Data Structures

Ch5

树和二叉树

Trees & Binary Trees

2024 年 10 月 11日

学而不厌 诲人不倦

Chapter 5　树和二叉树

5.1 引言

5.1 引言

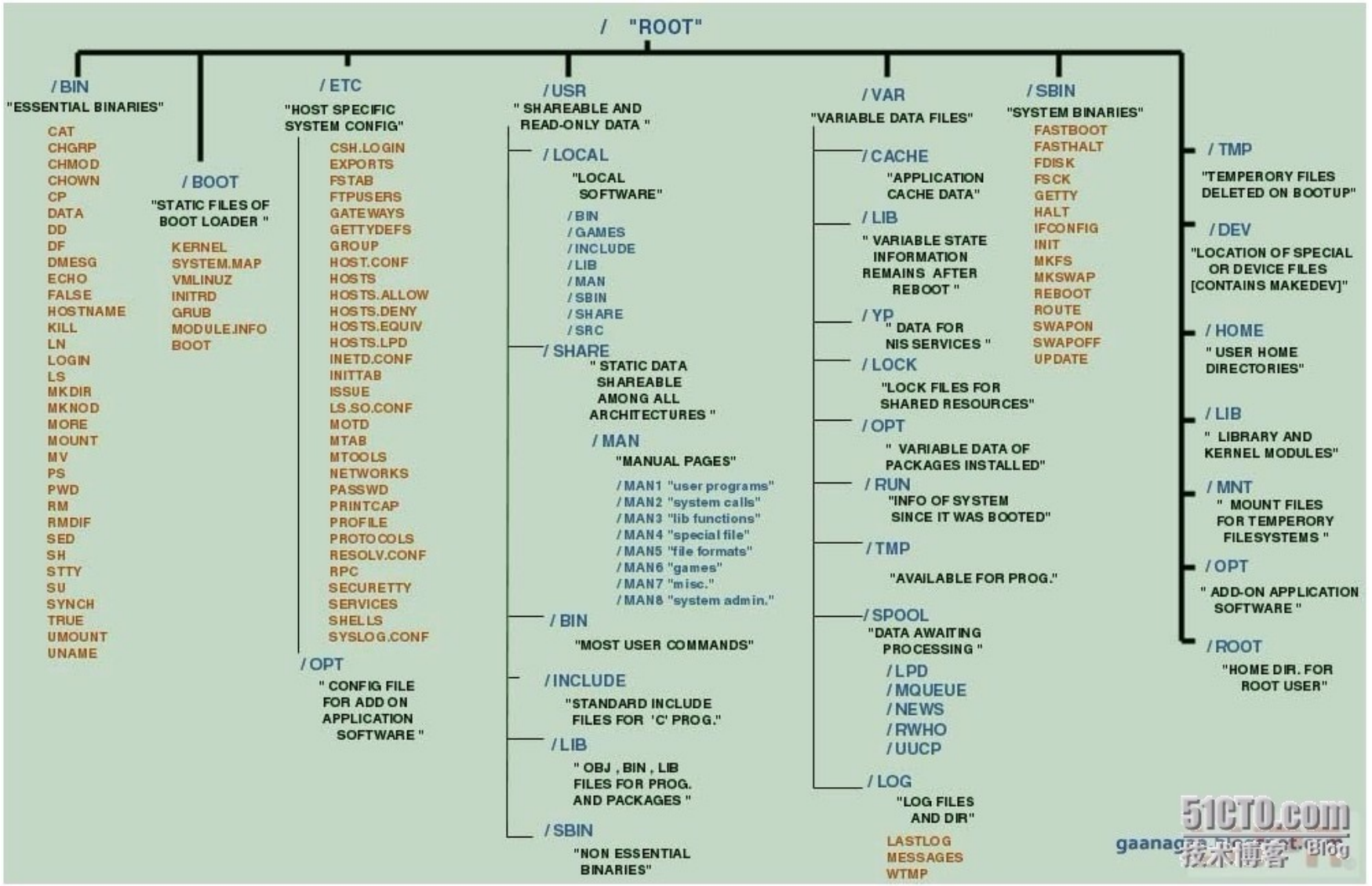
树

Not see the forest for the trees

See both the trees and the forest

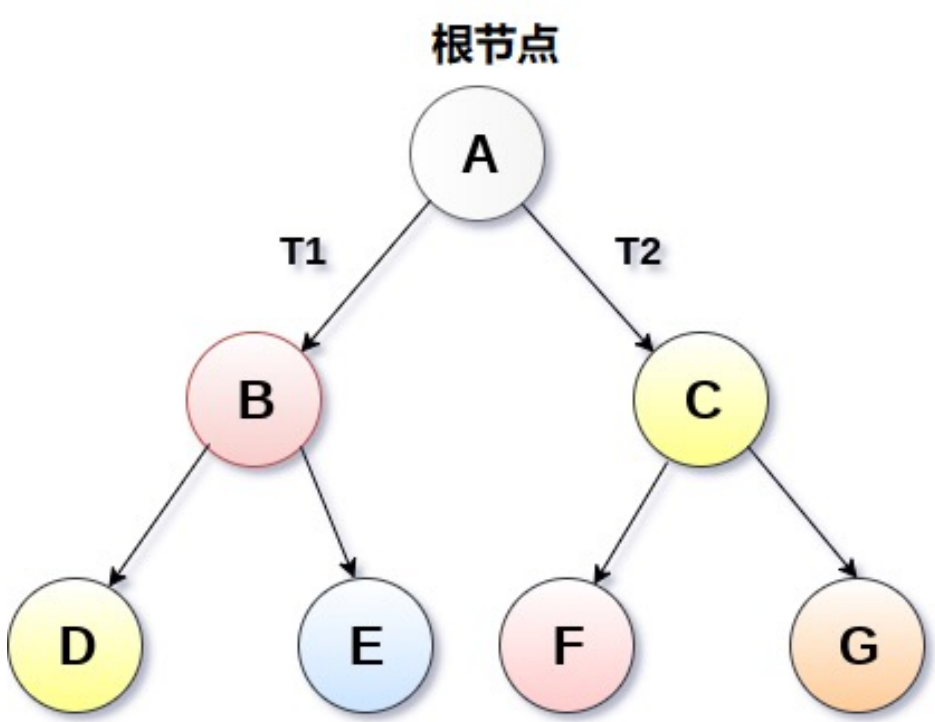
5.1 引言

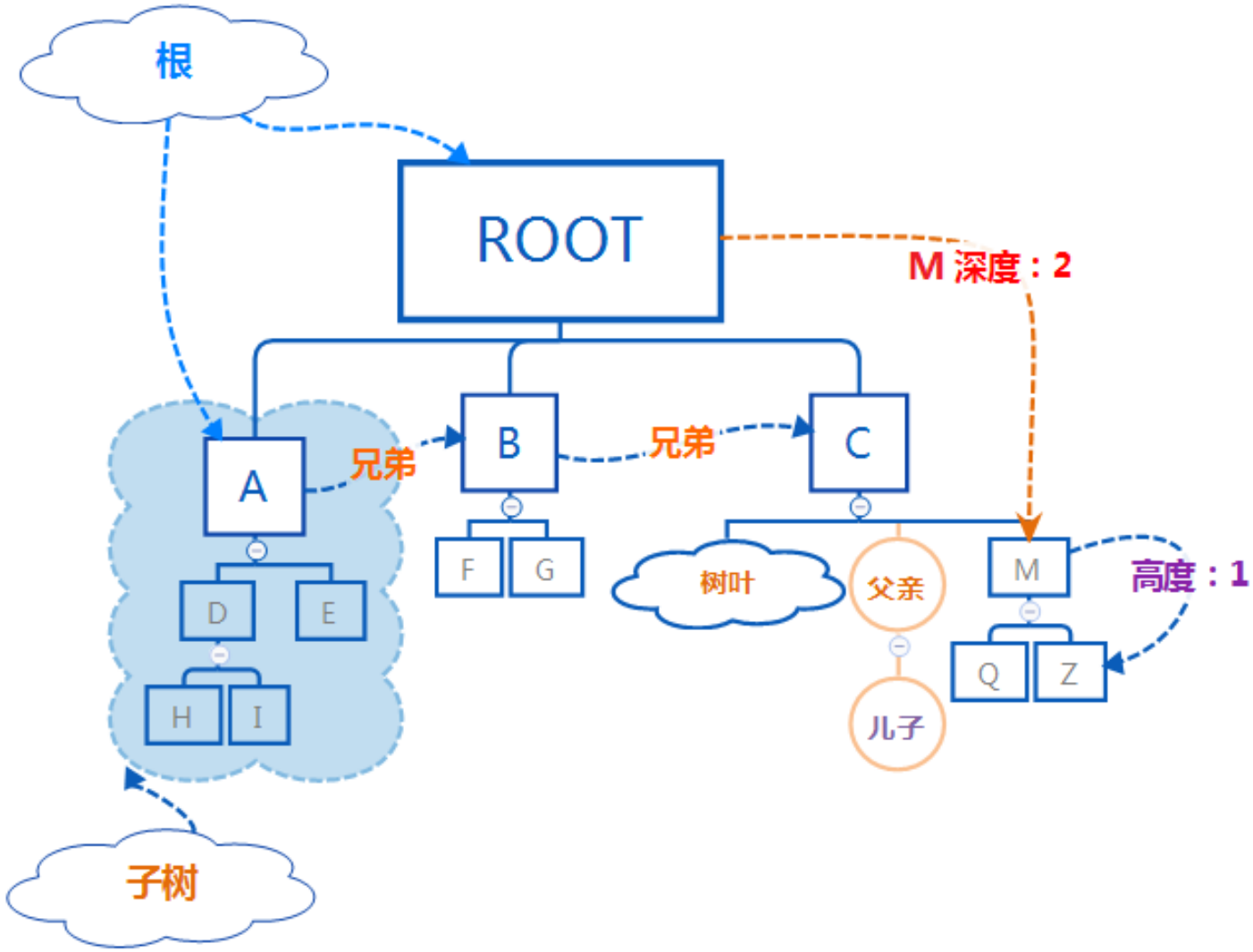
树


/ "ROOT"
/ BIN
"ESSENTIAL BINARIES"
CAT
CHGRP
CHMOD
CHOWN
CP
DATA
DD
DF
DMESG
ECHO
FALSE
HOSTNAME
KILL
LN
LOGIN
LS
MKDIR
MKNOD
MORE
MOUNT
MV
PS
PWD
RM
RMDIF
SED
SH
STTY
SU
SYNCH
TRUE
UMOUNT
UNAME
/ BOOT
"STATIC FILES OF BOOT LOADER "
KERNEL
SYSTEM.MAP
VMLINUZ
INITRD
GRUB
MODULE.INFO
BOOT
/ ETC
"HOST SPECIFIC SYSTEM CONFIG"
CSH.LOGIN
EXPORTS
FSTAB
FTPUSERS
GATEWAYS
GETTYDEFS
GROUP
HOST.CONF
HOSTS
HOSTS.ALLOW
HOSTS.DENY
HOSTS.EQUIV
HOSTS.LPD
INETD.CONF
INITTAB
ISSUE
LS.SO.CONF
MOTD
MTAB
MTOOLS
NETWORKS
PASSWD
PRINTCAP
PROFILE
PROTOCOLS
RESOLV.CONF
RPC
SECURETTY
SERVICES
SHELLS
SYSLOG.CONF
/ OPT
" CONFIG FILE FOR ADD ON APPLICATION SOFTWARE "
/ USR
" SHAREABLE AND READ-ONLY DATA "
/ LOCAL
"LOCAL SOFTWARE"
/ BIN
/ GAMES
/ INCLUDE
/ LIB
/ MAN
/ SBIN
/ SHARE
/ SRC
/ SHARE
" STATIC DATA SHAREABLE AMONG ALL ARCHITECTURES "
/ MAN
"MANUAL PAGES"
