

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

《CSP-J 初赛真题分类解析》

Day03-基础数据结构

主讲人：小武老师



声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

基础数据结构

数组与字符串、链表、栈、队列、树、图





基础数据结构



数据结构



数组的优缺点？

查询索引

$O(1)$

插入操作

$O(n)$

删除操作

$O(n)$

查询元素

$O(n)$



数组与字符串



数组与字符串

数组

一维数组

多维数组

字符串

C风格字符串

`scanf("%s", s)`

`printf("%s", s)`

`char ch = getchar()`

`putchar(ch)`

`while ((ch=getchar()) != 'EOF')`

当getchar读到文件末尾或者结束时，
它会返回一个EOF，此时结束循环。

C++中的string类

`string`

`getline(cin, s)`



字符数组



注意使用到了 `cstring` 头文件，其中一些的用法如下：

<code>strlen(s)</code>	获得字符数组的 长度 ，数到 ' \0'
<code>strcpy(a,b)</code>	将 b 字符数组的数据 复制 到 a
<code>strcmp(a,b)</code>	比较两个字符串， $a > b$ 则返回 1
<code>char a[100] ;</code> <code>strcpy(a, "hello");</code>	给字符数组 赋值常量



string类



使用 string 头文件来操作字符串，其中一些的用法如下

功能	说明
string s;	定义一个名字为 s 的字符串变量
s+=str 或 s.append(str)	在字符串 s 后面拼接字符串 str
s<str	比较字符串 s 的是否在字符串 str 的字典序之前
s.size() 或 s.length()	得到字符串 s 的长度
s.substr(pos,len)	截取字符串 s，从第 pos 个位置开始 len 个字符，并返回这个字符串
s.insert(pos, str)	字符串 s 的第 pos 个字符之前，插入字符串 str，并返回这个字符串
s.find(str, [pos])	字符串 s 中从第 pos 个字符开始寻找 str，并返回位置，如果找不到返回 string::npos。pos 可以省略，默认值是 0



子串与子序列



子串

字符串 S 的子串 $S[i..j]$, $i \leq j$, 表示 S 串中从 i 到 j 这一段, 也就是顺次排列:
 $S[i], S[i+1], \dots, S[j]$ 形成的字符串。有时也会用 $S[i..j]$, $i > j$ 来表示空串。

子序列

字符串 S 的 子序列 是从 S 中将若干元素提取出来并不改变相对位置形成的序列, 即:
 $S[p_1], S[p_2], \dots, S[p_k]$, $1 \leq p_1 \leq p_2 \leq \dots \leq p_k \leq |S|$ 形成的字符串。

hello 的子串和子序列?



子串与子序列



子串

有n个字符的字符串的子串个数为：

$$n + (n - 1) + (n - 2) + \cdots + 2 + 1 = \frac{n(n+1)}{2}, \text{ 再加一个空串}$$

子序列

有n个字符的字符串的子序列个数为：

$$C_n^0 + C_n^1 + C_n^2 + \cdots + C_n^n = 2^n$$



子串与子序列



【2017普及组选择题】若串 $S = \text{"copyright"}$ ，其子串的个数是（ 46 ）。

【2012普及组选择题】原字符串中任意一段连续的字符所组成的新字符串称为子串。
则字符“AAABBBCCC”共有（ 36 ）个不同的非空子串。 要减去9个重复子串

【2016普及组选择题】以下关于字符串的判定语句中正确的是（ A ）。

- A. 字符串是一种特殊的线性表
- B. 串的长度必须大于零
- C. 字符串不可以用数组来表示
- D. 空格字符组成的串就是空串



链表



链表

单向链表

循环链表

双向链表

可达信奥—小武老师—keda.ac



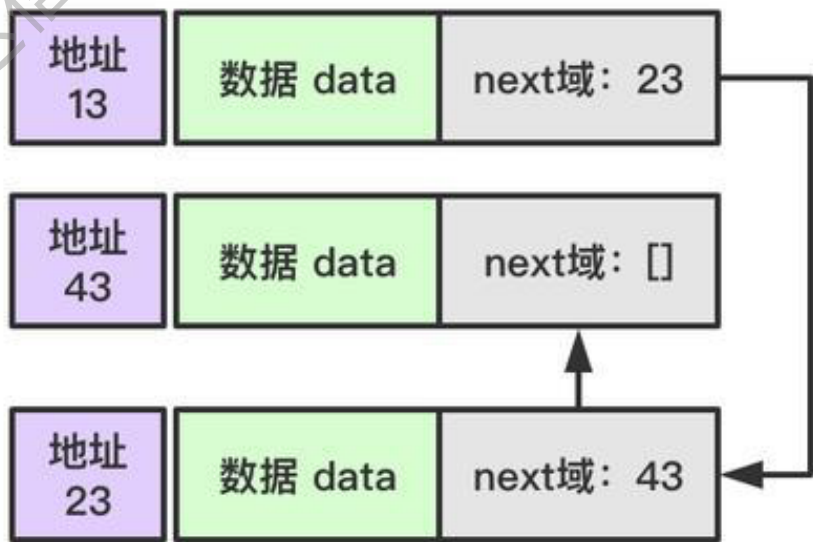
单向链表



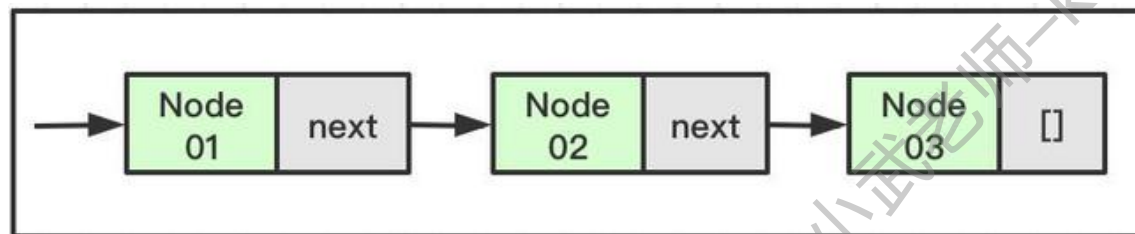
存储方式分类

顺序存储结构。如数组，可以随机存取表中的任一元素，逻辑关系相邻的两个元素在物理位置上也是相邻的。缺点是长度不确定时需要较大存储空间。线性表的容量一经定义就难以扩充，插入和删除线性表元素时，需要移动大量元素，浪费时间。

