加粗部分,你会发现和二叉树中序遍历递归代码几乎完全一样

```
void InThreading(BiThrTree p)
{
    if(p)
    {
        InThreading(p->lchild); /* 递归左子树线索化 */
        if(!p->lchild) /* 没有左孩子 */
        {
            p->LTag=Thread; /* 前驱线索 */
            p->lchild=pre; /* 左孩子指针指向前驱 */
        }
        if(!pre->rchild) /* 前驱没有右孩子 */
        {
            pre->RTag=Thread; /* 后继线索 */
            pre->rchild=p; /* 前驱右孩子指针指向后继(当前结点p) */
        }
        pre=p; /* 保持pre指向p的前驱 */
        InThreading(p->rchild); /* 递归右子树线索化 */
    }
}
```

解读: 表示如果某结点的左指针域为空,因为其前驱结点刚刚访问过,赋值给pre. 所以可以将pre 赋值给p->lchild. 并修改p->LTag = Thread 完成线上结点的线索化

解读: 如果该结点的前驱没有右孩子,则表示p就是pre的后继.

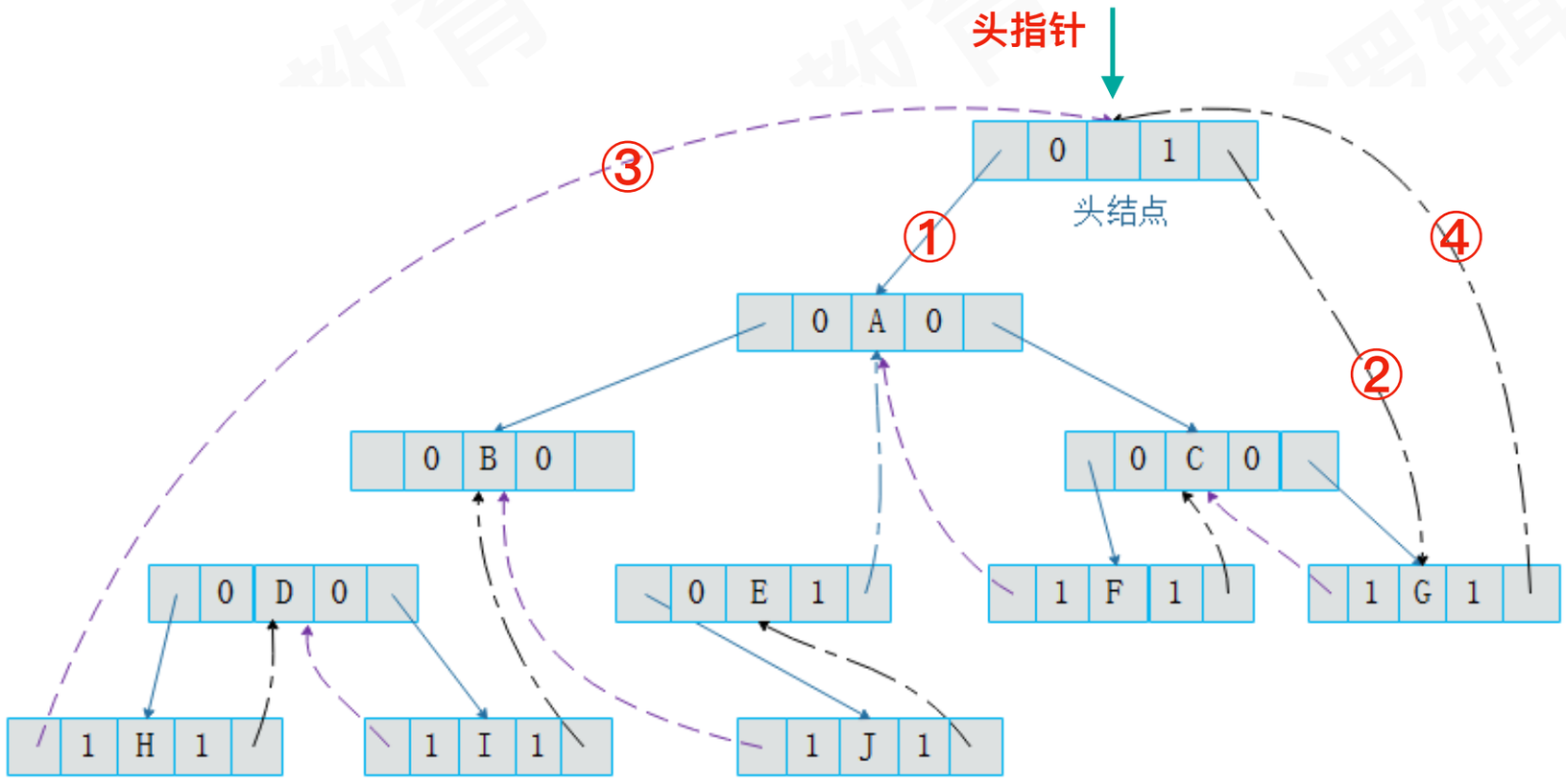
所以此时pre->RTag = Thread 完成线索化; pre->rchild pre的后继就是p

解读: 完成,前驱和后继的判断; 将当前的 p 赋值给pre

课程研发:CC老师
课程授课:CC老师

线索二叉树—双向链表结构

线索化的二叉树,在进行遍历时, 其实就等价于操作一个双向链表结构; 因为类似双向链表结构,所以我们在二叉树线索链表上添加一个头结点.



① 将头结点的lchild 域指针指向二叉树的根结点

② 将头结点的rchild 域指针指向中心遍历时最后一个结点

③ 使二叉树中序序列中的第一个结点中,lchild 域指针和最后一个结点的rchild 域指针均指向头结点

这样做的好处,你可以从第一个结点起顺后继进行遍历,也可以从最后一个结点起顺前驱进行遍历

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

线索二叉树—中序遍历

```
/*中序遍历二叉线索树T*/
Status InOrderTraverse_Thr(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;
        if(!visit(p->data)) /* 访问其左子树为空的结点 */
            return ERROR;
        while(p->RTag==Thread&&p->rchild!=T)
        {
            p=p->rchild;
            visit(p->data); /* 访问后继结点 */
        }
        p=p->rchild;
    }
    return OK;
}
```

```
while(p->RTag==Thread&&p->rchild!=T)
    {
        p=p->rchild;
        visit(p->data); /* 访问后继结点 */
    }
    p=p->rchild;
```

解读: while(p->RTag==Thread&&p->rchild!=T) 由于结点H的RTag == Tread,且不是指向头结点, 因此打印H的后继D,之后因为D的RTag是Link 则退出循环.

解读: 对应上图①的过程, 让指针p指向根结点开始遍历

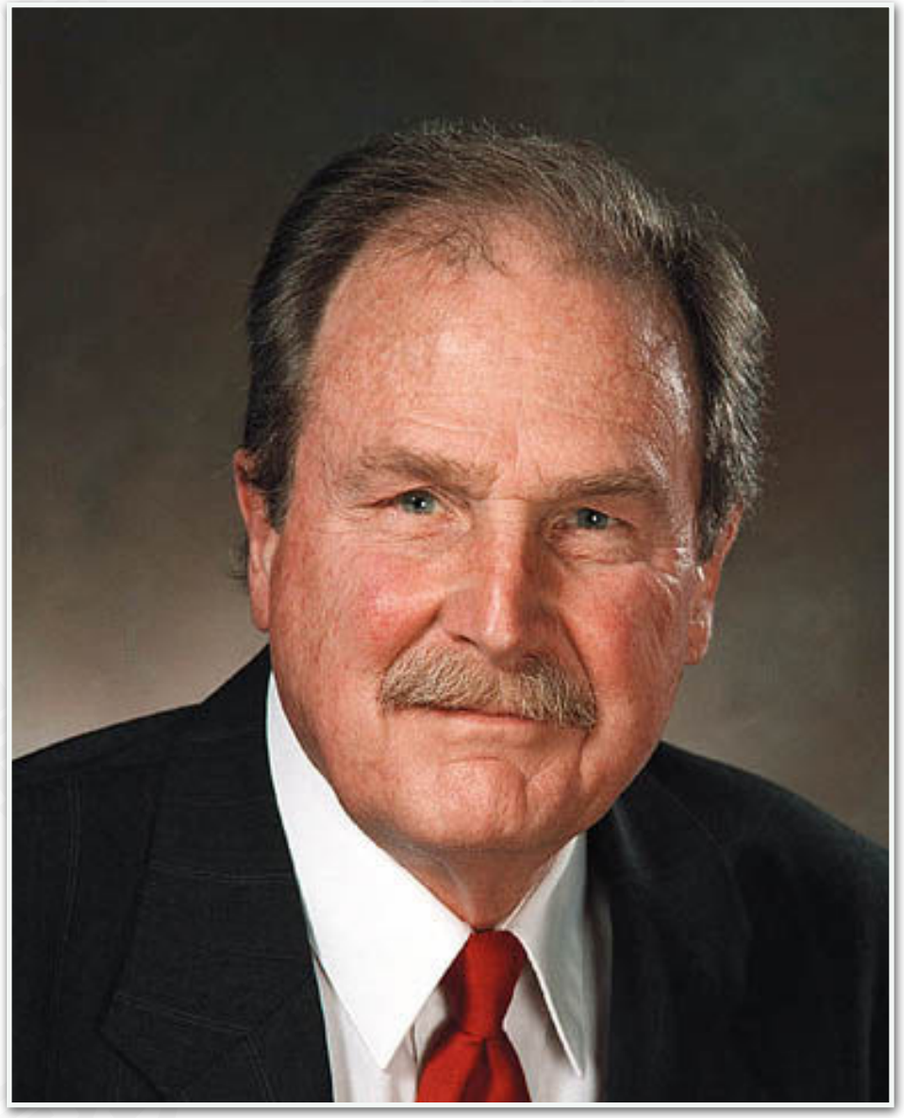
解读:循环直到图④的出现,意味着p指向了头结点,与T相等.(T是指向头结点的指针),结束循环,否则就一直循环下去

```
while(p->LTag==Link)
        p=p->lchild;
```

解读: while(p->LTag == Link) 这个循环, 就是由A->B->D->H, 此时H结点的LTag不是Link ,则循环结束

逻辑教育
Logic education

戴维.哈夫曼(David Huffman)



著名的“霍夫曼编码”的发明人戴维·霍夫曼 (David Albert Huffman)已于1999年10月17日因癌症去世，享年74岁.
他发明了著名的“霍夫曼编码”