/ MAN1 "user programs"
/ MAN2 "system calls"
/ MAN3 "lib functions"
/ MAN4 "special file"
/ MAN5 "file formats"
/ MAN6 "games"
/ MAN7 "misc."
/ MAN8 "system admin."
/ BIN
"MOST USER COMMANDS"
/ INCLUDE
"STANDARD INCLUDE FILES FOR 'C' PROG."
/ LIB
" OBJ , BIN , LIB FILES FOR PROG. AND PACKAGES "
/ SBIN
"NON ESSENTIAL BINARIES"
/ VAR
"VARIABLE DATA FILES"
/ CACHE
"APPLICATION CACHE DATA"
/ LIB
" VARIABLE STATE INFORMATION REMAINS AFTER REBOOT "
/ YP
" DATA FOR NIS SERVICES "
/ LOCK
"LOCK FILES FOR SHARED RESOURCES"
/ OPT
" VARIABLE DATA OF PACKAGES INSTALLED"
/ RUN
"INFO OF SYSTEM SINCE IT WAS BOOTED"
/ TMP
"AVAILABLE FOR PROG."
/ SPOOL
"DATA AWAITING PROCESSING "
/ LPD
/ MQUEUE
/ NEWS
/ RWHO
/ UUCP
/ LOG
"LOG FILES AND DIR"
LASTLOG
MESSAGES
WTMP
/ SBIN
"SYSTEM BINARIES"
FASTBOOT
FASTHALT
FDISK
FSCK
GETTY
HALT
IFCONFIG
INIT
MKFS
MKSWAP
REBOOT
ROUTE
SWAPON
SWAPOFF
UPDATE
/ TMP
"TEMPERORY FILES DELETED ON BOOTUP"
/ DEV
"LOCATION OF SPECIAL OR DEVICE FILES [CONTAINS MAKEDEV]"
/ HOME
" USER HOME DIRECTORIES"
/ LIB
" LIBRARY AND KERNEL MODULES"
/ MNT
" MOUNT FILES FOR TEMPERORY FILESYSTEMS "
/ OPT
" ADD-ON APPLICATION SOFTWARE "
/ ROOT
"HOME DIR. FOR ROOT USER"