链式存储结构。可以用一组任意的存储单元（可以是连续的，也可以是不连续的）存储线性表的数据元素，可以充分利用存储器的零碎空间。



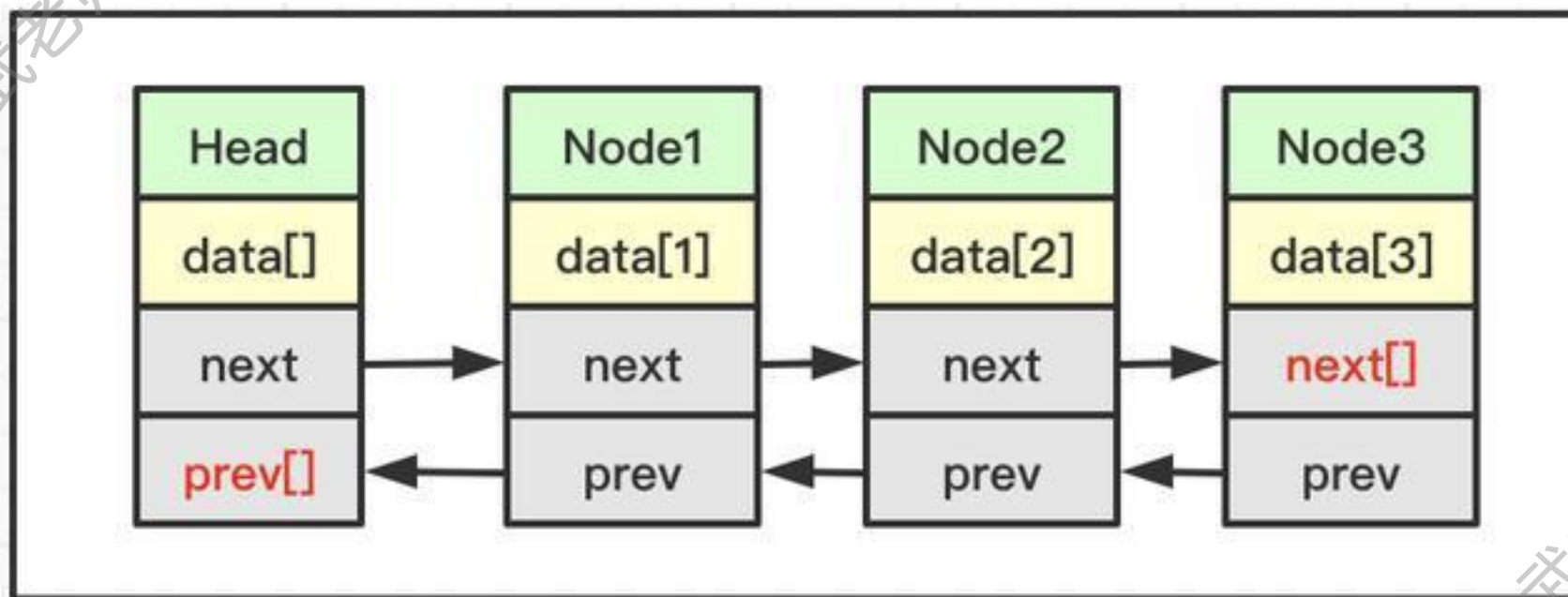
内存存储



逻辑结构

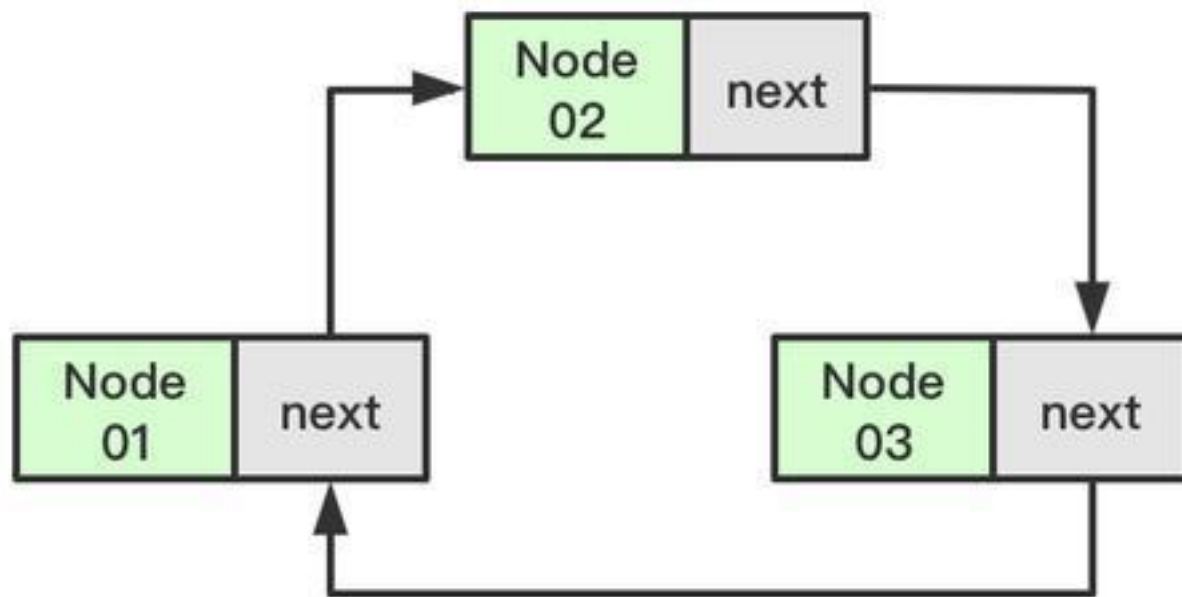


双向链表





循环链表





链表真题



【2015提高组选择题】线性表若采用链表存储结构，要求内存中可用存储单元地址（D）。

- A. 必须连续
- B. 部分地址必须连续
- C. 一定不连续
- D. 连续不连续均可

【2019CSP-J选择题】链表不具备的特点是（D）。

- A. 插入删除不需要移动元素
- B. 不必事先估计存储空间
- C. 所需空间与线性表长度成正比
- D. 可随机访问任一元素



链表真题



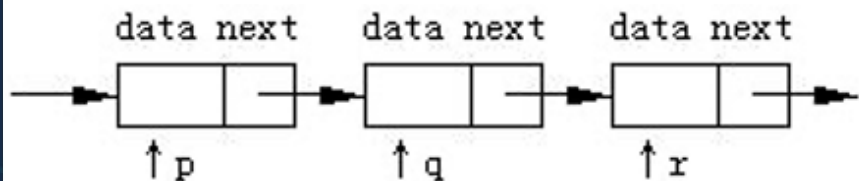
【2017普及组选择题】向一个栈顶指针为 hs 的链式栈中插入一个指针 s 指向的结点时，应执行（B）。

- A. $hs \rightarrow next = s;$
- B. $s \rightarrow next = hs; hs = s;$
- C. $s \rightarrow next = hs \rightarrow next; hs \rightarrow next = s;$
- D. $s \rightarrow next = hs; hs = hs \rightarrow next;$

【2014提高组选择题】有以下结构体说明和变量定义，如图所示，指针 p 、 q 、 r 分别指向一个链表中的三个连续结点。

```
struct node {  
    int data;  
    node *next;  
} *p, *q, *r;
```

现要将 q 和 r 所指结点的先后位置交换，同时要保持链表的连续，以下程序段中错误的是（D）。



- A. $q \rightarrow next = r \rightarrow next; p \rightarrow next = r; r \rightarrow next = q;$
- B. $p \rightarrow next = r; q \rightarrow next = r \rightarrow next; r \rightarrow next = q;$
- C. $q \rightarrow next = r \rightarrow next; r \rightarrow next = q; p \rightarrow next = r;$
- D. $r \rightarrow next = q; q \rightarrow next = r \rightarrow next; p \rightarrow next = r;$