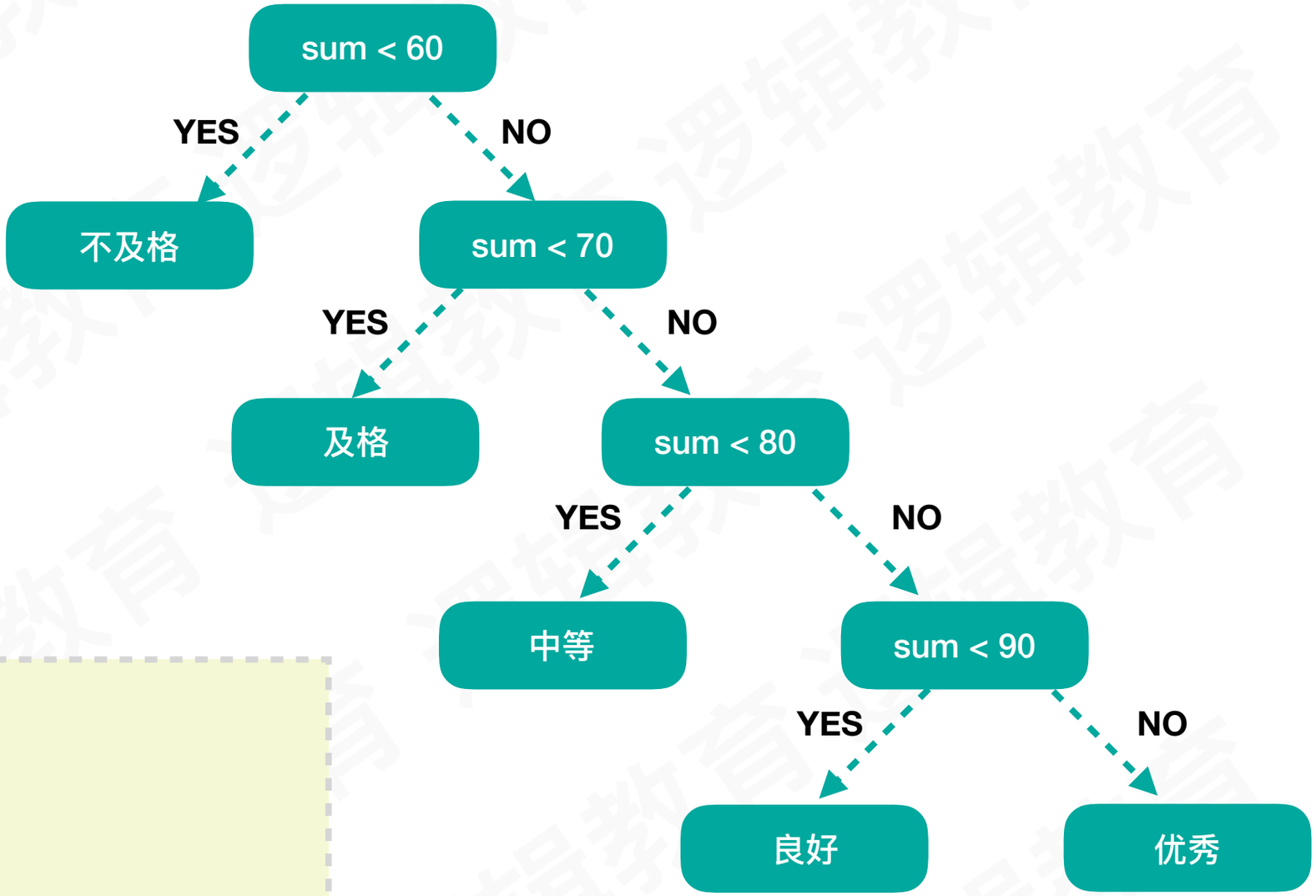
除了霍夫曼编码外,霍夫曼在其他方面也还有不少创造,比如他设计的二叉最优搜索树算法就被认为是同类算法中效率最高的,因而被命名为霍夫曼算法,是动态规划(dynamic programming)的一个范例。霍夫曼在MIT一直工作到1967年。之后他转入加州大学的Santa Cruz分校,是该校计算机科学系的创始人,1970—1973年任系主任。1994年霍夫曼退休。

霍夫曼除了获得计算机先驱奖以外,还在1973年获得IEEE的McDowell奖。1998年IEEE下属的信息论分会为纪念信息论创立50周年,授予他Golden Jubilee奖。霍夫曼去世前不久的1999年6月,他还荣获以哈明码发明人命名的哈明奖章(Hamming Medal)。

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

哈夫曼思考



代码片段:

```
if( sum < 60)
   result = “不及格”;
else if(sum < 70)
   result = “及格”;
else if(sum < 80)
   result = “中等”;
else if(sum < 90)
   result = “良好”;
else
   result = “优秀”;
```

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

哈夫曼思考

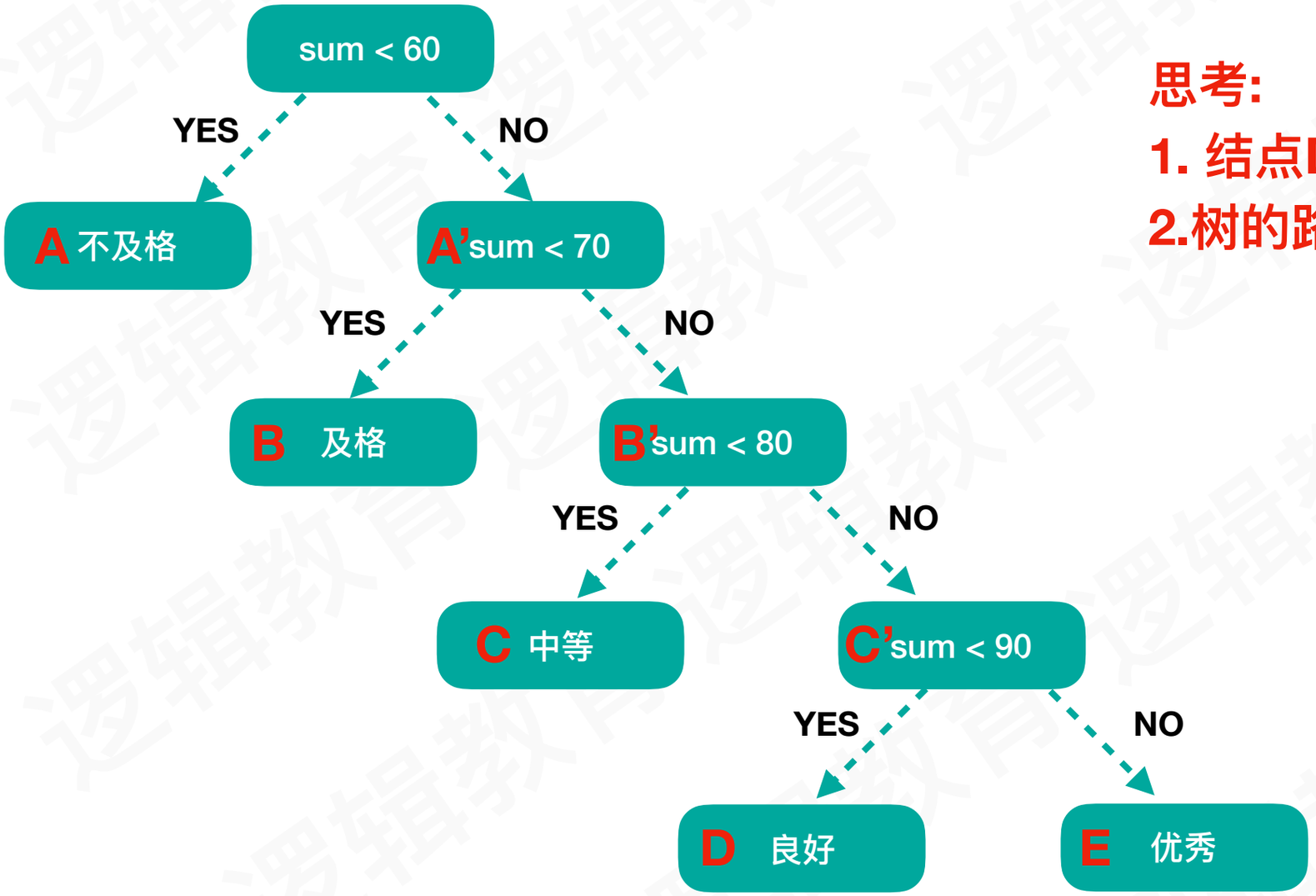
8.1 学生成绩分布规律

分数	0-59	60-69	70-79	80-89	90-100
所占比例	5%	15%	40%	30%	10%

成绩比重: 在70~89分之间占用了70% 但是都是需要经过3次判断才能得到正确的结果. 那么如果数量集非常大时,这样的比较就会出现效率问题.

课程研发:CC老师
课程授课:CC老师

哈夫曼思考

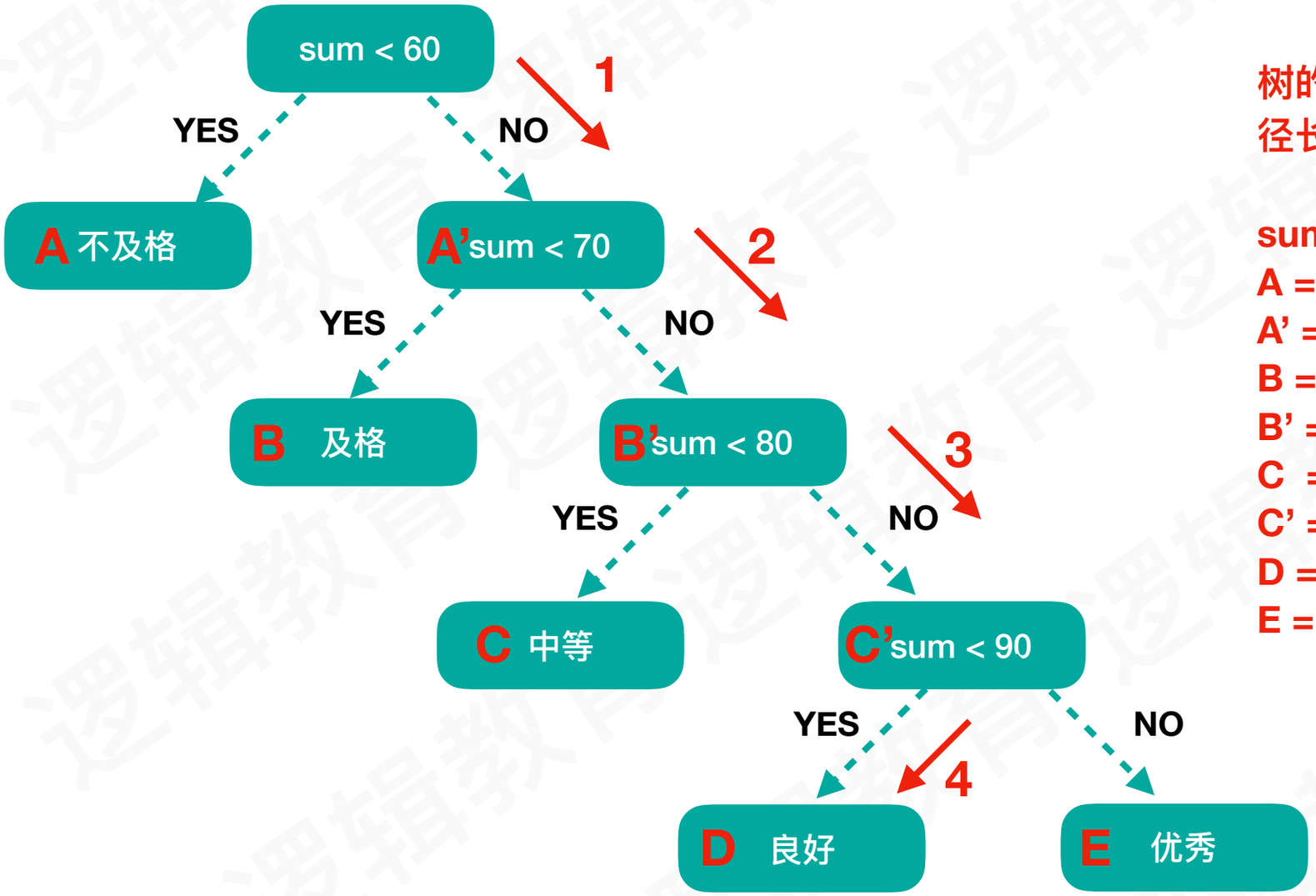


思考:

1. 结点D 的路径长度是?
2.树的路径长度?