5.1 引言

树


根节点
A
T1
T2
B
C
D
E
F
G


根
ROOT
M 深度：2
A
兄弟
B
兄弟
C
D
E
F
G
H
I
树叶
父亲
M
高度：1
Q
Z
儿子
子树

5.2 树的逻辑结构

5-2-1　树的定义和基本术语

1. 树的定义

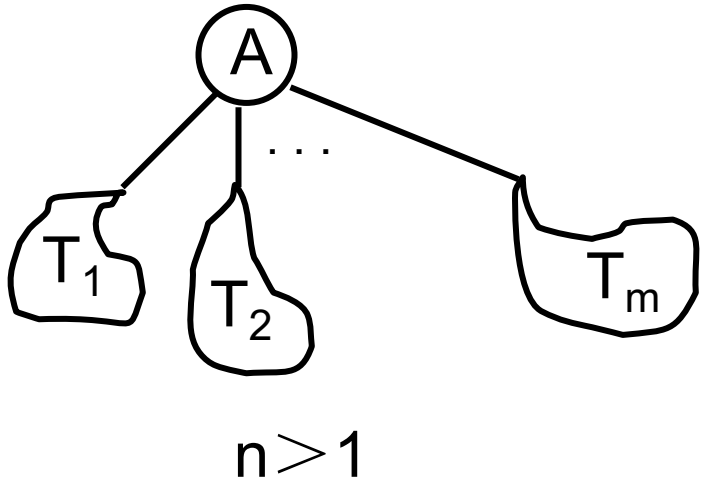
树(Tree)：是具有层次结构的 n(n≥0) 个结点的有限集。

n = 0，空树 。

n > 0，有且仅有一个称为根的结点。

n > 1，除根结点外，其余结点可分为 m(m > 0) 个互不相交的有限子集 $T_1, T_2, \ldots T_m$，其中每个子集都称为根结点的子树 。

只有根结点的树



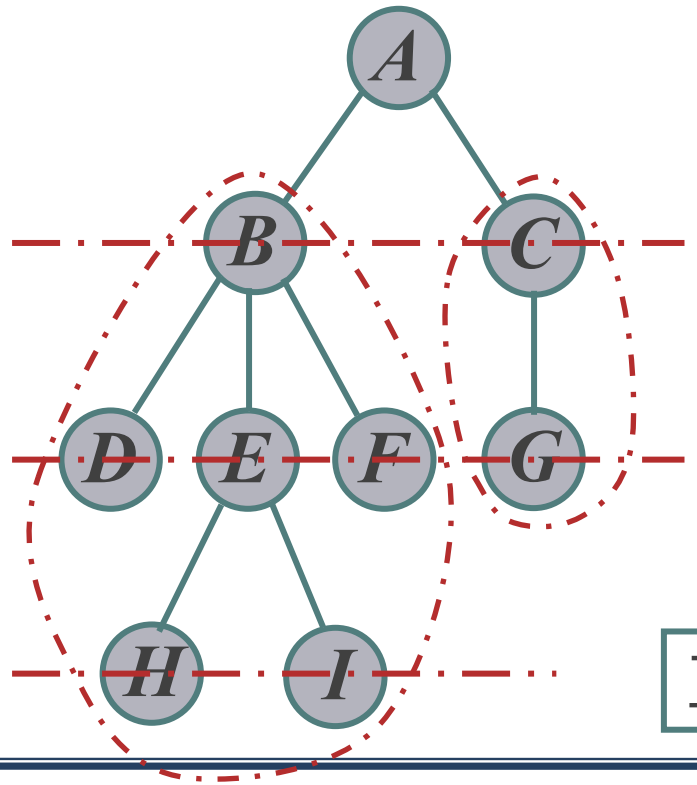
n＞1

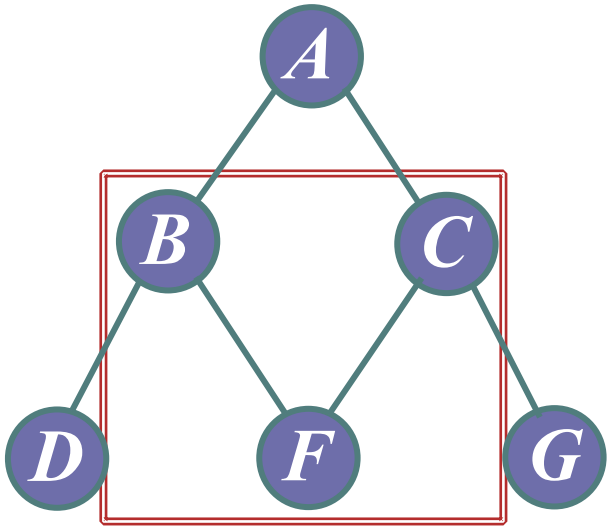
树的定义是采用递归方法

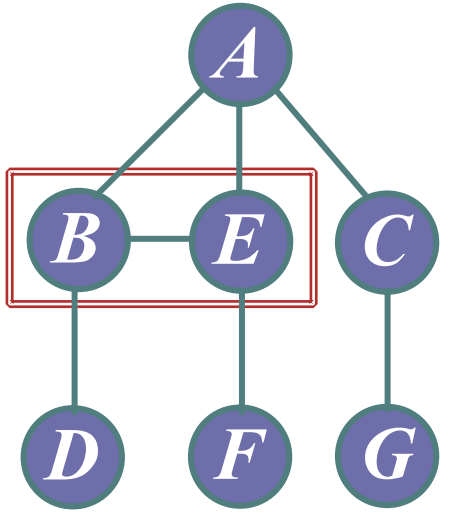
2. 树的逻辑特征

互不相交的具体含义是什么？

- 结点：结点不能属于多个子树
- 边：子树之间不能有关系





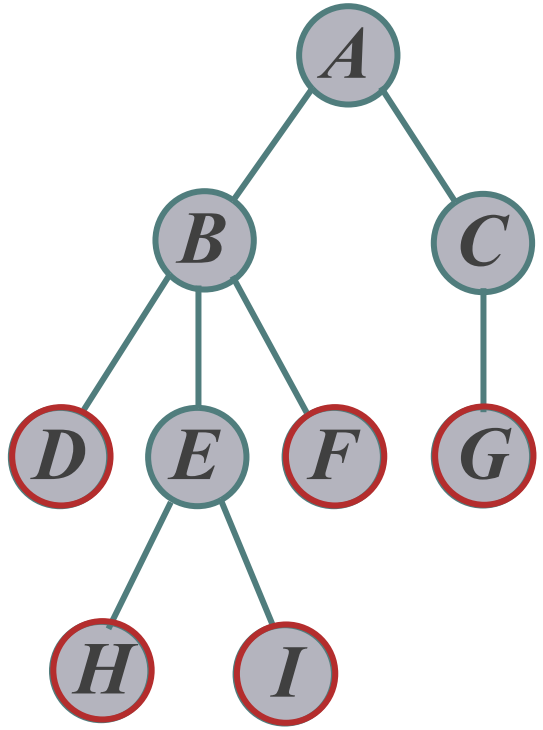


互不相交 ⇨ 没有回路 ⇨ 树结构具有层次性

3. 树的基本术语

- 结点的度：结点所拥有的子树的个数
- 树的度：树中各结点度的最大值
- 叶子结点：度为 0 的结点，也称为终端结点
- 分支结点：度不为 0 的结点，也称为非终端结点