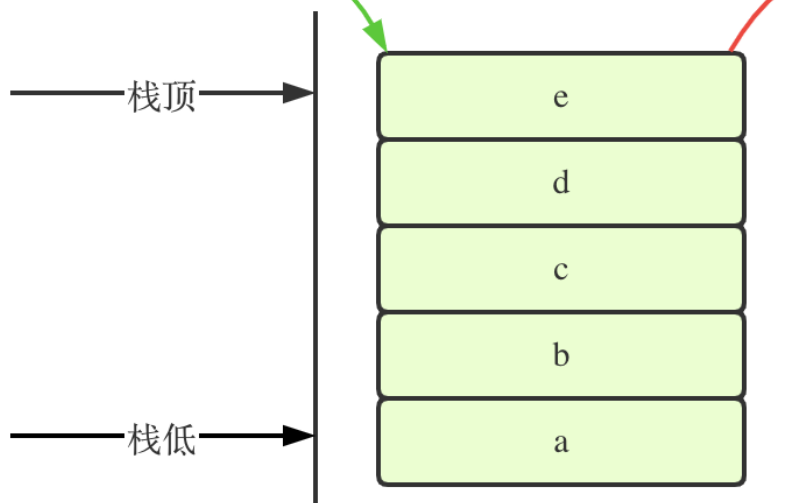
栈—LIFO表



栈是OI中常用的一种线性数据结构，栈的修改是按照后进先出的原则进行的，因此栈通常被称为是后进先出（last in first out）表，简称LIFO表。

按a、b、c、d、e顺序入栈

按e、d、c、b、a顺序出栈



栈的基本操作

- (1) 初始化
- (2) 判空
- (3) 求栈中实际元素的个数
- (4) 进栈（压栈）
- (5) 出栈
- (6) 取栈顶元素

https://blog.csdn.net/yhl_jxy



栈—LIFO表



例题(CSP-J2021) 对于入栈顺序为a、b、c、d、e的序列，下列 () 不是合法的出栈顺序。

① a, b, c, d, e

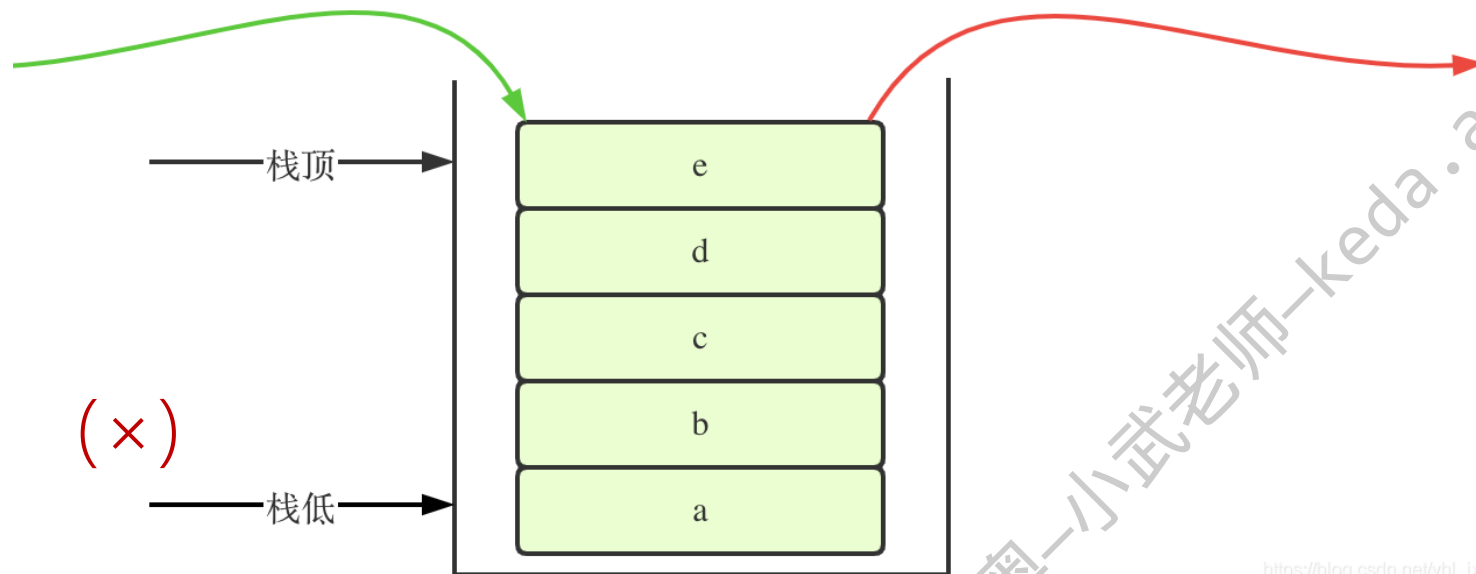
② e, d, c, b, a

③ b, a, c, d, e

④ c, d, a, e, b

按a、b、c、d、e顺序入栈

按e、d、c、b、a顺序出栈



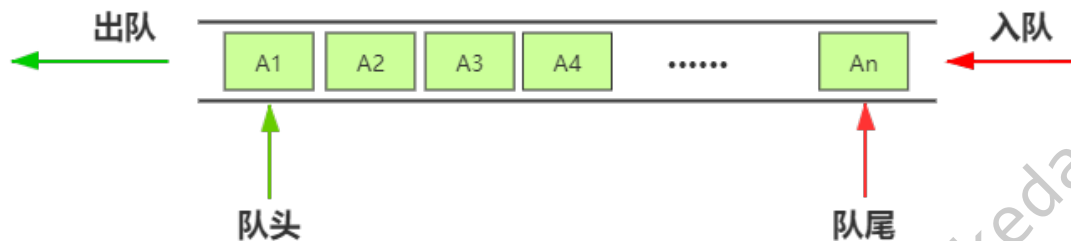
(×)



队列—FIFO表



队列 (queue) 是一种具有「先进入队列的元素一定先出队列」性质的表。由于该性质，队列通常也被称为先进先出 (first in first out) 表，简称 **FIFO** 表。就像排队买东西，排在前面的人买完东西后离开队伍（删除），而后来的人总是排在队伍末尾（插入）。



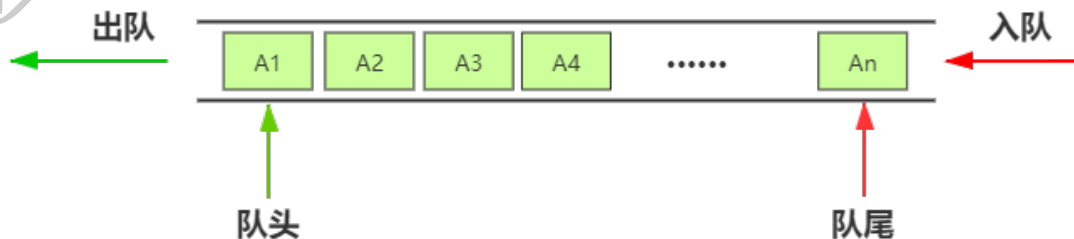
- **队空**：队列中没有任何元素。
- **队满**：队列空间已全被占用。
- **溢出**：当队列已满，却还有元素要入队，就会出现“上溢 (overflow)”；当队列已空，却还要做“出队”操作，就会出现“下溢 (underflow)”。两种情况合在一起称为队列的“溢出”。



队列—FIFO表



队列 (queue) 是一种具有「先进入队列的元素一定先出队列」性质的表。由于该性质，队列通常也被称为先进先出 (first in first out) 表，简称 FIFO 表。就像排队买东西，排在前面的人买完东西后离开队伍 (删除)，而后来的人总是排在队伍末尾 (插入)。



方法一：数组模拟一个队列

```
int q[SIZE], head = 1, tail = 0;
```

- 插入元素 (入队) : `q[++tail]=x;`
- 删除元素 (出队) : `++head;`
- 访问队首/队尾: `q[head]/q[tail]`
- 清空队列: `head=1; tail=0;`



栈和队列真题



【2017普及组选择题】对于入栈顺序为 a, b, c, d, e, f, g 的序列，下列（ C ）不可能是合法的出栈序列

- A. a, b, c, d, e, f, g
- B. a, d, c, b, e, g, f
- C. a, d, b, c, g, f, e
- D. g, f, e, d, c, b, a

【2015提高组选择题】今有一空栈 S，对下列待进栈的数据元素序列 a, b, c, d, e, f 依次进行进栈，进栈，出栈，进栈，进栈，出栈的操作，则此操作完成后，栈 S 的栈顶元素为（ C ）。