课程研发:CC老师
课程授课:CC老师

哈夫曼思考



路径长度: 例如结点D 的路程长度为?

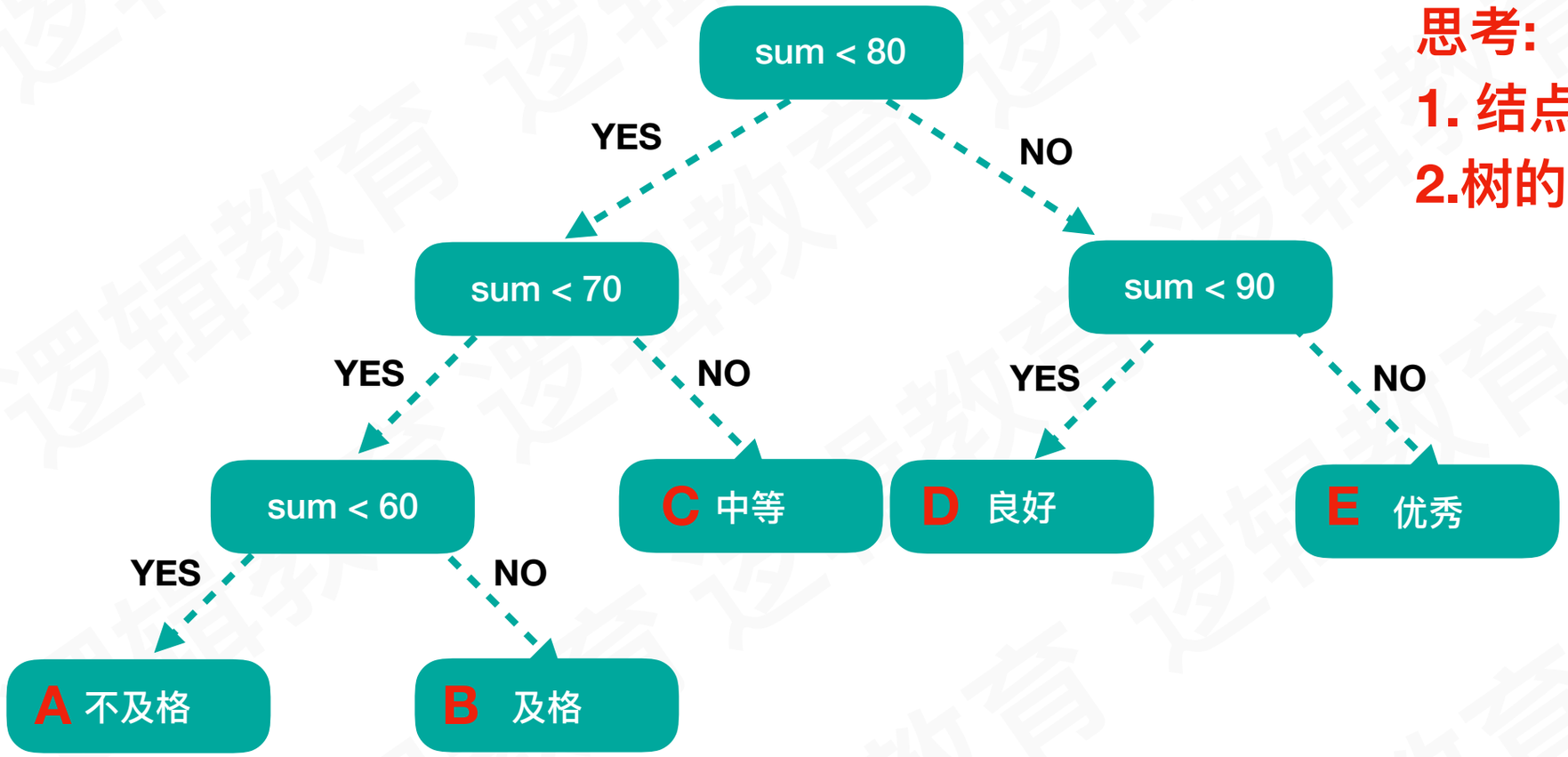
D = 4

树的路径长度就是从树根到每一个结点的路径长度之和

sum = 1+1+2+2+3+3+4+4 = 20
A = 1
A' = 1
B = 2
B' = 2
C = 3
C' = 3
D = 4
E = 4

课程研发:CC老师
课程授课:CC老师

哈夫曼思考



思考:

1. 结点D 的路径长度是?
2.树的路径长度?

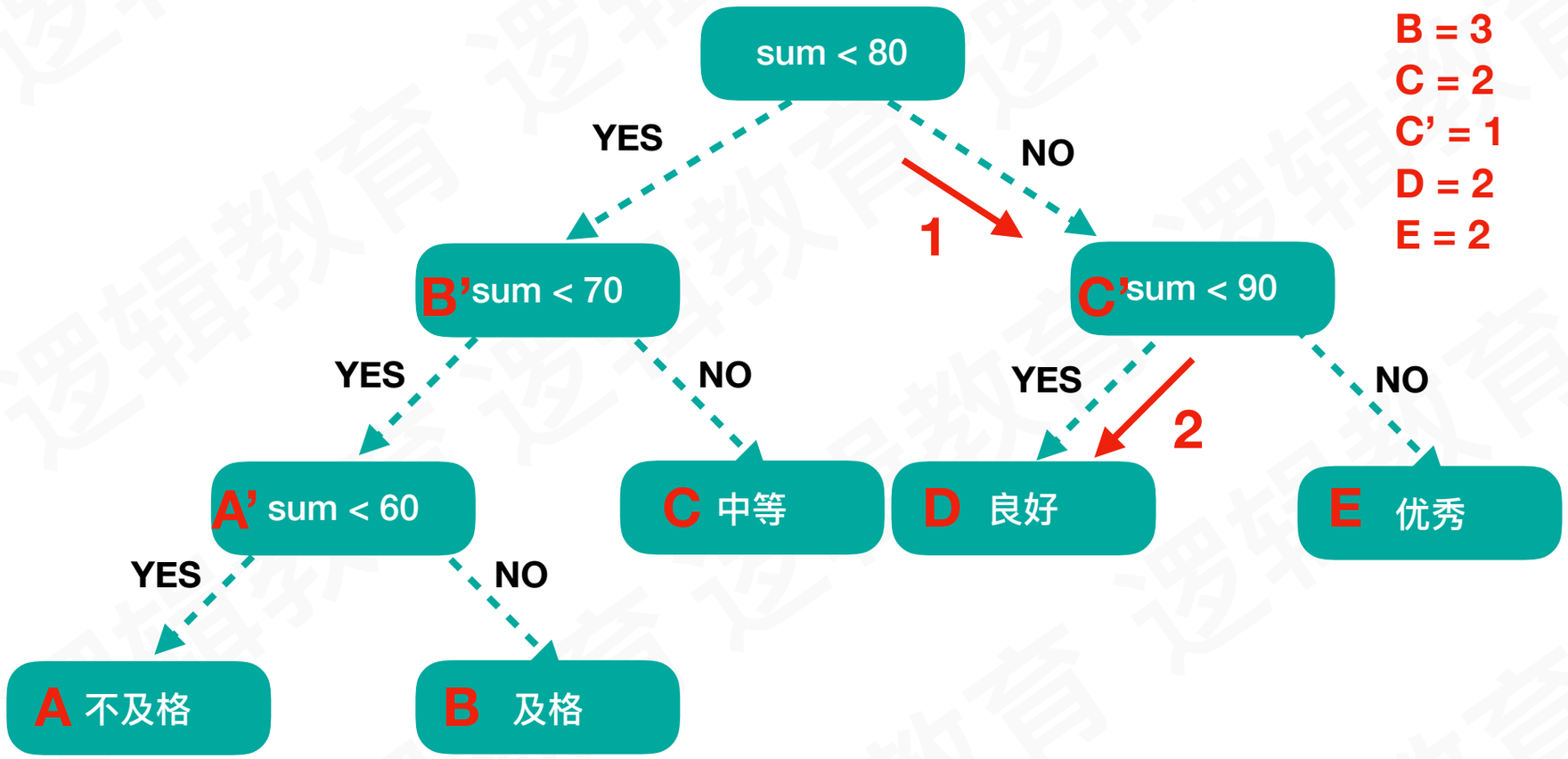
课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

哈夫曼思考

路径长度: 例如结点D 的路程长度为?