除根结点外，其余分支结点也称为内部结点。



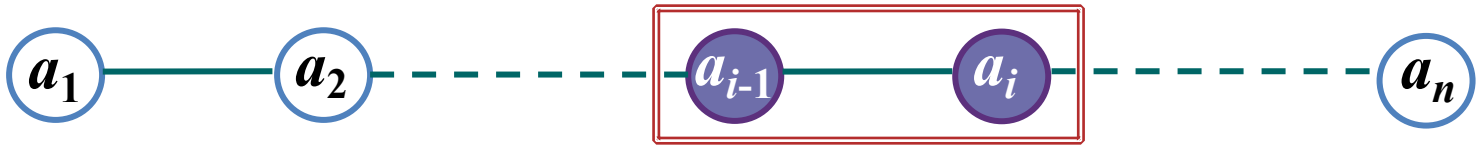
例，A 的度为 2 ， F 的度为 0 。

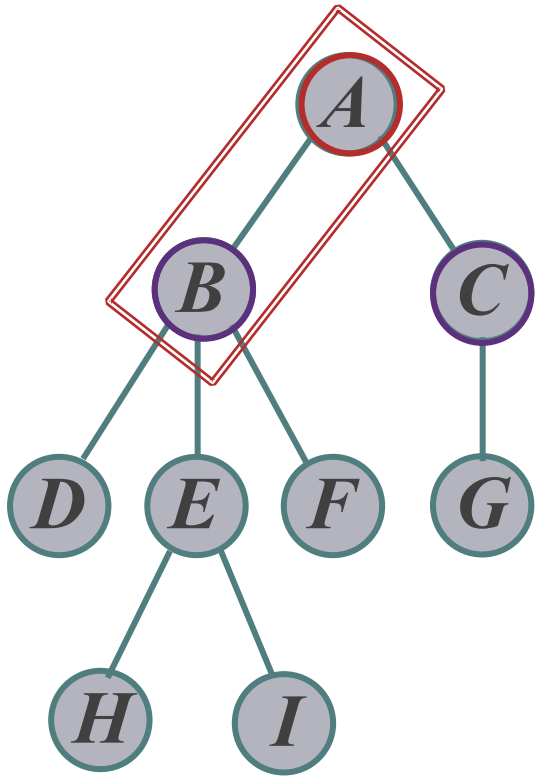
例，D,F,G,H,I 为叶子； A,B,C, E 为分支结点 。

例，B,C,E 为内部结点 。

3. 树的基本术语

- 孩子：树中某结点子树的根结点称为这个结点的孩子结点
- 双亲：这个结点称为它孩子结点的双亲结点
- 兄弟：具有同一个双亲的孩子结点互称为兄弟

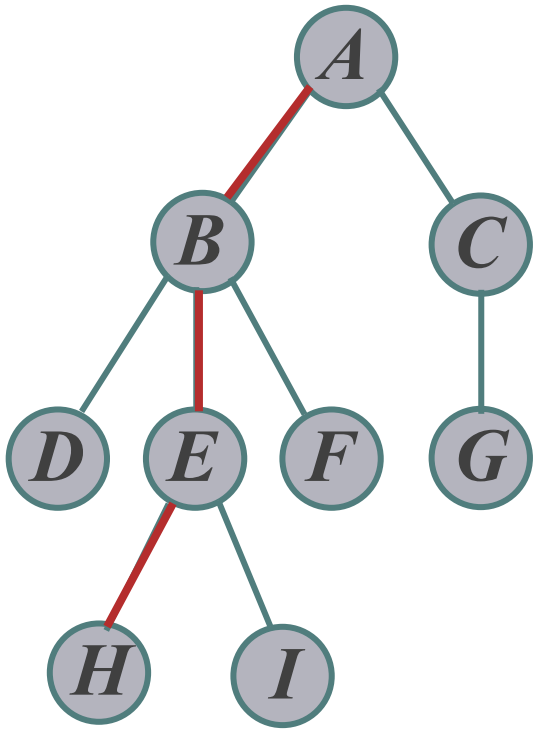




- 在线性结构中，逻辑关系表现为前驱——后继
- 在树结构中，逻辑关系表现为双亲——孩子

3. 树的基本术语

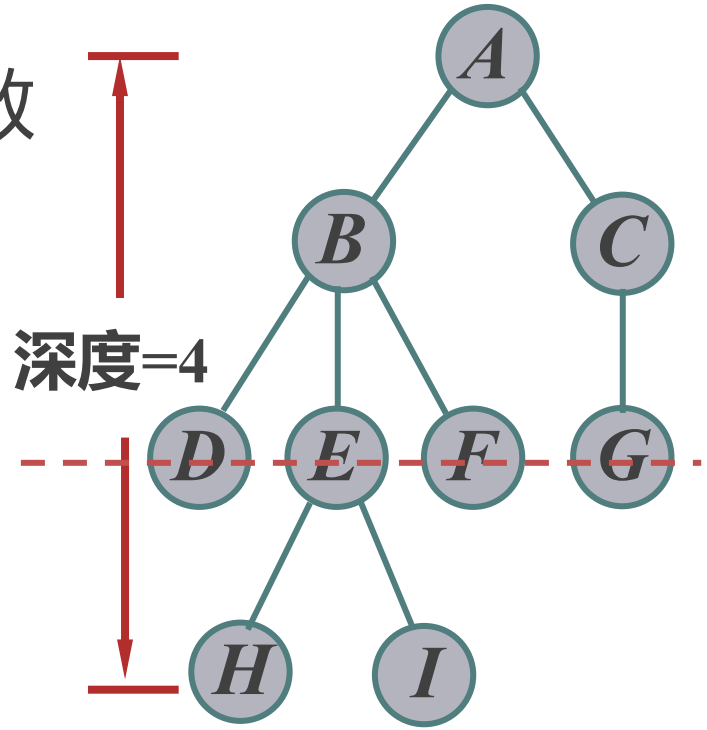
- 路径：结点序列 $n_1, n_2, \ldots, n_k$ 称为一条由 n_1 至 n_k 的路径，当且仅当满足如下关系：结点 n_i 是 n_{i+1} 的双亲（$1<=i<k$）
- 路径长度：路径上经过的边的个数
- 祖先、子孙：如果有一条路径从结点 x 到结点 y，则 x 称为 y 的祖先，而 y 称为 x 的子孙



- 在树结构中，路径是唯一的

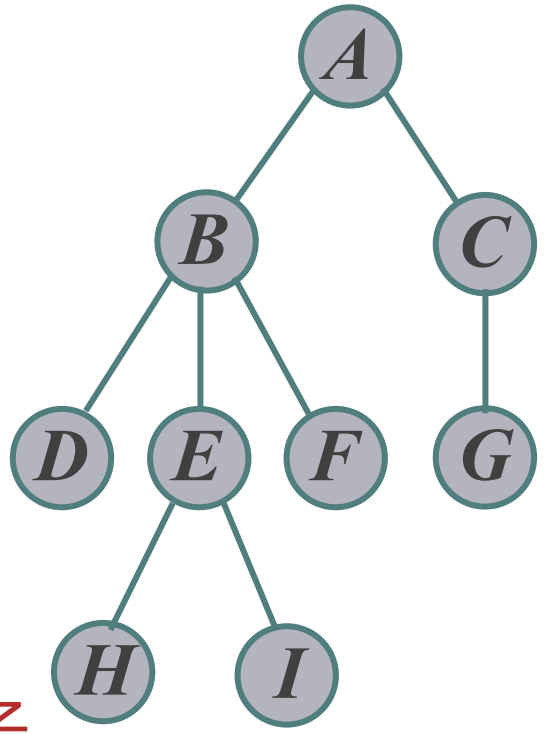
3. 树的基本术语

- 结点所在层数：根结点的层数为 1；对其余结点，若某结点在第 k 层，则其孩子结点在第 $k+1$ 层
- 树的深度（高度）：树中所有结点的最大层数
- 树的宽度：树中每一层结点个数的最大值



如果将树中结点的各子树看成从左到右是有序的(即不能互换)，则称该树为**有序树**，否则称为**无序树**。

4. 线性结构与树结构的比较



线性结构	树结构
开始结点（只有一个）:无前驱	根结点（只有一个）:无双亲
终端结点（只有一个）:无后继	叶子结点(可以有多个):无孩子
其它元素：一个前驱,一个后继	其它结点：一个双亲,多个孩子
一对一	**一对多**

5.2 树的逻辑结构

5-2-2　树的抽象数据类型定义

1. 树的抽象数据类型定义

ADT Tree
DataModel
　树由一个根结点和若干棵子树构成，树中结点具有层次关系
Operation
　InitTree：初始化一棵树
　DestroyTree：销毁一棵树
　PreOrder：前序遍历树
　PostOrder：后序遍历树
　LeverOrder：层序遍历树
endADT