栈和队列真题



【2012普及组选择题】如果一个栈初始时空，且当前栈中的元素从栈顶到栈底依次为a，b，c，另有元素d已经出栈，则可能的入栈顺序是（ **D** ）。

- A. a, d, c, b
- B. b, a, c, d
- C. a, c, b, d
- D. d, a, b, c

【2012普及组选择题】在程序运行过程中，如果递归调用的层数过多，会因为（ **A** ）引发错误。

- **A.系统分配的栈空间溢出** - B.系统分配的堆空间溢出
- C.系统分配的队列空间溢出 - D.系统分配的链表空间溢出

数据结构





树形数据结构



树是一种非线性的数据结构，用它能很好地描述有分支和层次特性的数据集合。树型结构在现实世界中广泛存在，是一种“一对多”（One-to-Many）的关系描述，如：

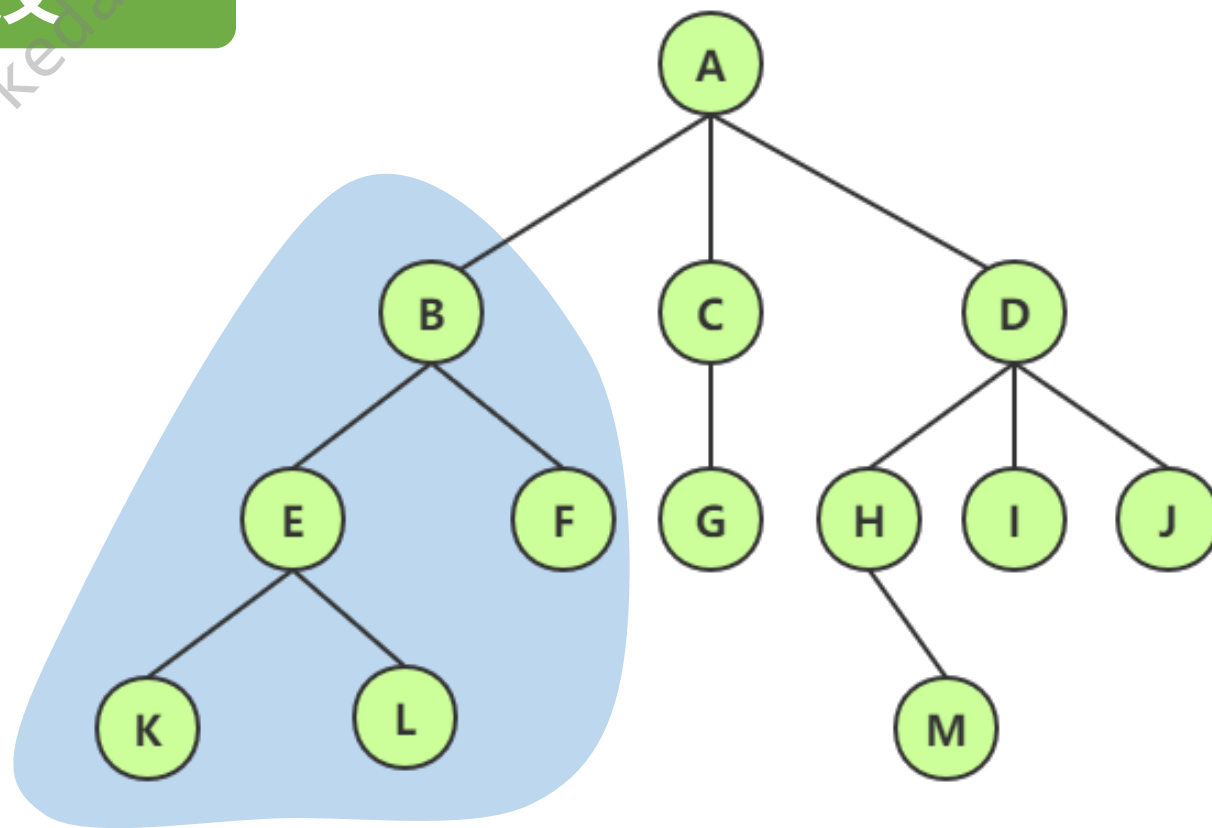
- 社会组织机构的组织关系
- 家族族谱
- 数据库系统中，树形结构是数据库层次模型的基础
- 各种索引和目录，计算机的文件系统



很明显，树结构不仅能表示数据间的指向关系，还能表示出数据的层次关系，而有很明显的递归性质



树的定义



根节点

父结点（双亲结点）

子（孩子）结点

兄弟结点

子树

空树

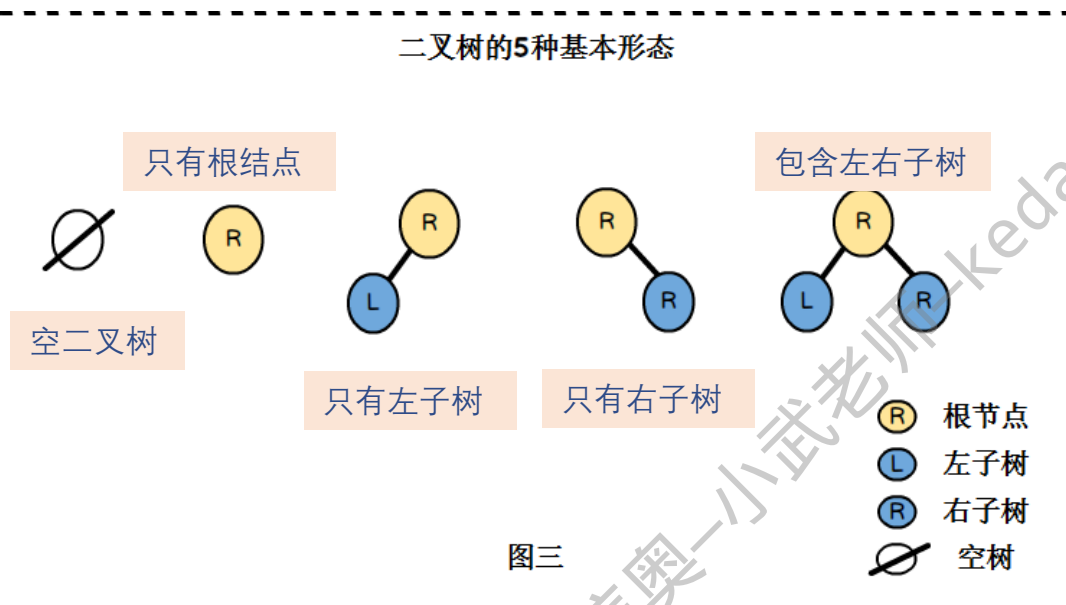
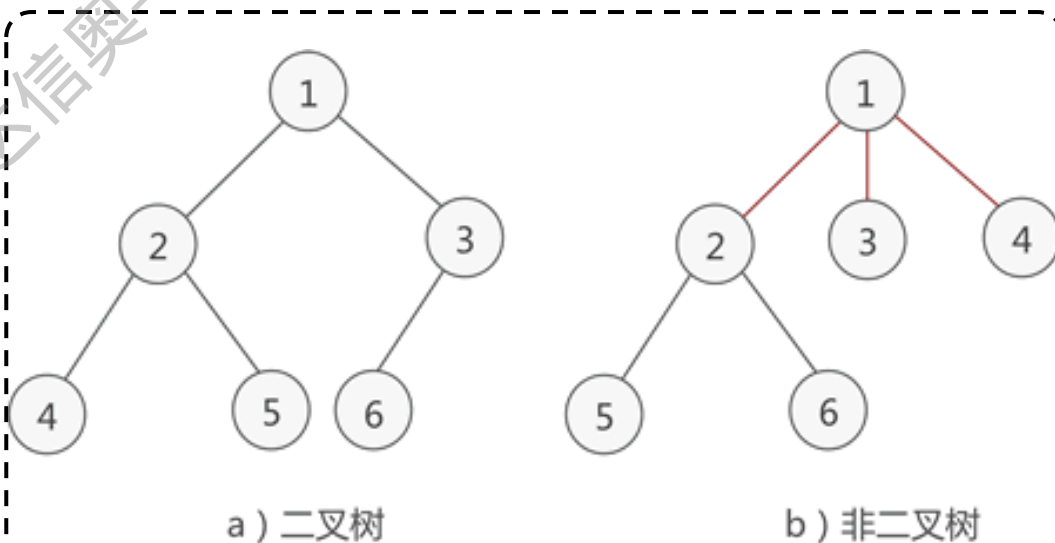
一棵树是由 n 个元素组成的有限集合，每个元素称为一个结点（node），有一个特定的结点，称为根结点或树根（root），除根结点外，其余结点能分成 m 个互不相交的有限集合，其中的每个子集又都是一棵树，这些集合称为这棵树的子树。