D = 2

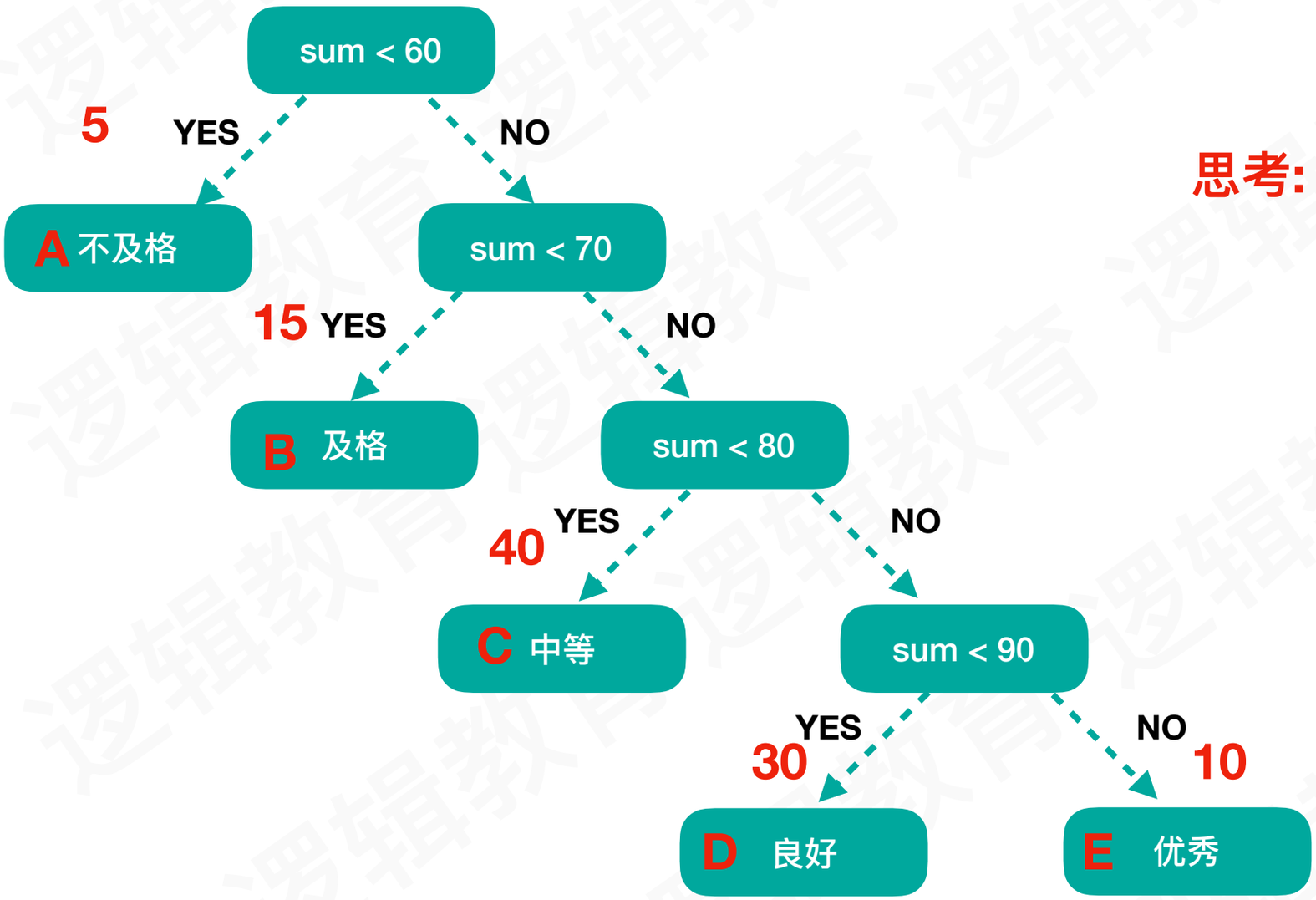


树的路径长度就是从树根到每一个结点的路径长度之和

sum = 1+2+3 +3 +2 +1 + 2 + 2 = 16
B’ = 1
A’ = 2
A = 3
B = 3
C = 2
C’ = 1
D = 2
E = 2

课程研发:CC老师
课程授课:CC老师

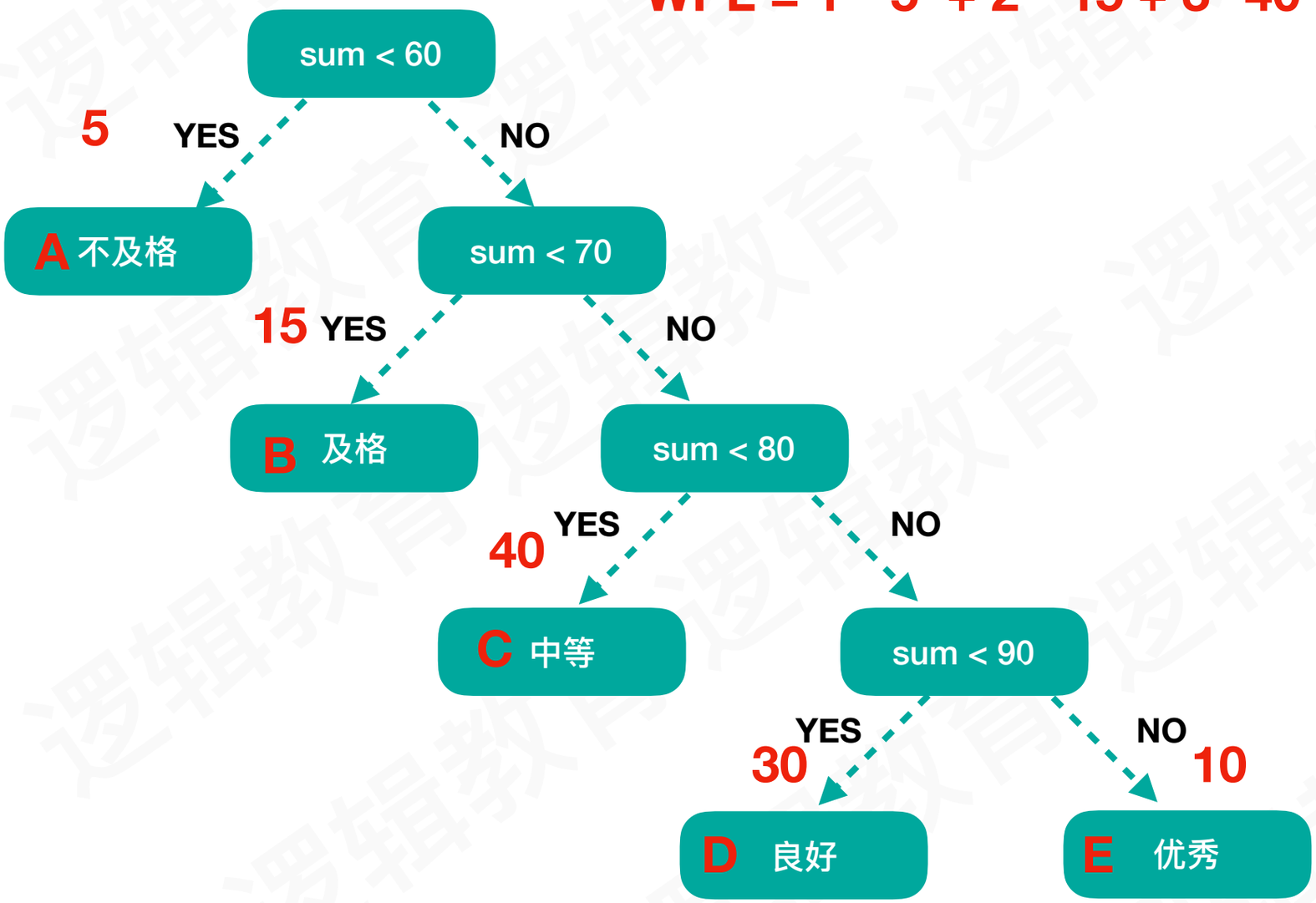
哈夫曼思考



思考: 计算这颗树的WPL?

课程研发:CC老师
课程授课:CC老师

哈夫曼思考

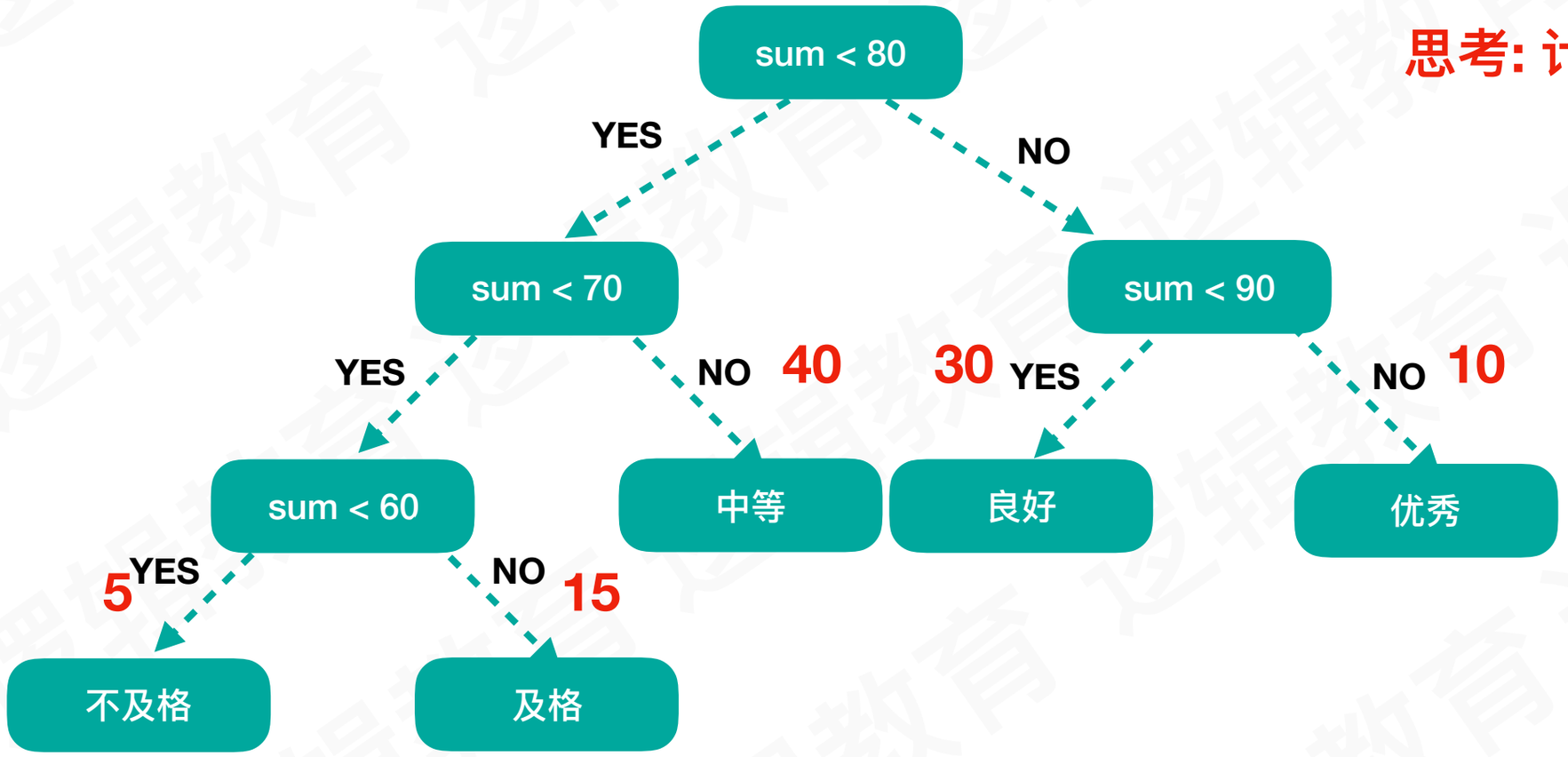


思考: 计算这颗树的WPL?