简单起见，只讨论树的遍历

2. 树的遍历

什么是遍历？ 线性结构如何遍历？

简言之，遍历是对数据集合进行没有遗漏、没有重复的访问

树的遍历：从根结点出发，按照某种次序访问树中所有结点，并且每个结点仅被访问一次

前序（根）、后序（根）和层序（次）等

抽象操作，可以是对结点进行的各种处理，这里简化为输出结点的数据

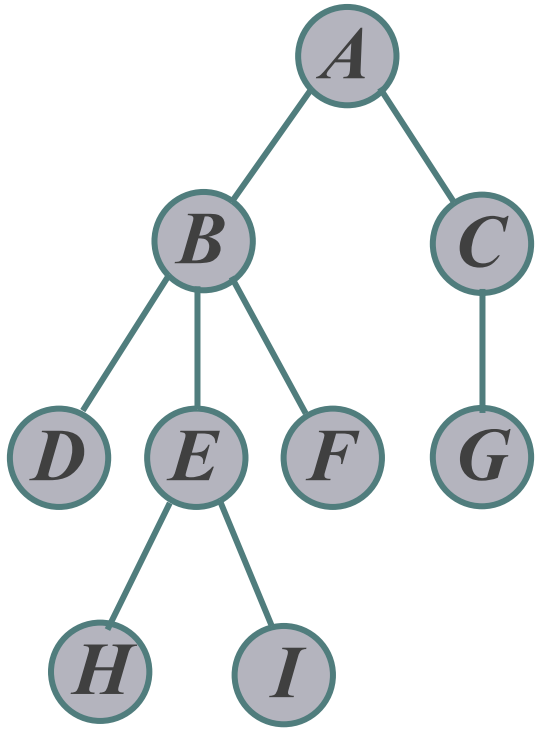
2. 树的遍历

树的前序遍历操作定义：

若树为空，则空操作返回；否则

（1）访问根结点

（2）从左到右前序遍历根结点的每一棵子树



A B D E H I F C G

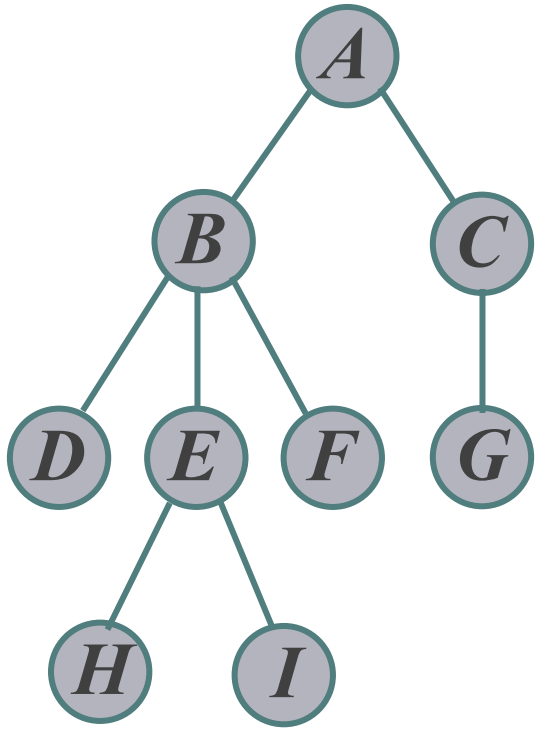
2. 树的遍历

树的后序遍历操作定义：

若树为空，则空操作返回；否则

（1）从左到右后序遍历根结点的每一棵子树

（2）访问根结点

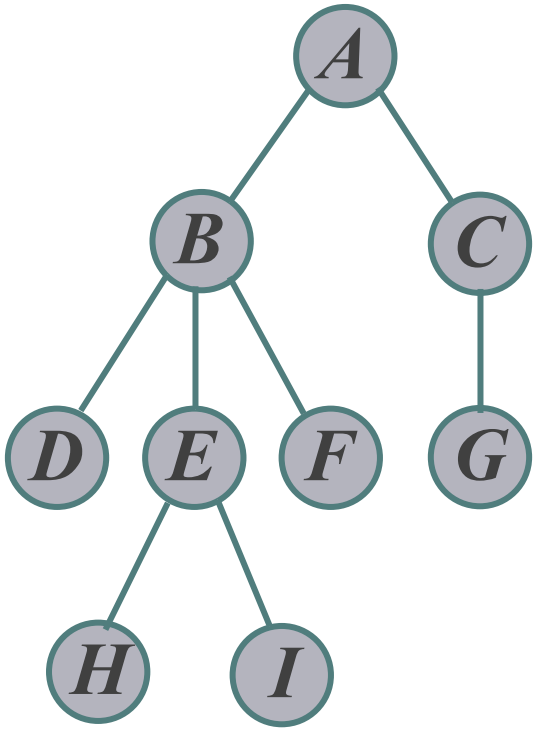


D H I E F B G C A

2. 树的遍历

树的层次遍历操作定义：

从树的根结点开始，自上而下逐层遍历，在同一层中，按从左到右的顺序对结点逐个访问



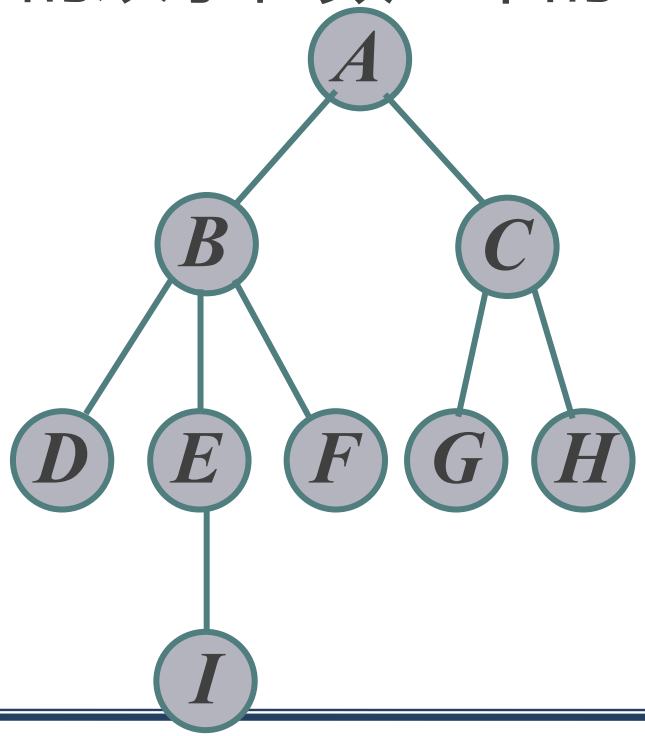
A B C D E F G H I

5.3 树的存储结构

5-3-1　树的双亲表示法

1. 树的双亲表示法

树的双亲表示法：用一维数组存储树中各个结点（一般按层序存储）的数据信息以及该结点的双亲在数组中的下标



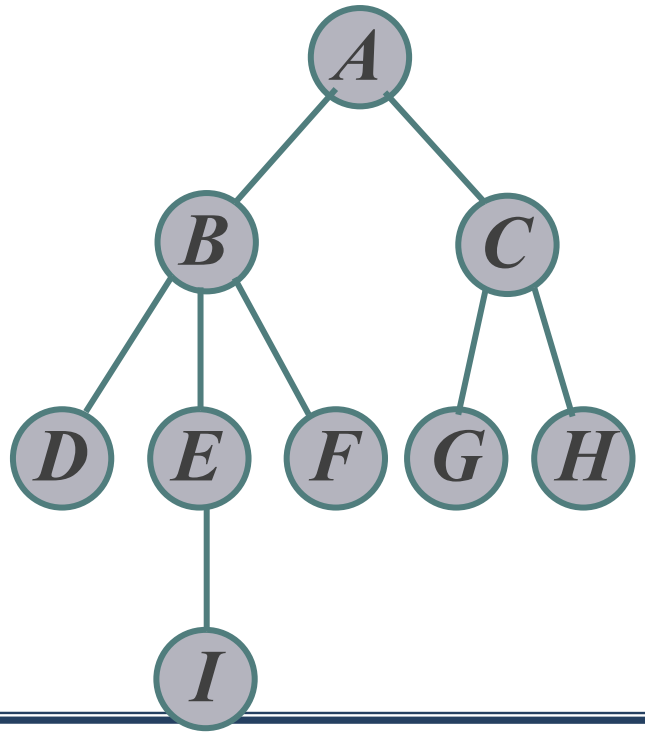
data	Parent

数据域　指示器，指向其父结点

	data	parent
0	A	−1
1	B	0
2	C	0
3	D	1
4	E	1
5	F	1
6	G	2
7	H	2
8	I	4

2. 树的双亲表示法性能分析

- 如何查找双亲结点？时间性能？ $O(1)$
- 如何查找孩子结点？时间性能？ $O(n)$



	data	parent	firstchild
0	A	−1	1
1	B	0	3
2	C	0	6
3	D	1	−1
4	E	1	8
5	F	1	−1
6	G	2	−1
7	H	2	−1
8	I	4	−1