二叉树



在计算机科学中，二叉树是一种“树”数据结构，树上的每个节点**最多有两个孩子**，分别为左孩和右孩。两颗子树分别为左子树、右子树。二叉树有5种基本形态。

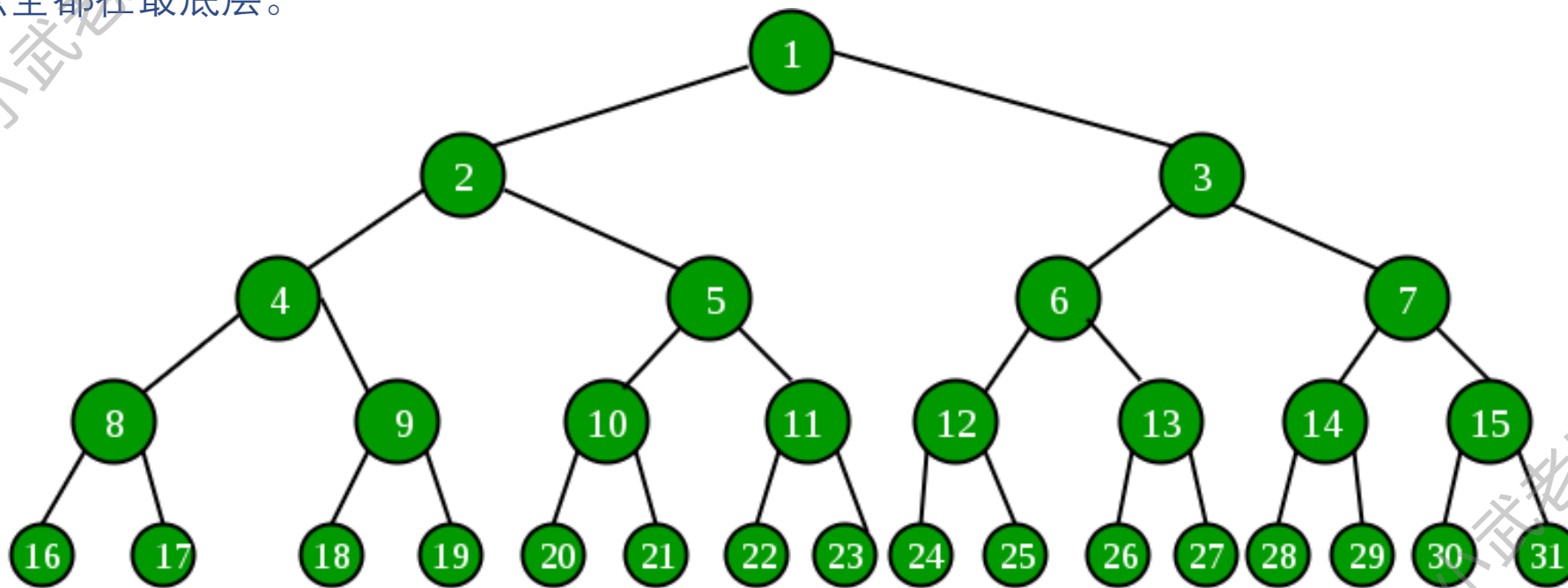




二叉树类型—满二叉树



如果二叉树中除了叶子结点，每个节点都有左右两个子节点，则此二叉树称为满二叉树。叶子结点全都在最底层。



满二叉树总的节点个数为

$$2^n - 1$$

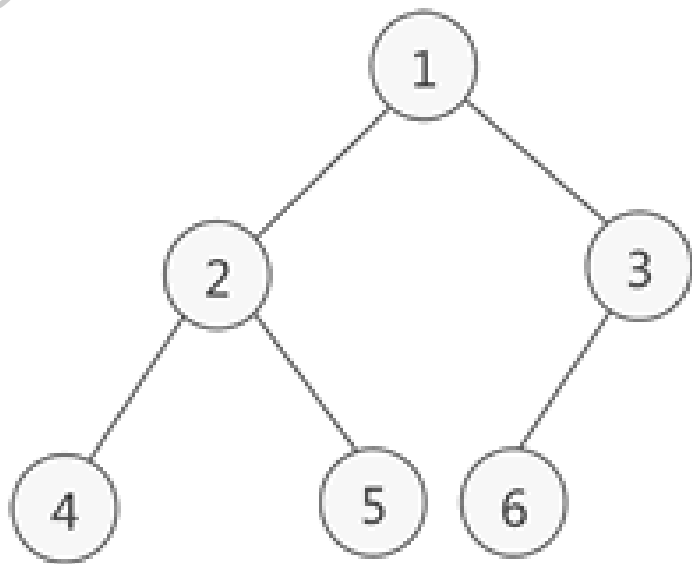
(n 为树的层数)