WPL = 1 * 5 + 2 * 15 + 3 *40 +4 * 30 + 4 * 10 = 315

课程研发:CC老师
课程授课:CC老师

哈夫曼思考



思考: 计算这颗树的WPL?

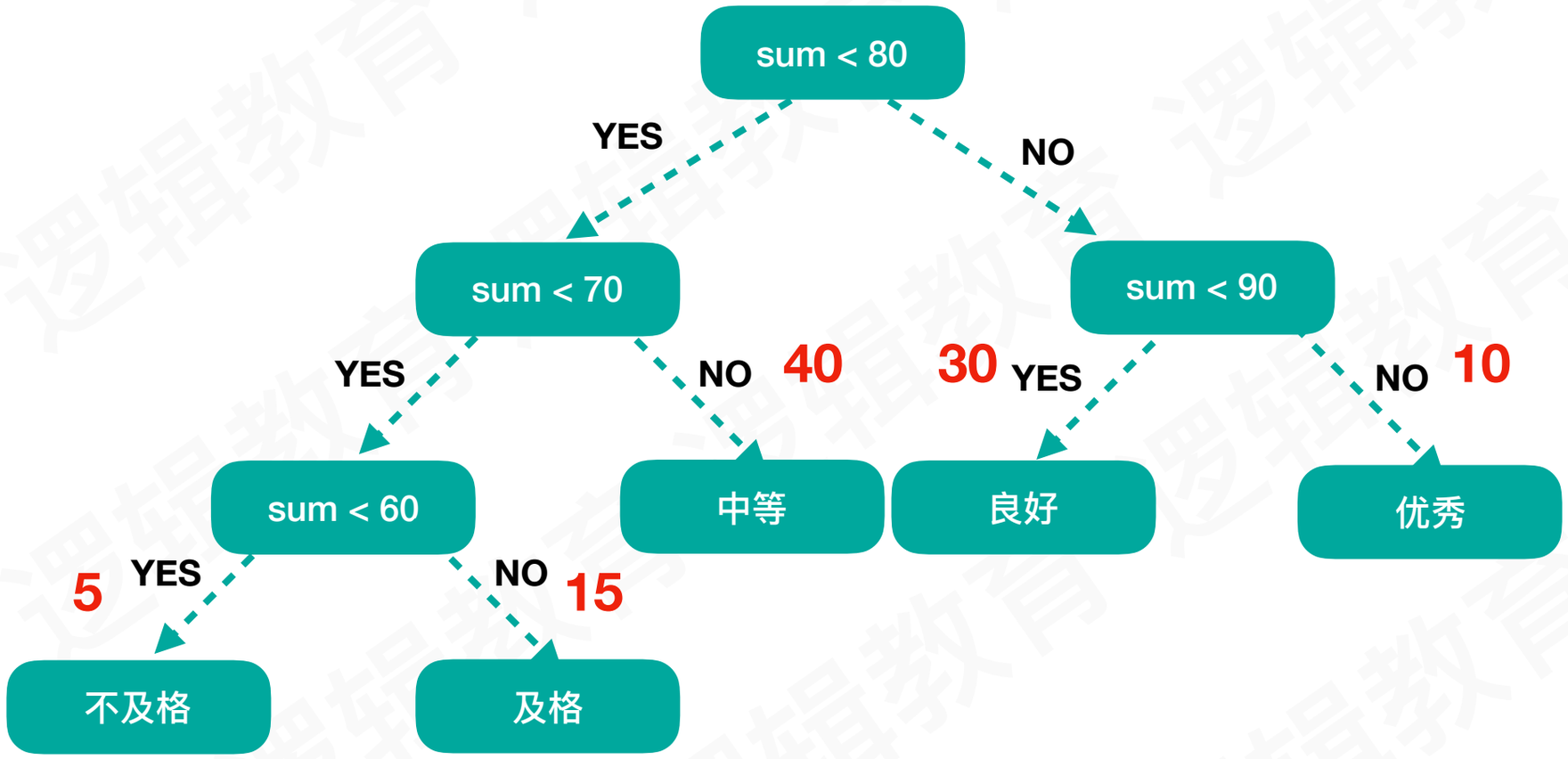
课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

哈夫曼思考

思考: 计算这颗树的WPL?

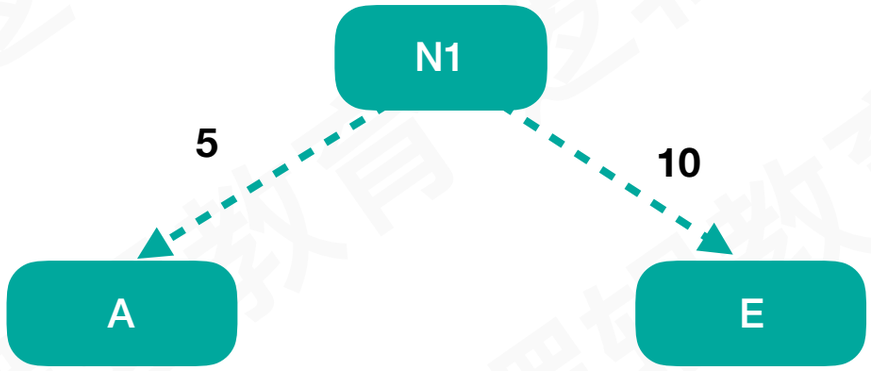
WPL = 5 * 3 + 15 * 3 + 40 * 2 + 30 * 2 +10 * 2 = 220