3. 树的双亲表示法的实现

```
template <typename DataType>
struct PNode
{
    DataType data;  //数据域
    int parent;       //指示器，指向其父结点
}; // 定义树结点

# define  MAX_TREE_SIZE  100 //定义最大结点数

typedef   struct    {
    PNode  nodes[MAX_TREE_SIZE]; //顺序结构存储
    int       r，n；     //根的位置和结点总数
}  PTree ;      //定义树
```

	data	parent
0	A	−1
1	B	0
2	C	0
3	D	1
4	E	1
5	F	1
6	G	2
7	H	2
8	I	4

5.3 树的存储结构

5-3-2　树的孩子表示法

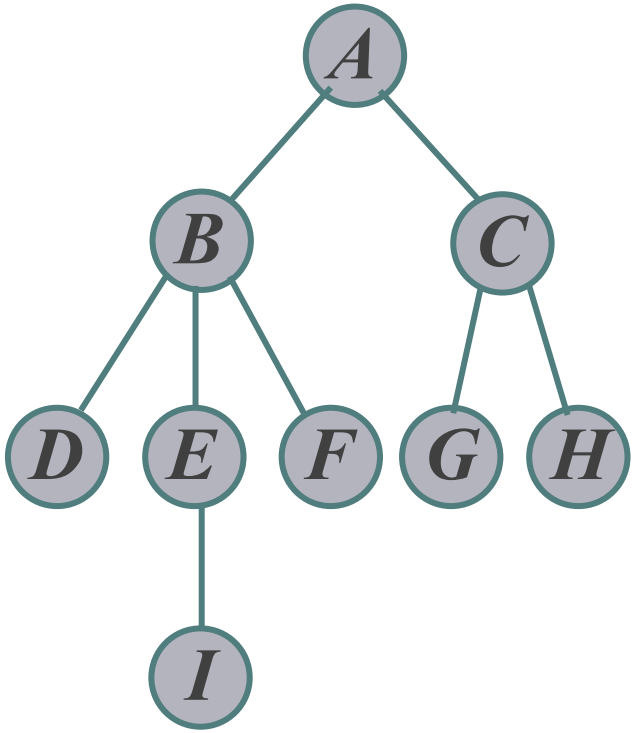
1. 树的孩子表示法

如何表示结点的孩子呢？ 方案一：指针域的个数等于树的度