二叉树类型—完全二叉树

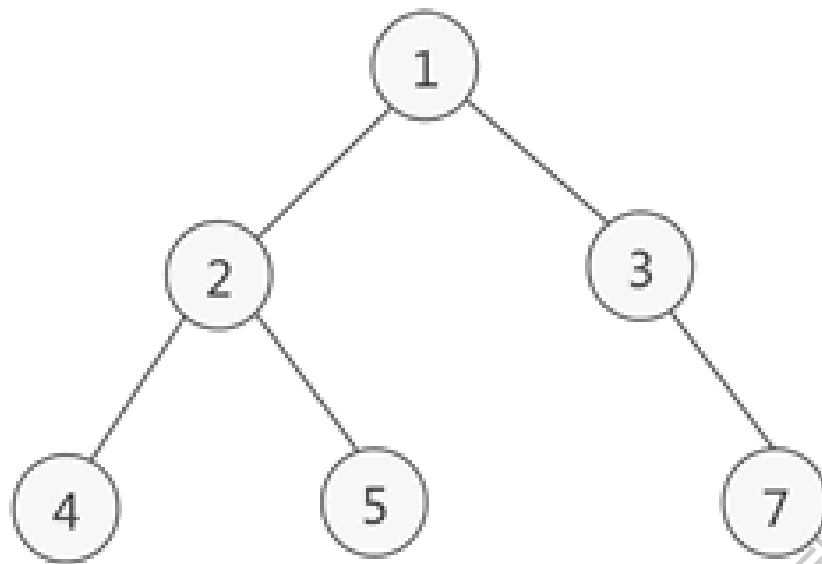


如果二叉树中除去最后一层节点为满二叉树，且最后一层的结点依次从左到右分布，则此二叉树被称为完全二叉树。



a) 完全二叉树

n 个结点的完全二叉树的层数



b) 非完全二叉树

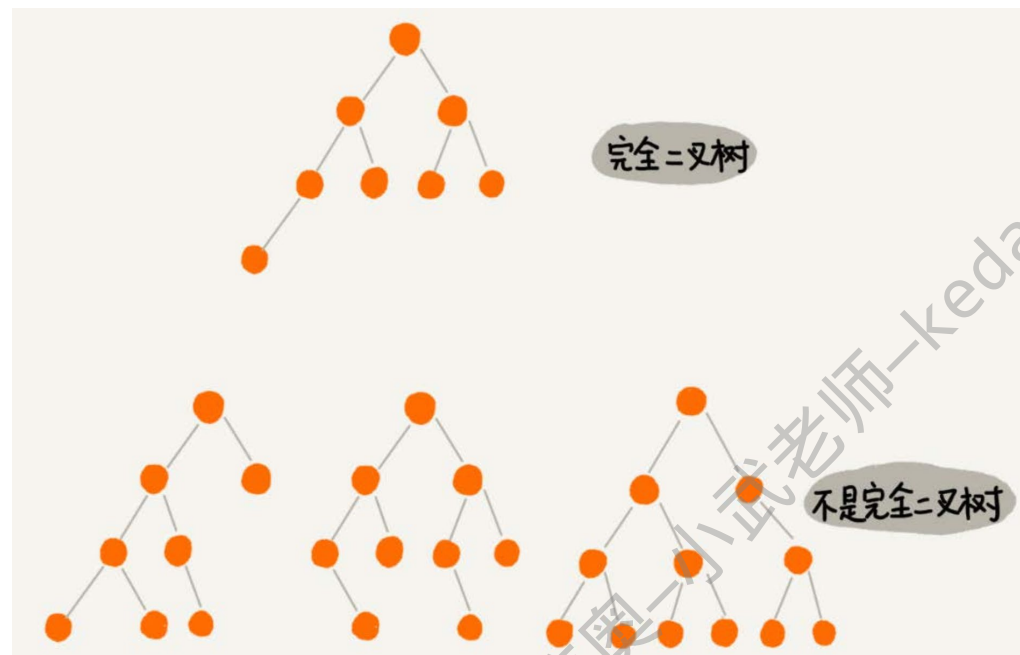
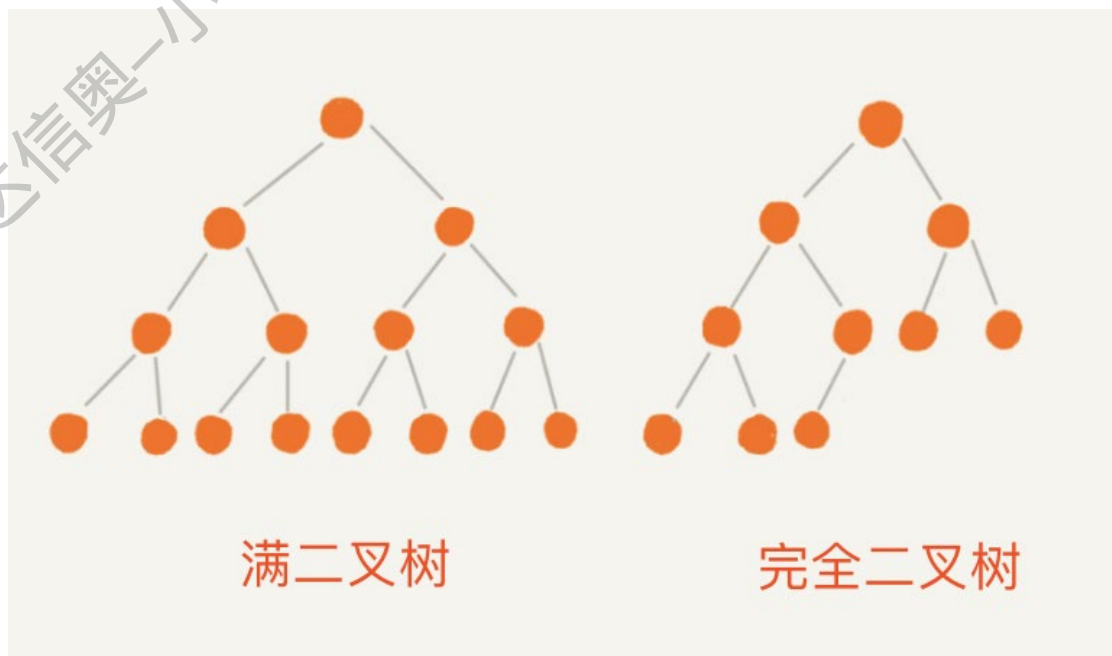
$\lfloor \log_2 n \rfloor + 1$