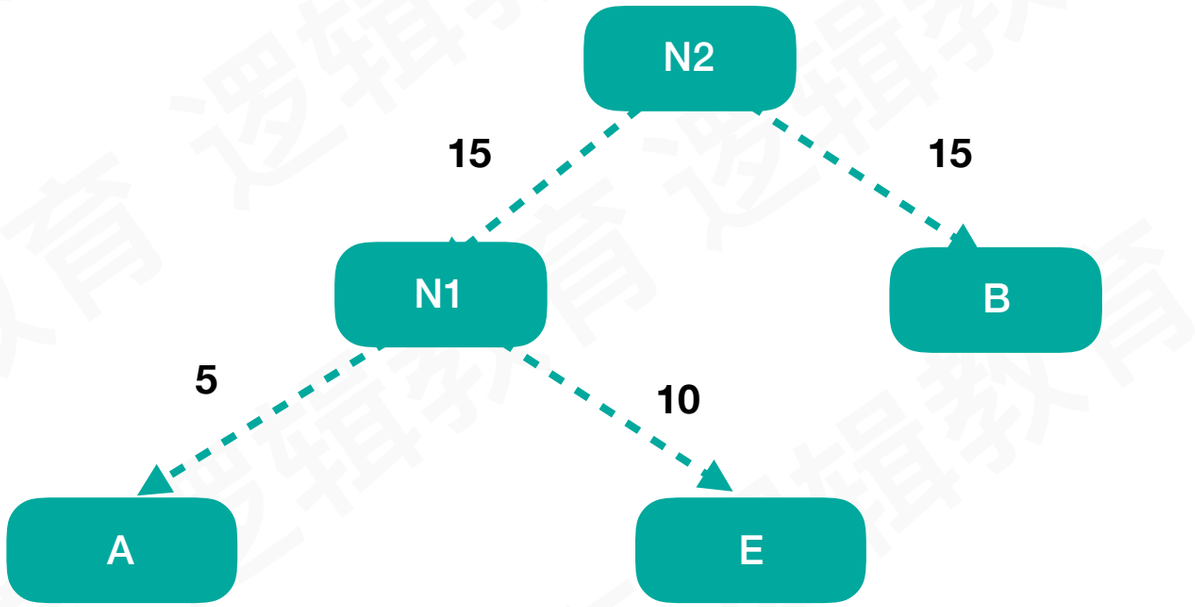


课程研发:CC老师
课程授课:CC老师

哈夫曼树的构建

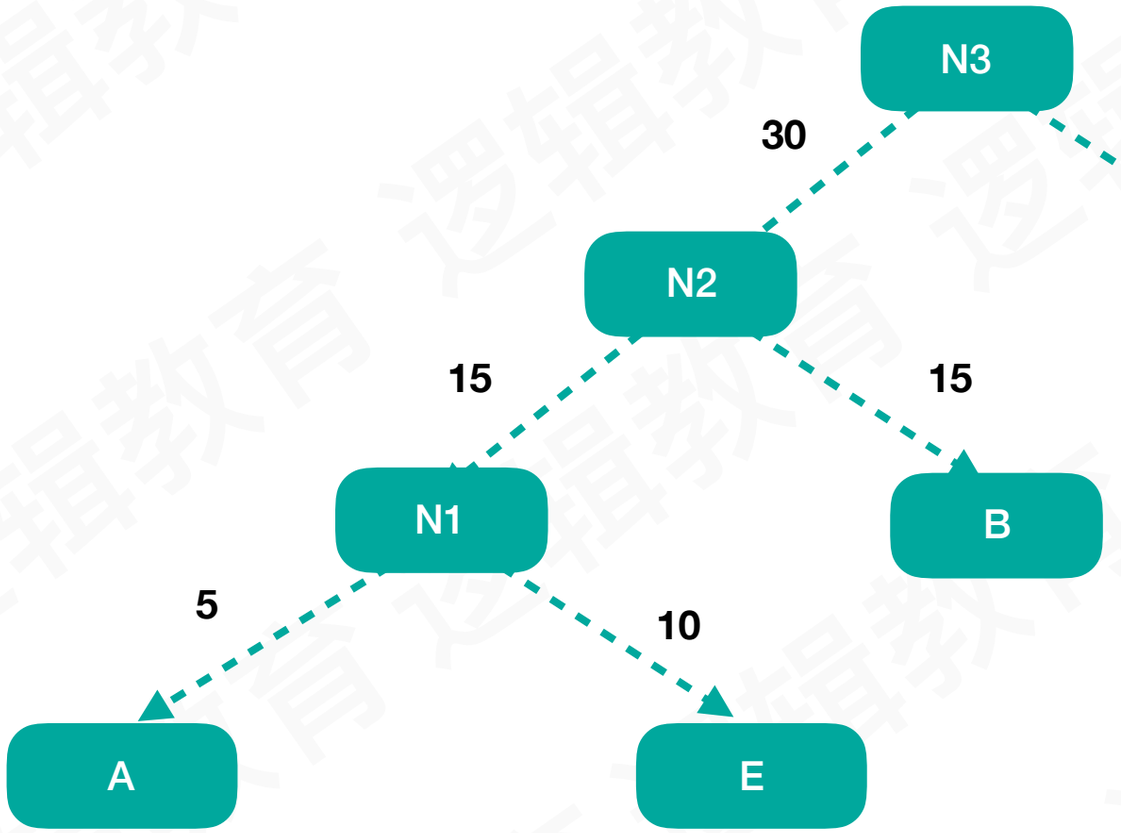
A 5
E 10
B 15
D 30
C 40





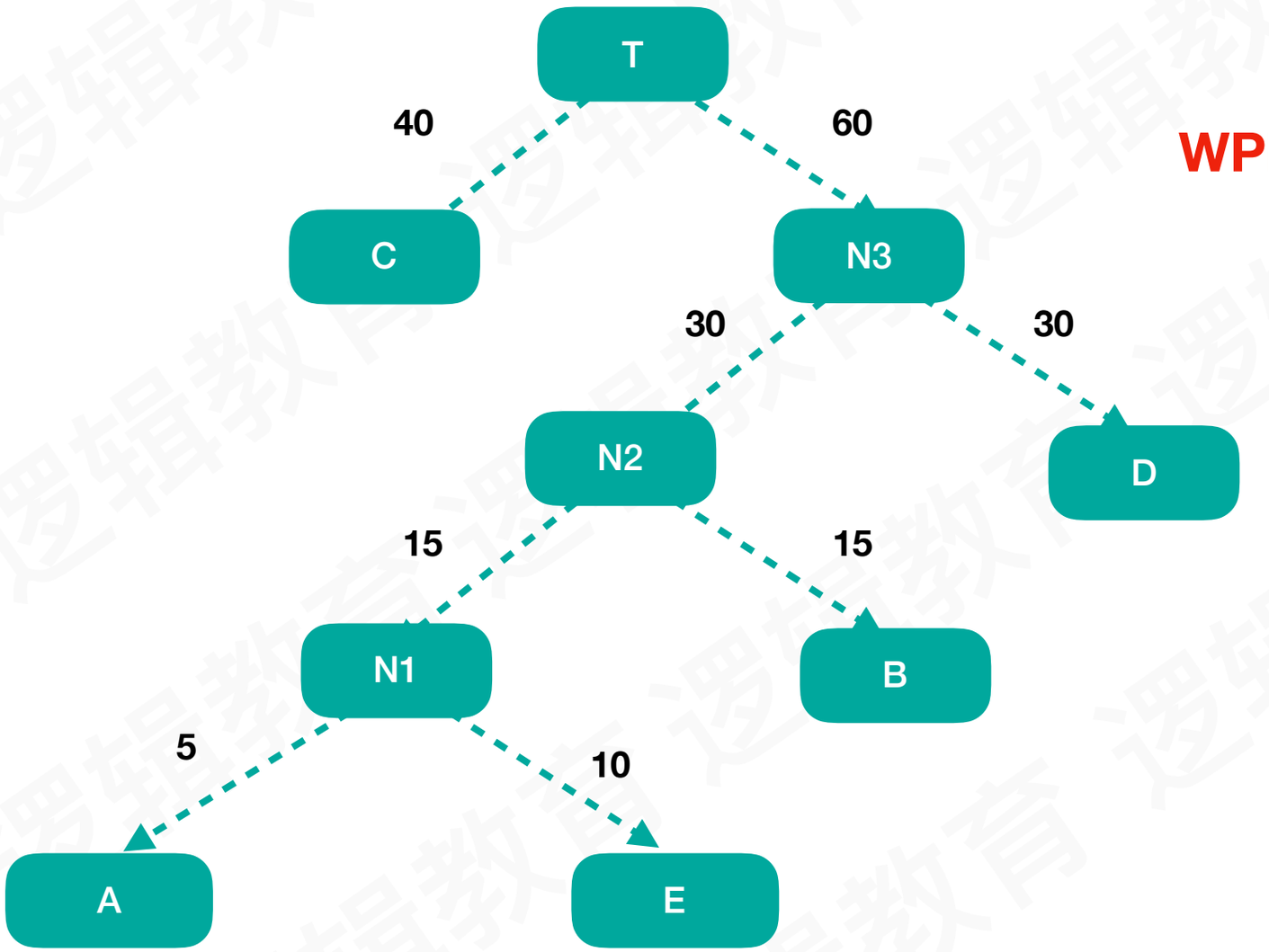
课程研发:CC老师
课程授课:CC老师

哈夫曼树的构建



课程研发:CC老师
课程授课:CC老师

哈夫曼树的构建



思考: 计算这颗树的WPL?