data	child1	child2	……	childd

其中：data：数据域，存放该结点的数据信息

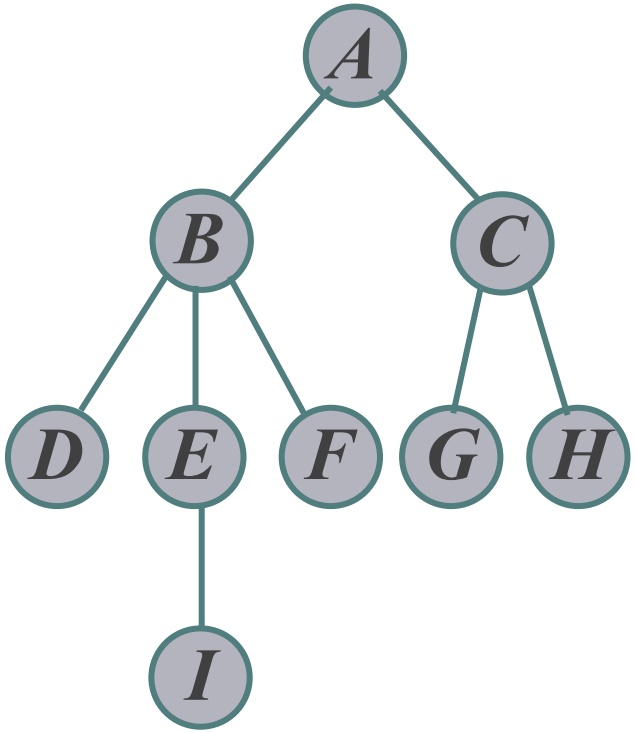
child1~childd：指针域，指向该结点的孩子

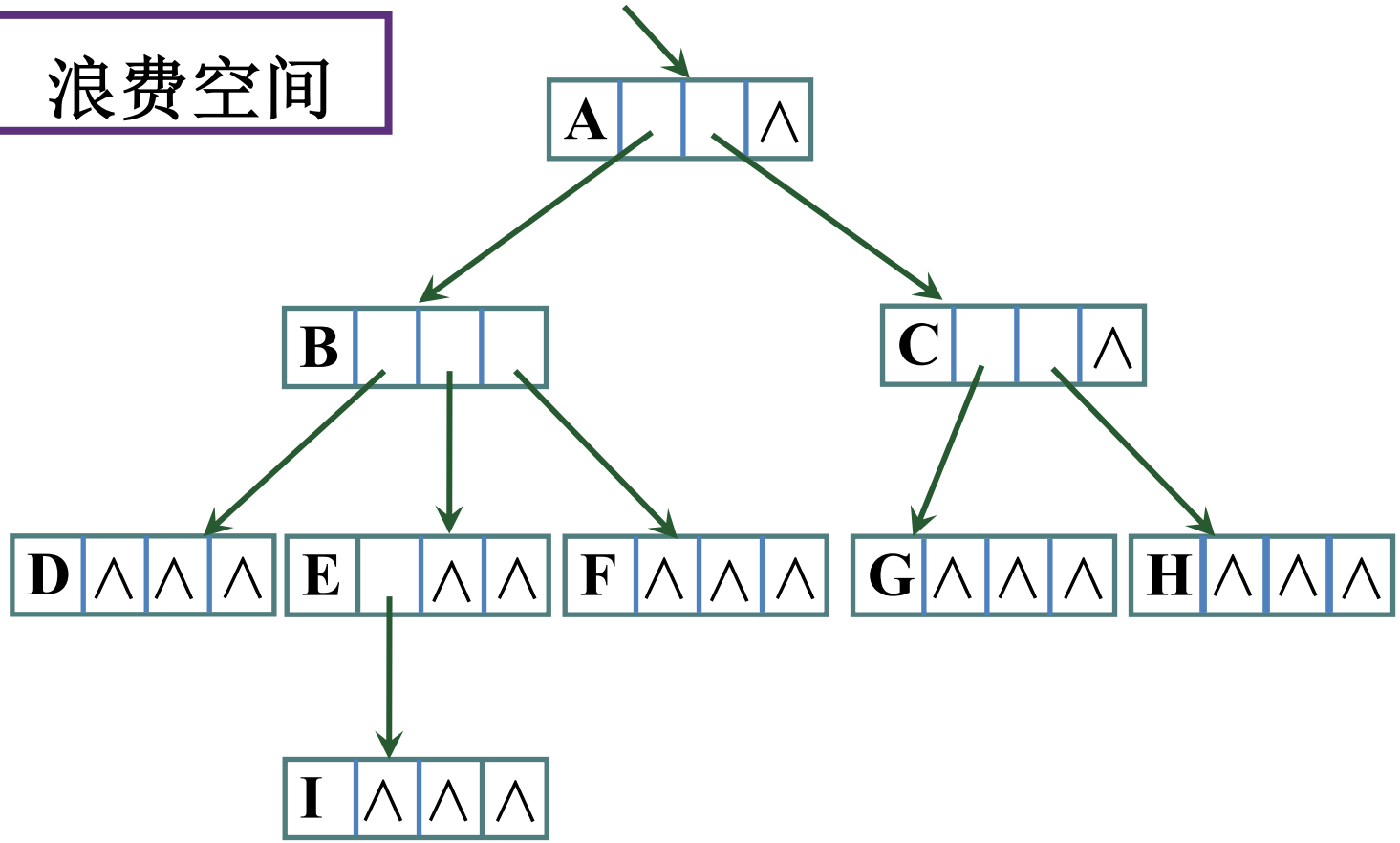


1. 树的孩子表示法

如何表示结点的孩子呢？ 方案一：指针域的个数等于树的度

缺点：浪费空间

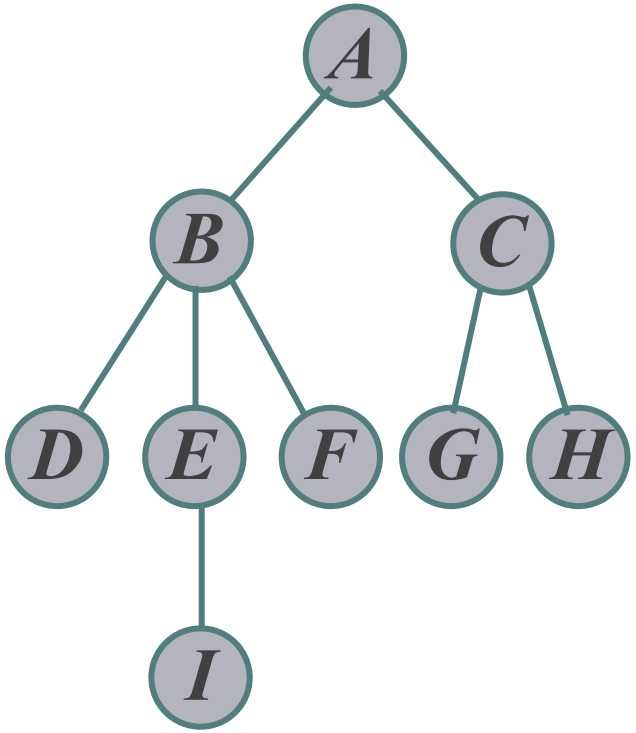




1. 树的孩子表示法

如何表示结点的孩子呢？ 方案二：指针域的个数等于该结点的度

data	degree	child1	child2	……	childd



其中：data：数据域，存放该结点的数据信息

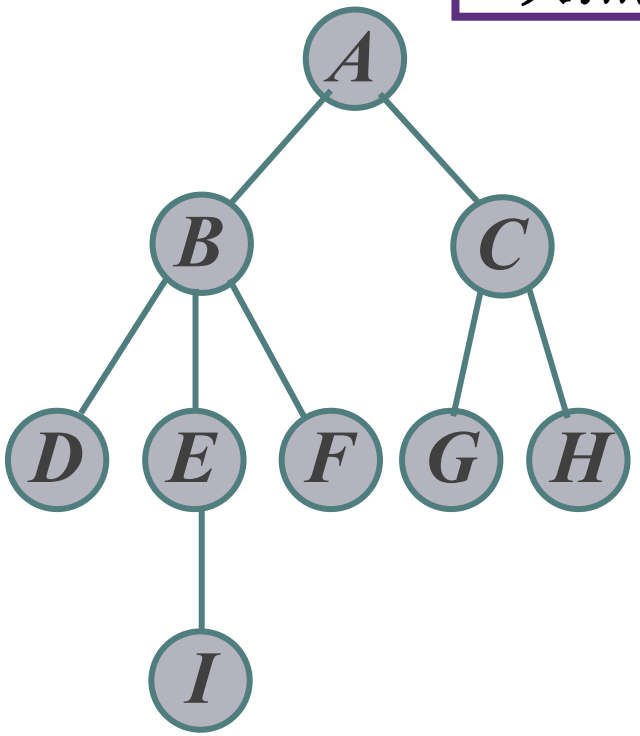
degree：数据域，存放该结点的度

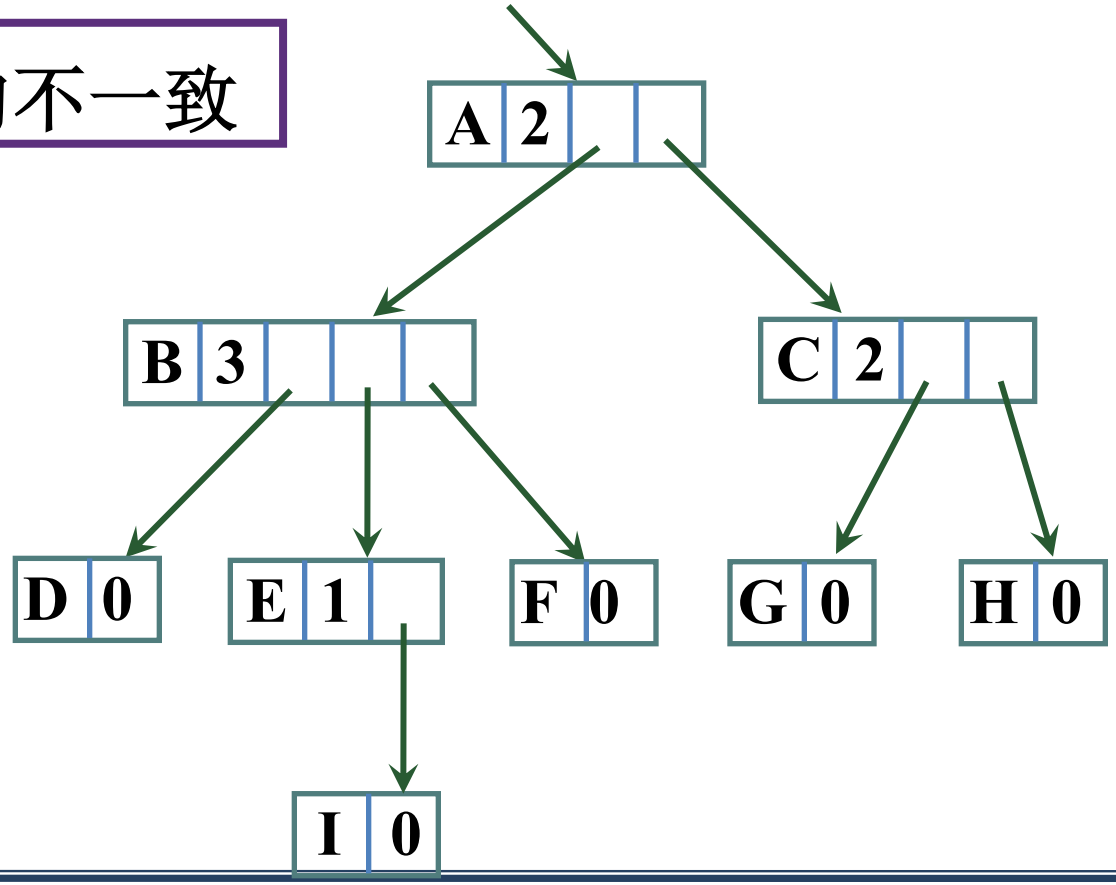