二叉树类型判断



如果二叉树中除了叶子结点，每个节点都有左右两个子节点，则此二叉树称为满二叉树。叶子结点全都在最底层。



满二叉树也是完全二叉树

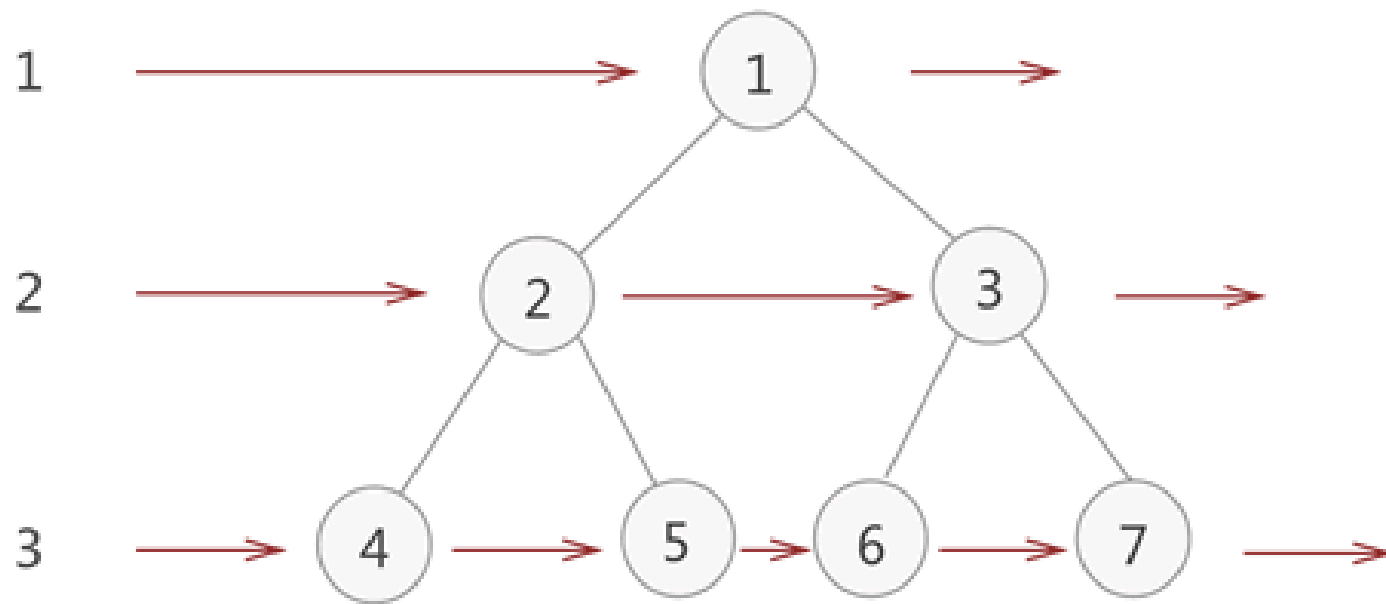


二叉树遍历



层次遍历

遍历二叉树可以算作是对树存储结构做的最多的操作。



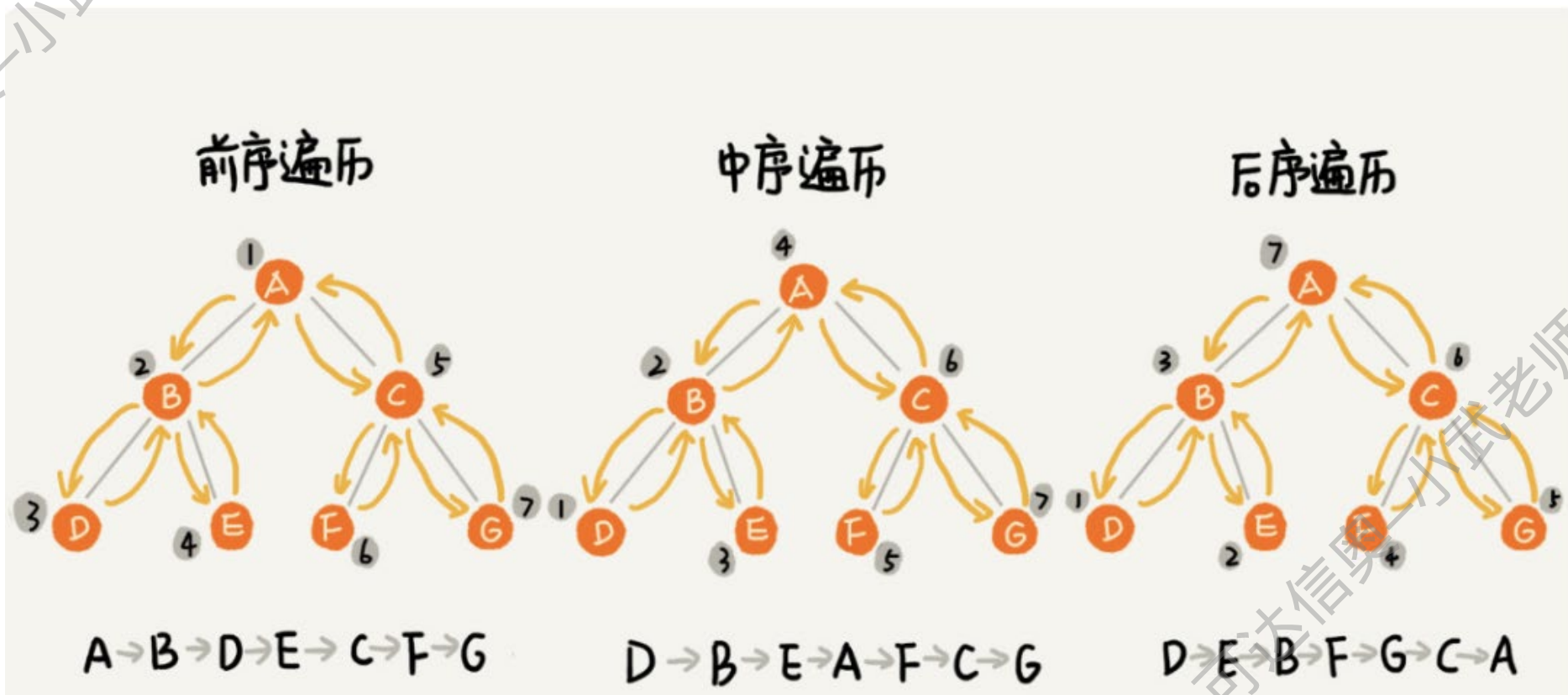
层次遍历二叉树示意图



二叉树遍历

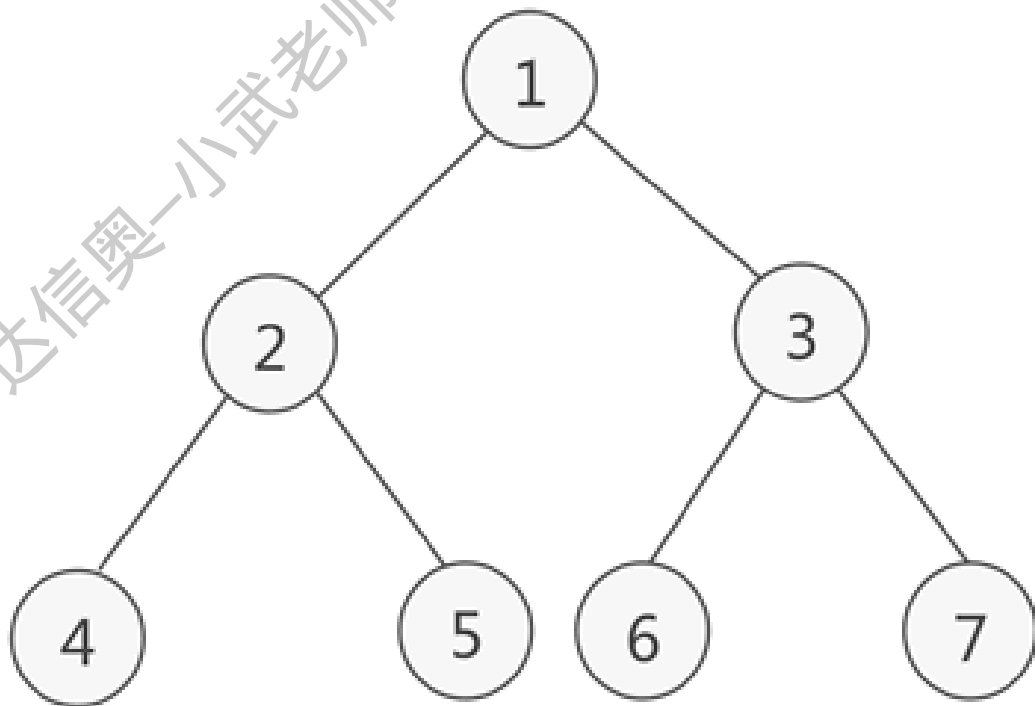


遍历一棵树，有经典三种方法：**前序遍历**、**中序遍历**、**后序遍历**，这里的序指的是父节点的遍历顺序，前序就是先遍历父节点，中序就是中间遍历父节点，后续就是最后遍历父节点。二叉树的前序、中序、后序遍历，其实是一个递归的过程。