WPL =40 * 1 + 30* 1 + 15 * 2 + 10 * 3 + 5 * 4 = 205

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

哈夫曼编码思考

文字内容: **ABCDEF**
文字内容: **BADCADFEED**

文字内容: ABCDEF
000 001 010 011 100 101

文字内容: BADCADFEED
001 000 011 010 000 011 101 100 100 011

课程研发:CC老师
课程授课:CC老师

哈夫曼编码思考

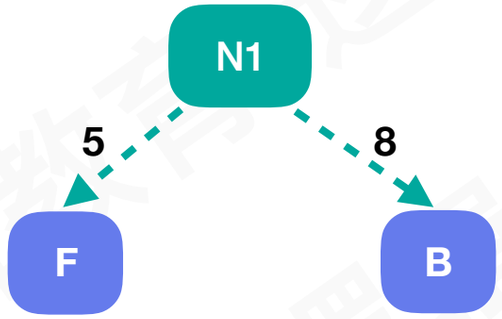
第①步

A 000 27		F 101 5
B 001 8		B 001 8
C 010 15	排序	C 010 15
D 011 15		D 011 15
E 100 30		A 000 27
F 101 5		E 100 30

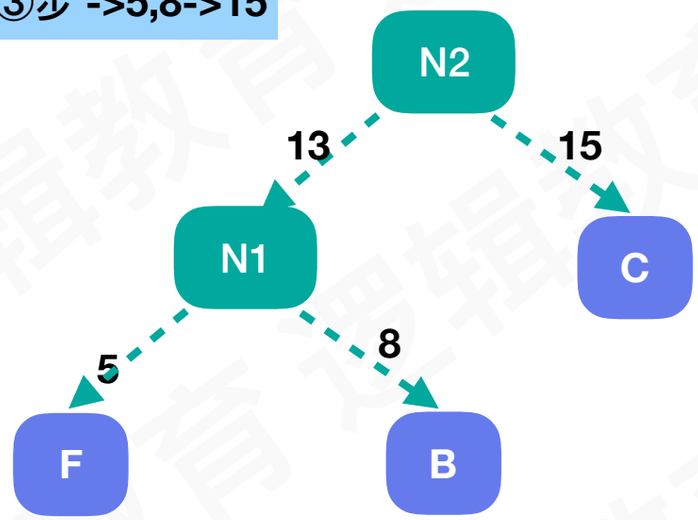
课程研发:CC老师
课程授课:CC老师

哈夫曼编码思考

第②步—> 5 , 8



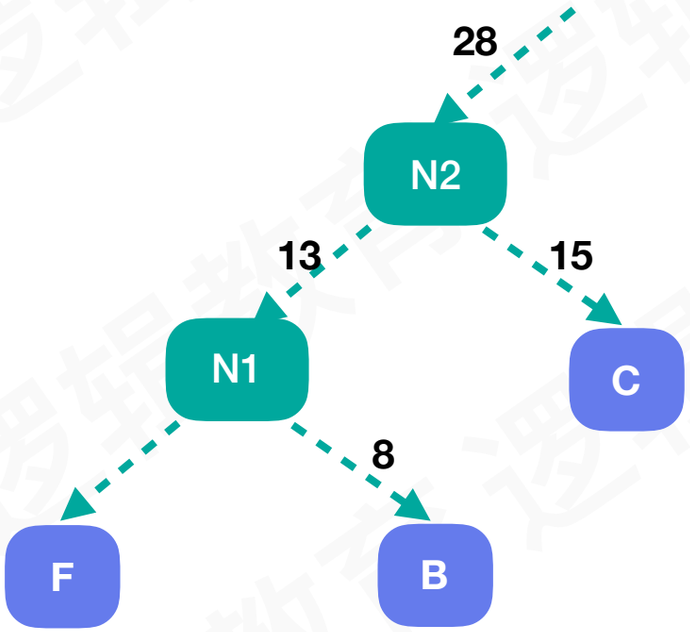
第③步 ->5,8->15



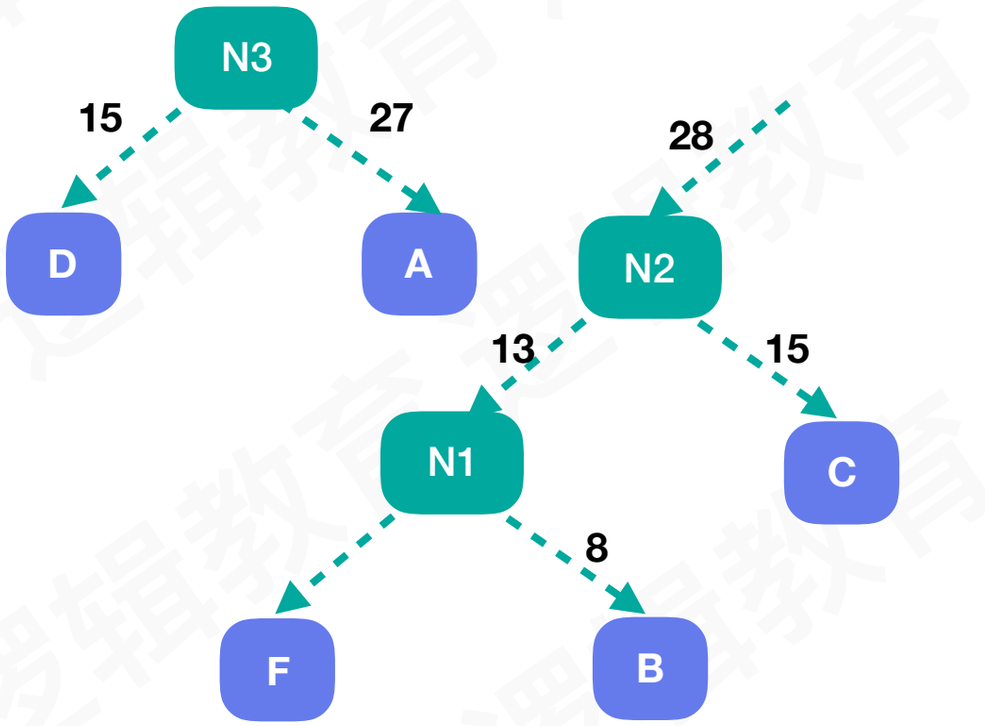
n+i

课程研发:CC老师
课程授课:CC老师

第④步 ->28

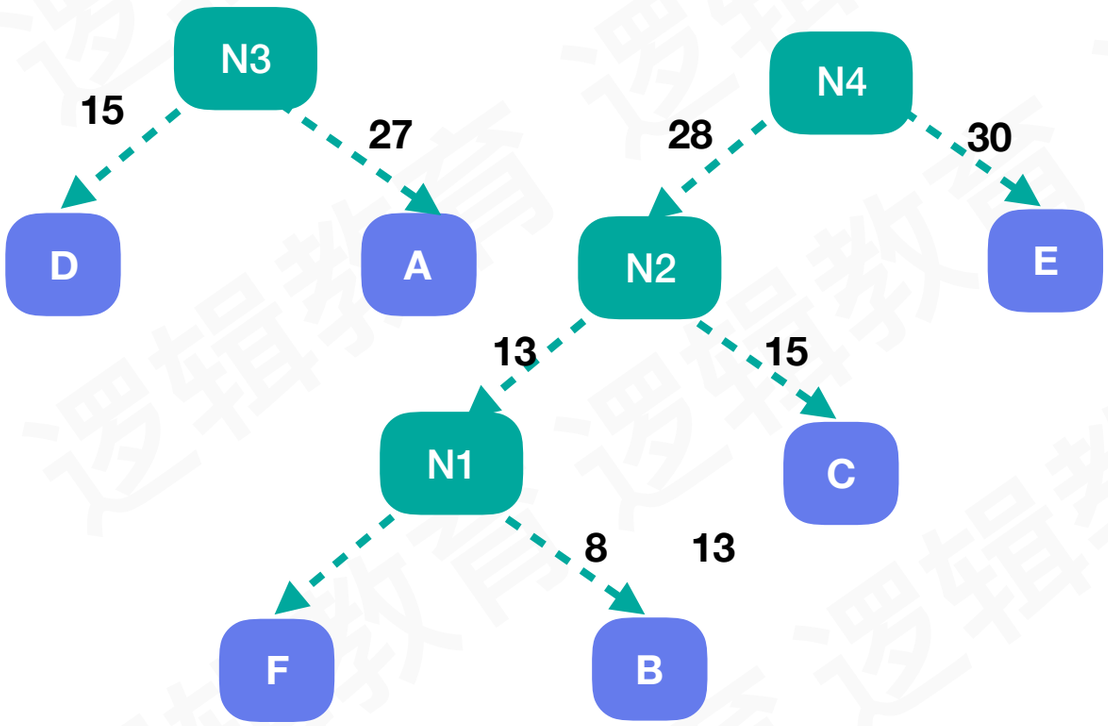


第⑤步 ->15,27

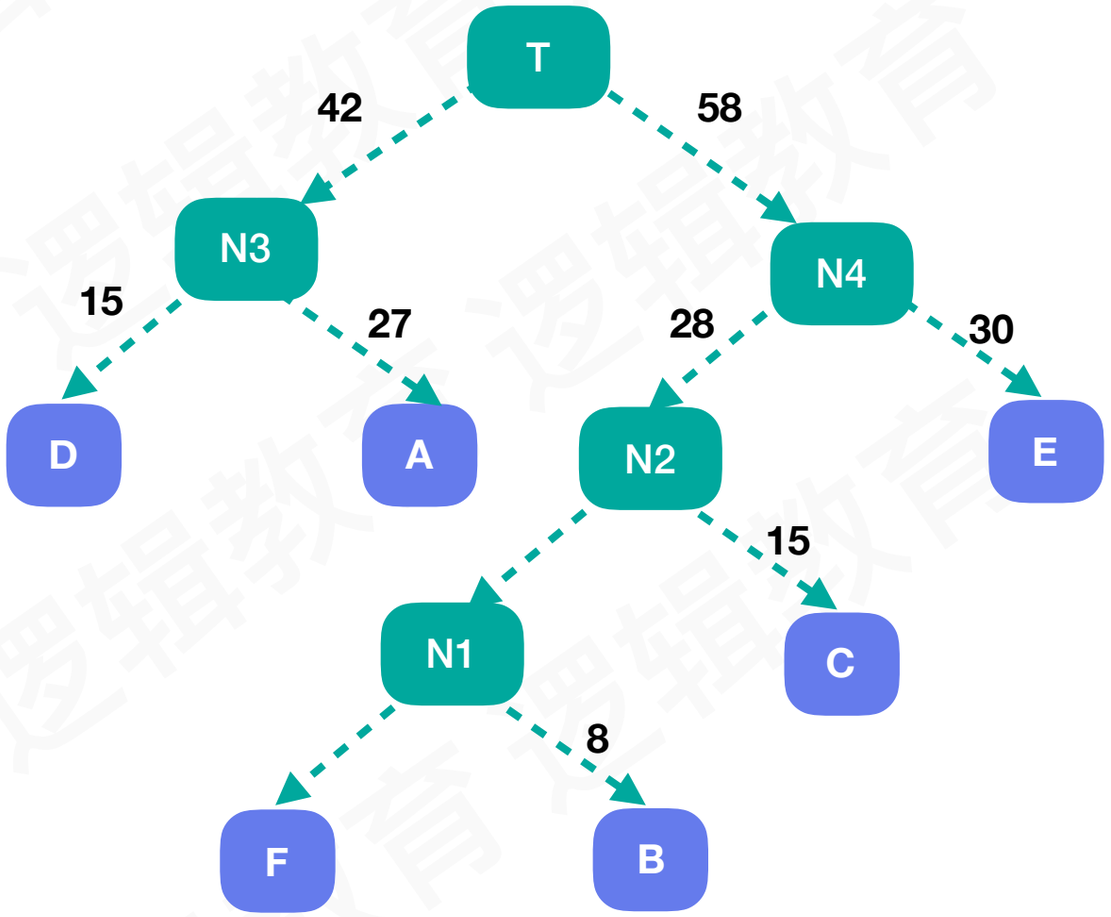


课程研发:CC老师
课程授课:CC老师

第⑥步 —>15,27,30

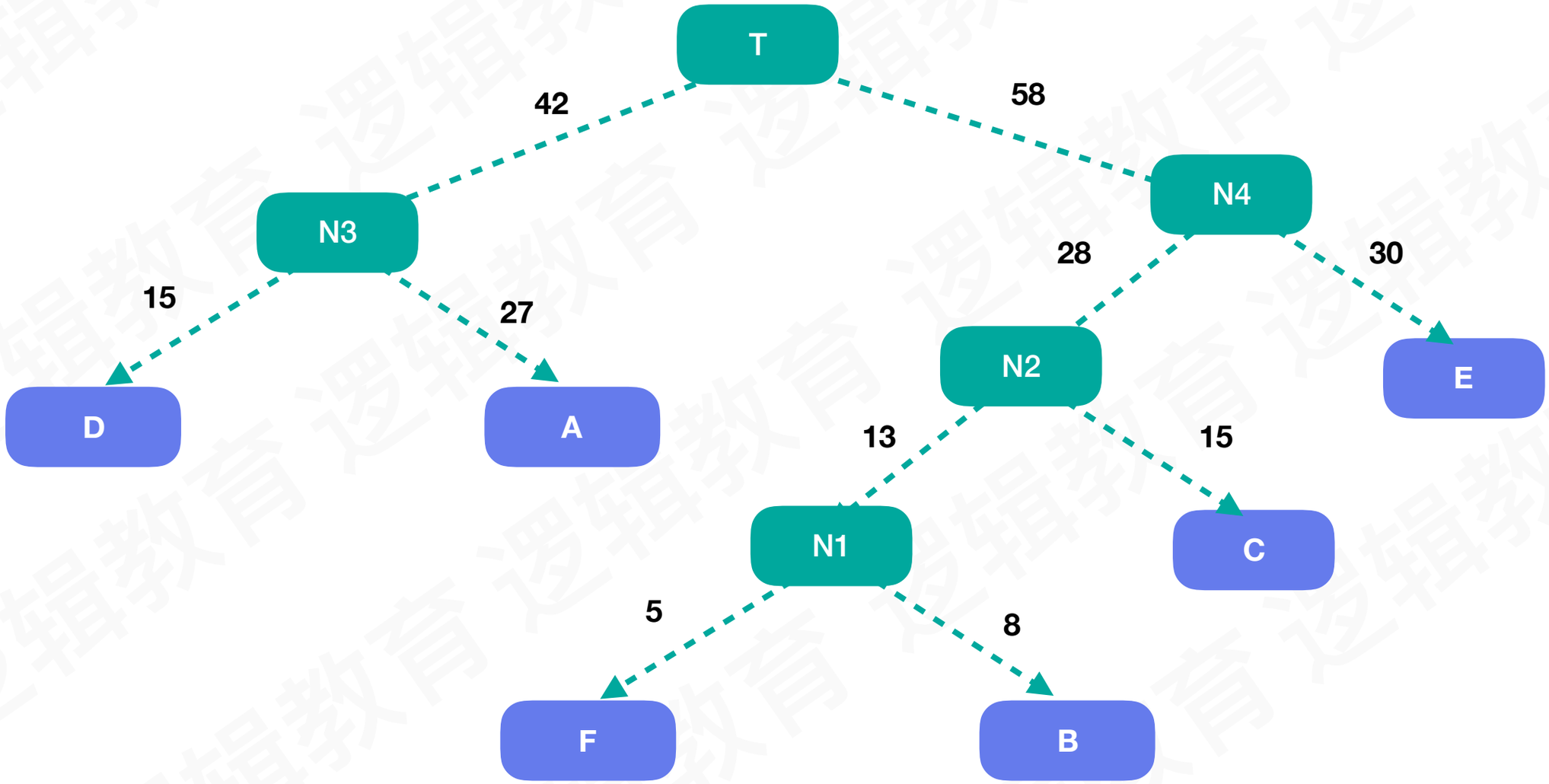


第⑦步 —>15,27,30



课程研发:CC老师
课程授课:CC老师

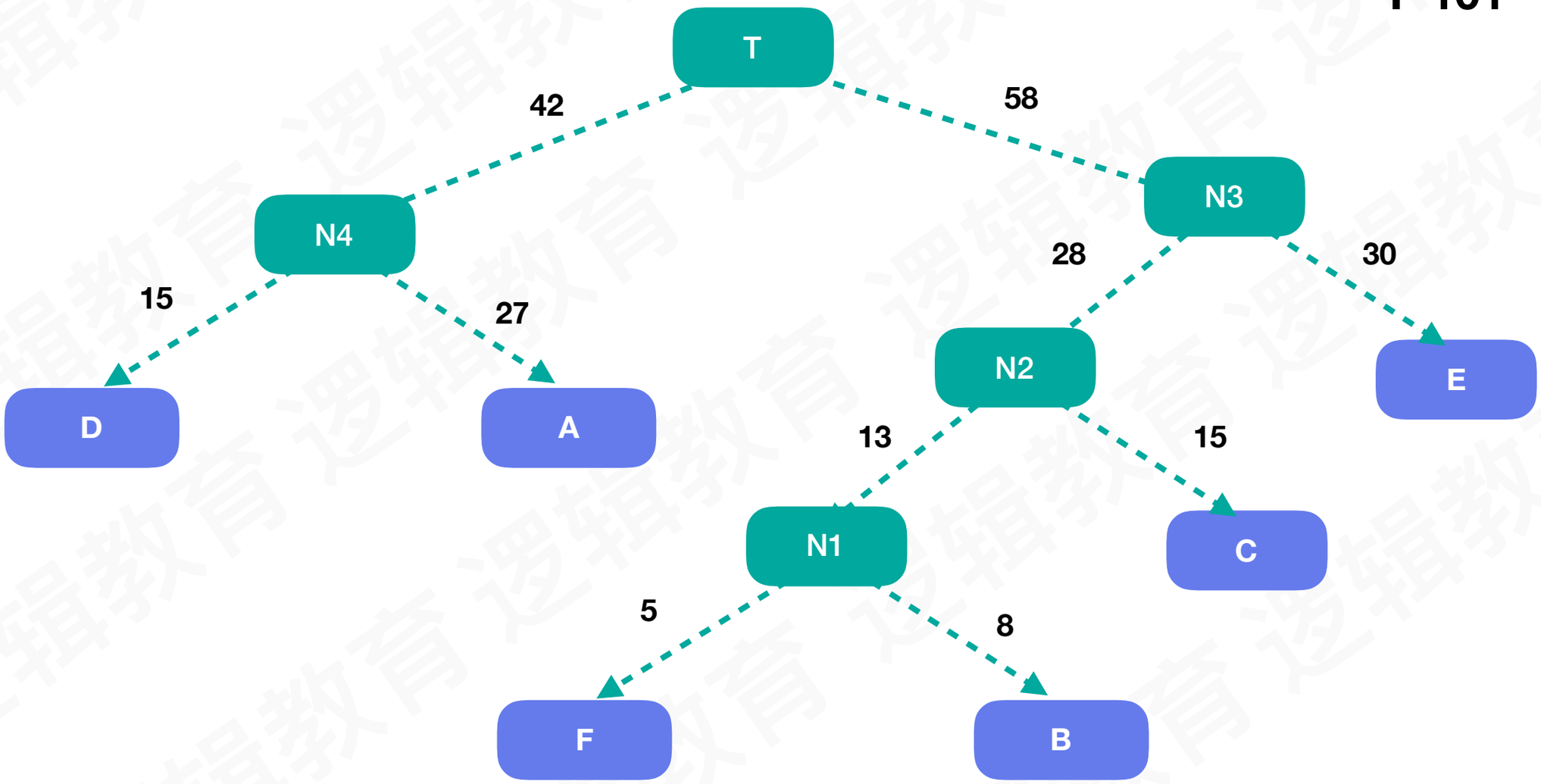
哈夫曼编码思考



课程研发:CC老师
课程授课:CC老师

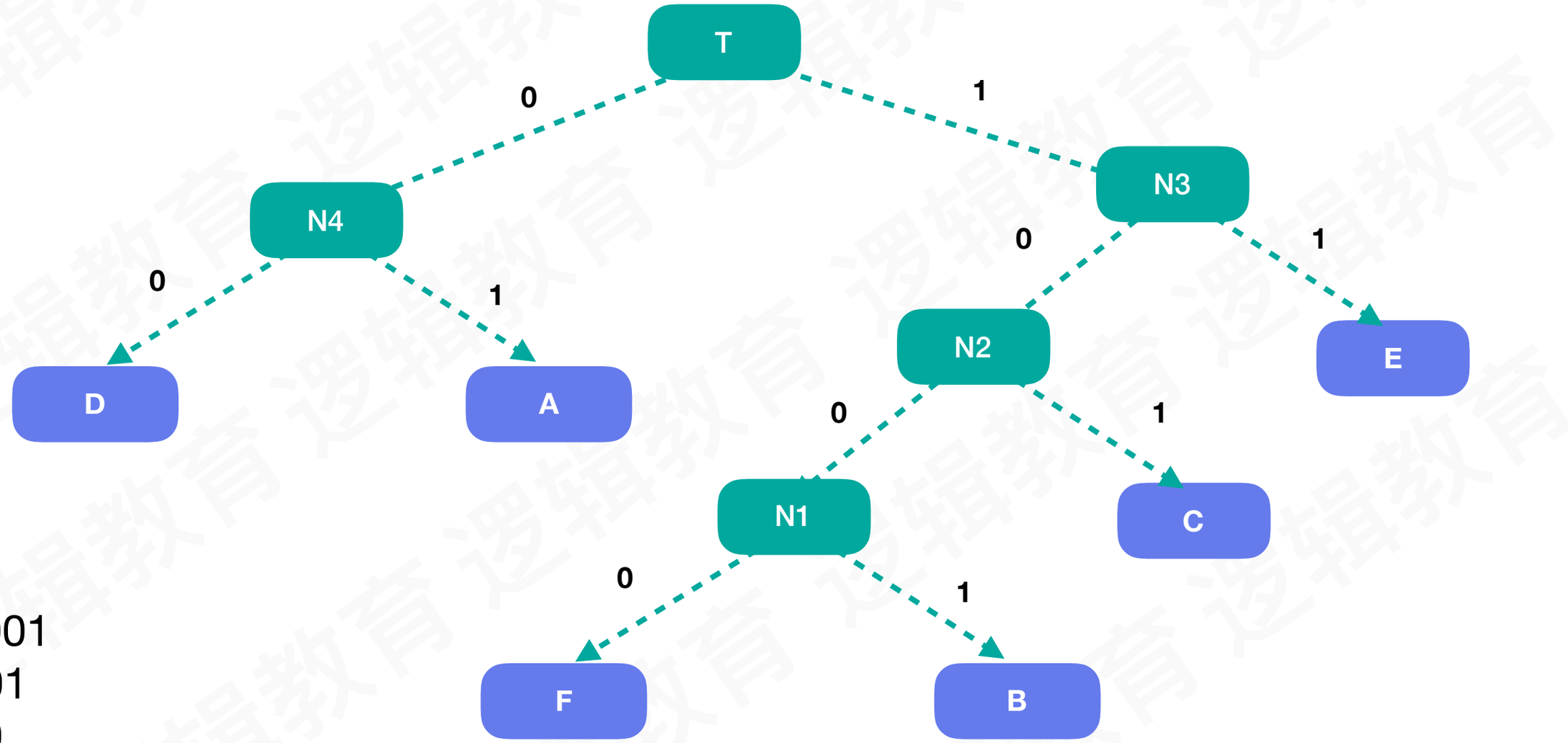
哈夫曼编码思考

A 000 27
B 001 8
C 010 15
D 011 15
E 100 30
F 101 5



课程研发:CC老师
课程授课:CC老师

哈夫曼编码思考



A 01
B 1001
C 101
D 00
E 11
F 1000

课程研发:CC老师
课程授课:CC老师

哈夫曼编码思考

A 01
B 1001
C 101
D 00
E 11
F 1001