二叉树遍历



$O(n)$

先序遍历得到的序列为：1 2 4 5 3 6 7

根

左子树

右子树

中序遍历得到的序列为：4 2 5 1 6 3 7

左子树

根

右子树

后序遍历得到的序列为：4 5 2 6 7 3 1

左子树

右子树

根

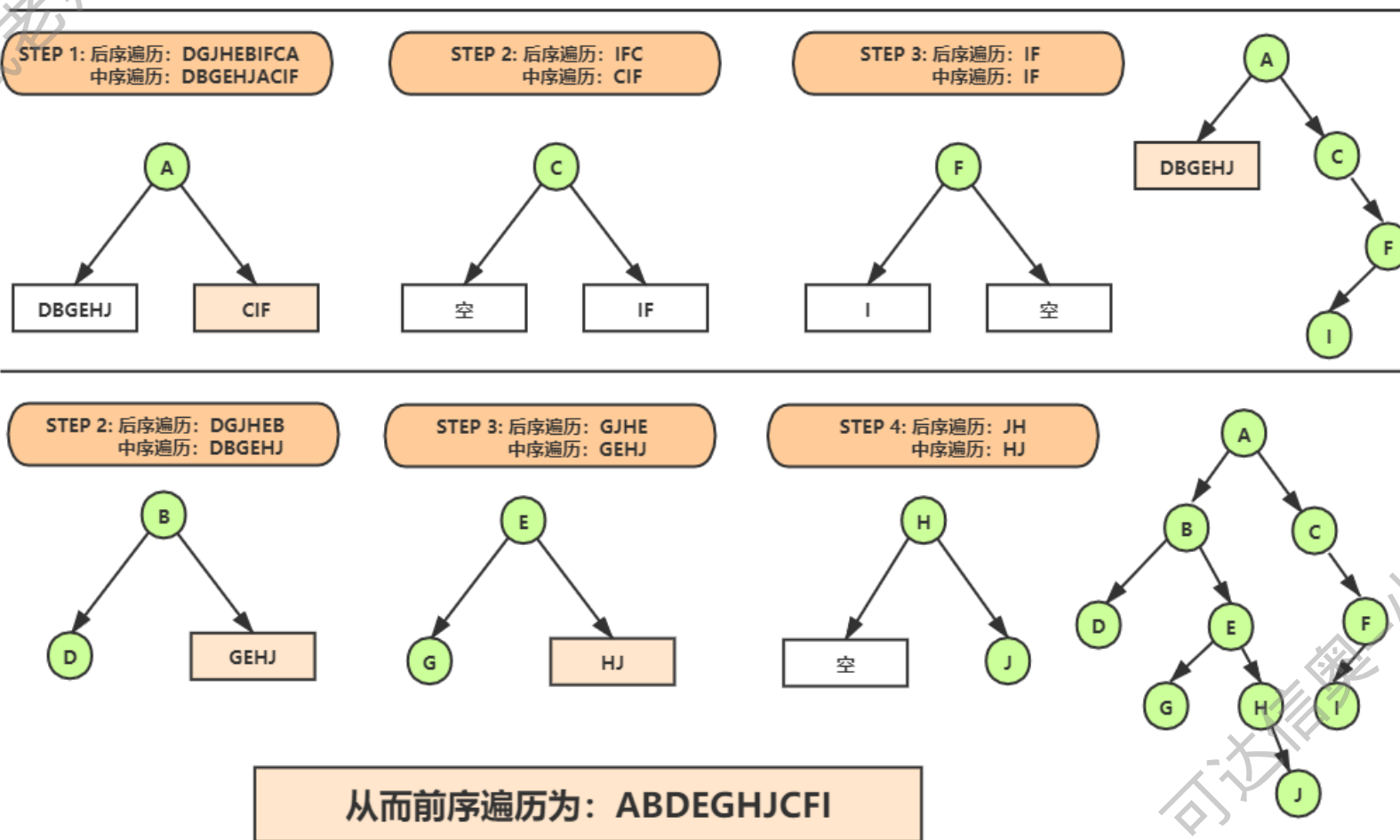


二叉树遍历



例1：已知中序后序，求前序

假设一棵二叉树的后序遍历序列为DGJHEBIFCA，中序遍历序列为DBGEHJACIF，则其前序遍历序列为：





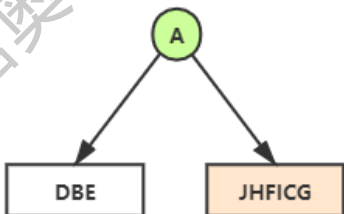
二叉树遍历



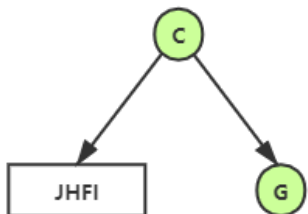
例2：已知中序前序，求后序

假设一棵二叉树的先序遍历序列为ABDECFHJIG, 中序遍历序列为DBEAJHFICG, 则其后序遍历序列为?

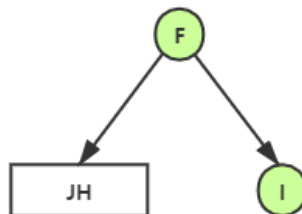
STEP 1: 前序遍历: ABDECFHJIG
中序遍历: DBEAJHFICG



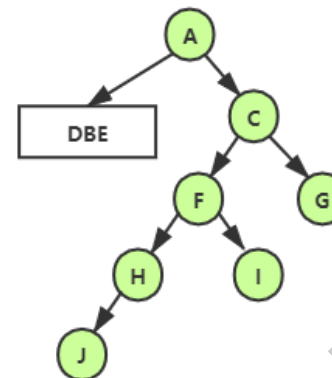
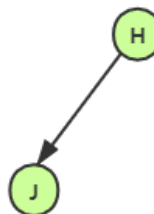
STEP 2: 前序遍历: CFHJIG
中序遍历: JHFICG



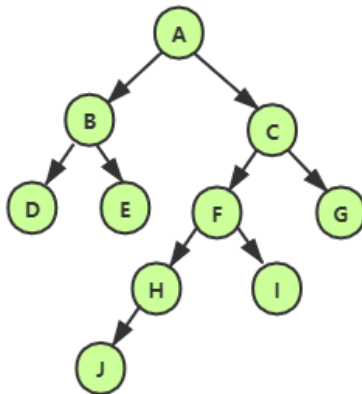
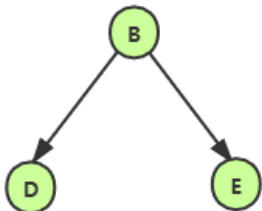
STEP 3: 前序遍历: FHJI
中序遍历: JHFI



STEP 3: 前序遍历: HJ
中序遍历: JH



STEP 2: 前序遍历: BDE
中序遍历: DBE



从而后序遍历为: DEBJHIFGCA
深度: 5



二叉树遍历



任务一：前序遍历序列为ABDECF，中序遍历序列为DBE AFC，则其后序遍历序列为？

DEBFCA

任务二：中序遍历输出结果为:[6,5,4,1,9,3,0] 后序遍历输出结果为:[6,5,9,1,0,3,4]，求前序？

[4,5,6,3,1,9,0]

任务三：某二叉树的中序遍历序列为CBADE，后序遍历序列为CBEDA，则前序遍历序列？

如果知道前序后序，能否确定中序？



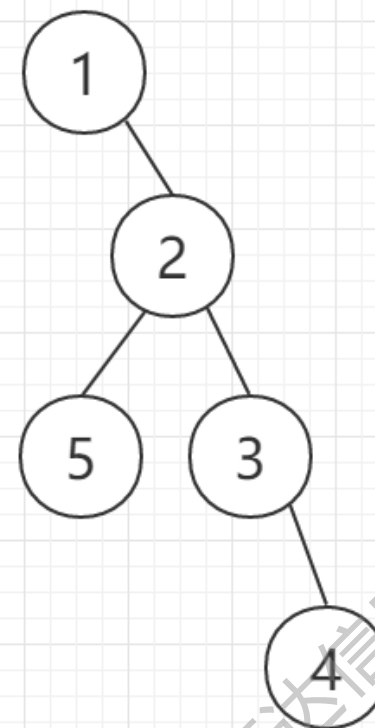
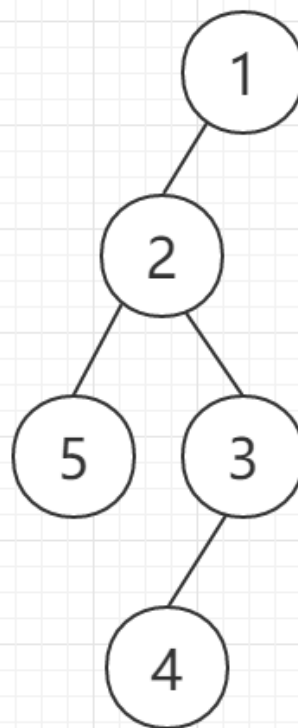
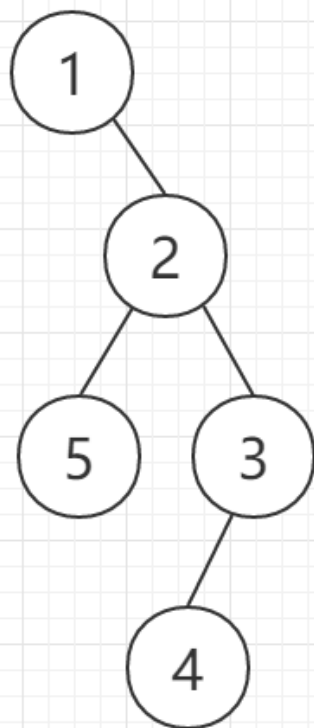
二叉树遍历



如果知道前序后序，能否确定中序？

•前序：1、2、5、3、4

后序：5、4、3、2、1



不止三种



真题解析



【2016普及组问题求解】约定二叉树的根节点高度为 1。一棵结点数为 2016 的二叉树最少有 1 个叶子结点；一棵结点数为 2016 的二叉树最小的高度值是 11。

$$2^{11}-1=2047$$

【2015提高组选择题】如果根的高度为 1，具有 61个结点的完全二叉树的高度为（ 6 ）。

高度为6的满二叉树有 2^6-1 个结点

【2015普及组问题求解】一棵结点数为 2015 的二叉树最多有 1008 个叶子结点。

高度为11的满二叉树有 $2^{11}-1=2047$ 个节点，高度为10的满二叉树有 $2^{10}-1=1023$ 个节点 $2015-1023=992$ ，可以得出第11层有992个叶子节点，然后第10层有 $512-992/2=16$ 个叶子节点。所以总的叶子节点数为 $992+16=1008$ 个



真题解析



【2019CSP-J选择题】假设一棵二叉树的后序遍历序列为DGJHEBIFCA，中序遍历序列为DBGEHJACIF，则其前序遍历序列为（ ）。

ABDEGHJCFI

【2007普及组选择题】已知7个结点的二叉树的先根遍历是1 2 4 5 6 3 7，中根遍历是4 2 6 5 1 7 3，则该二叉树的后根遍历是（ ）。

4 6 5 2 7 3 1

【2018提高组选择题】表达式 $a * d - b * c$ 的前缀形式是（ ）。

$- * a d * b c$

【2017提高组选择题】表达式 $a * (b + c) * d$ 的后缀形式是（ ）。

$a b c + * d *$

真题分类解析

阅读程序、基础语法、算法复杂度分析、模拟与枚举



课后习题与实验

Talk is cheap, show me the code !



声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

下节课见啦！