BADCADFEED 编码

原编码二进制: 001 000 011 010 000 011 101 100 100 011(共30个字符)

新编码二进制: 1001 01 00 101 01 00 1001 11 11 00(共25个字符)

课程研发:CC老师
课程授课:CC老师

哈夫曼算法的实现

哈夫曼树的结点的形式:

weight	parent	leftChild	rightChild	flag

哈夫曼树的存储表示:

```
typedef struct HaffNode//哈夫曼树的结点结构
{
    int weight;//权值
    int flag;//标记
    int parent;//双亲结点下标
    int leftChild;//左孩子下标
    int rightChild;//右孩子下标
}HaffNode;
```

```
flag 作用:

0: 表示该结点未合并到哈夫曼树中
1: 表示该结点已经合并到哈夫曼树中
```

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

哈夫曼树的实现

哈夫曼树的实现思路:

1. 初始化哈夫曼二叉树
2. 循环不断找到结点中,最小的2个结点值. 加入到哈夫曼树中.

课程研发:CC老师
课程授课:CC老师

哈夫曼编码代码实现

哈夫曼编码的结点的形式:

哈夫曼编码的存储表示:

```
typedef struct Code//存放哈夫曼编码的数据元素结构
{
    int bit[MaxBit];//数组
    int start;  //编码的起始下标
    int weight;//字符的权值
}Code;
```

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

哈夫曼编码的代码实现

哈夫曼树的实现思路:

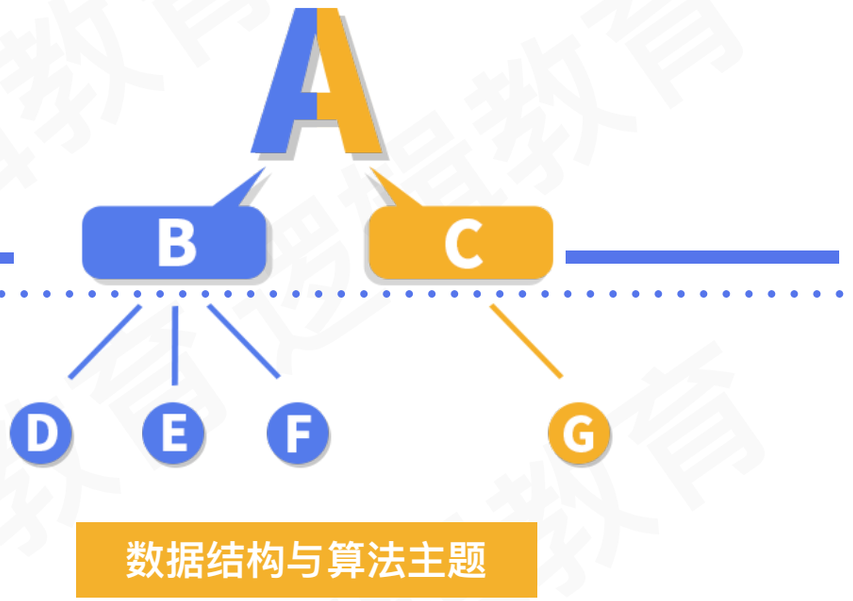
1. 获取根据权值构建的哈夫曼树
2. 循环遍历[0,n]个结点;
3. 创建临时结点cd ,从根结点开始对齐进行编码,左孩子为0,右孩子为1;
4. 将编码后的结点存储haffCode[i]
5. 设置HaffCode[i]的开始位置以及权值;

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

Class Ending !

thanks, see you next time



@CC老师

全力以赴，非同凡"想"

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

课程研发:CC老师
课程授课:CC老师