child1~childd：指针域，指向该结点的孩子

1. 树的孩子表示法

如何表示结点的孩子呢？ 方案二：指针域的个数等于该结点的度

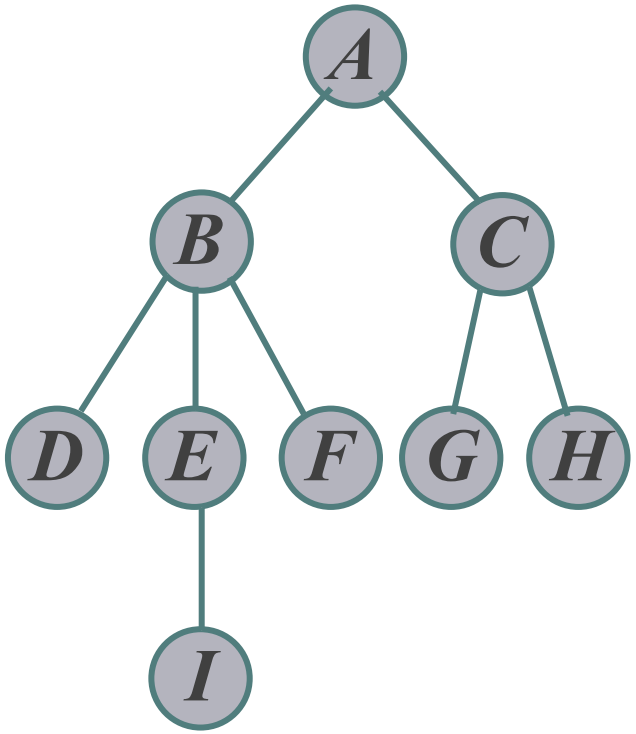
缺点：结点结构不一致

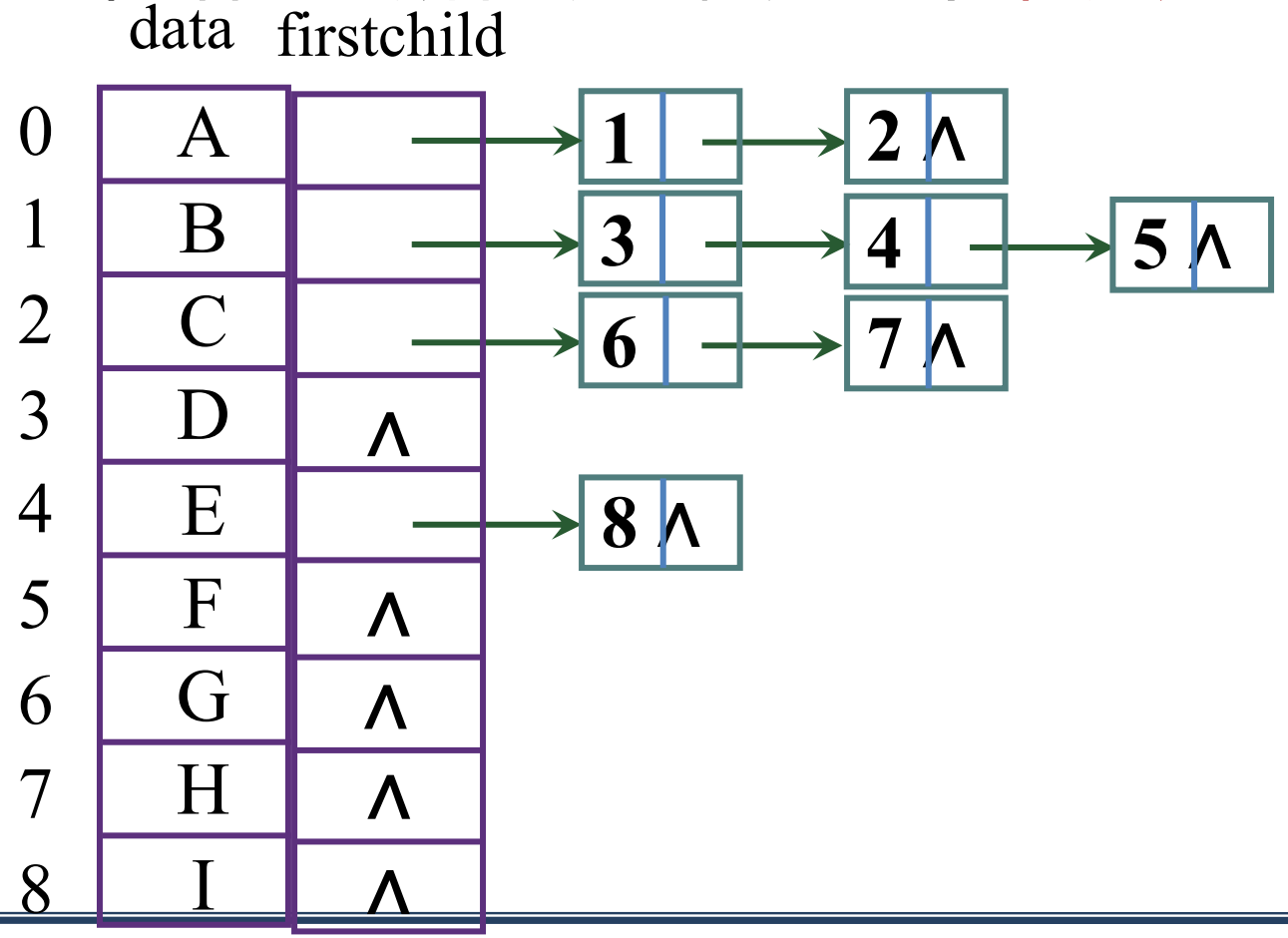




2. 树的孩子表示法的实现

如何表示结点的孩子呢？ 将结点的所有孩子构成一个单链表





2. 树的孩子表示法的实现

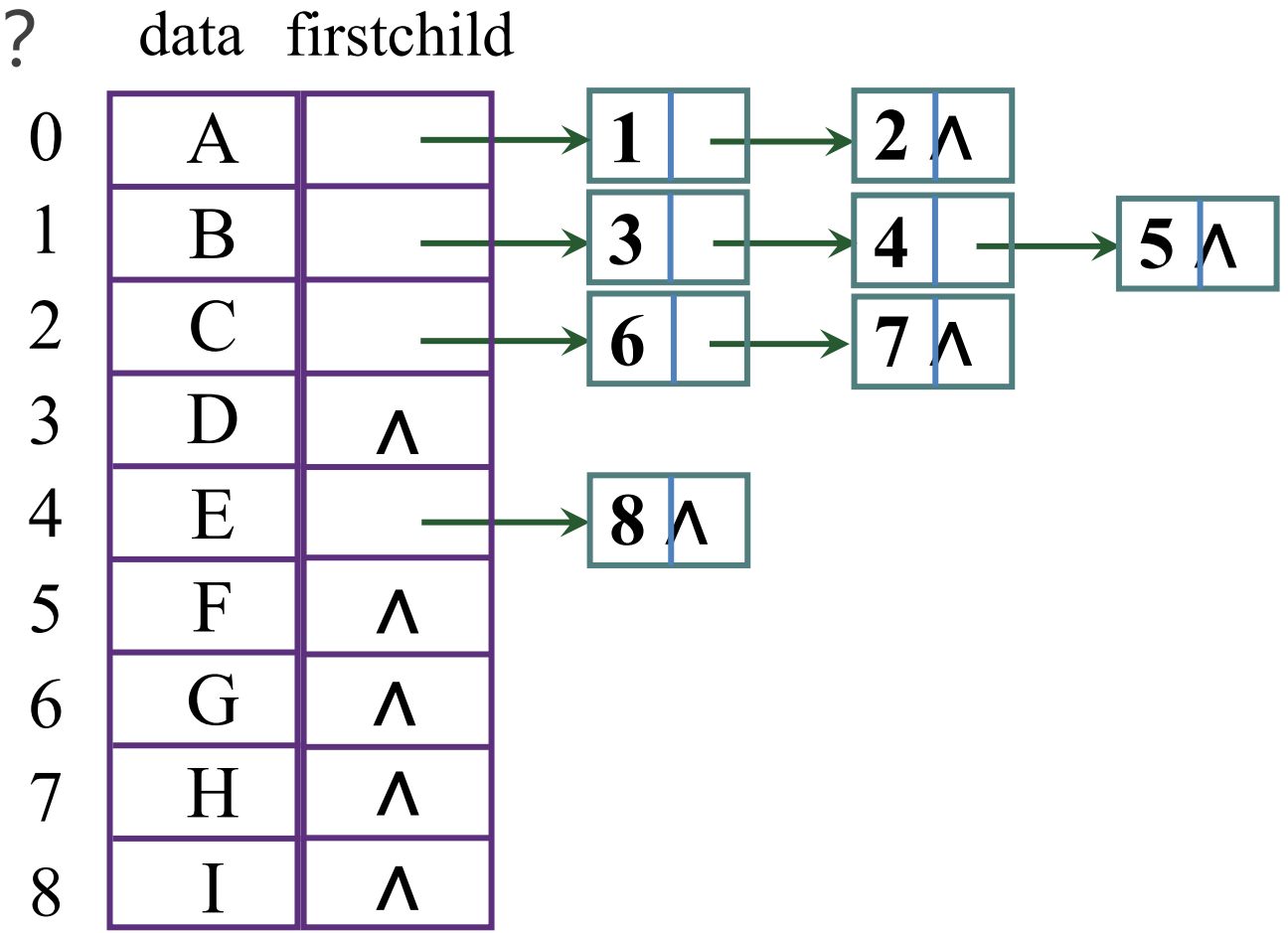
如何定义树的孩子表示法呢?

孩子结点

child	next

表头结点

data	firstchild



2. 树的孩子表示法的实现

如何定义树的孩子表示法呢?

孩子结点

child	next

```
struct CTNode                    //孩子结点
{
    int child;
    CTNode *next;
};
```

表头结点

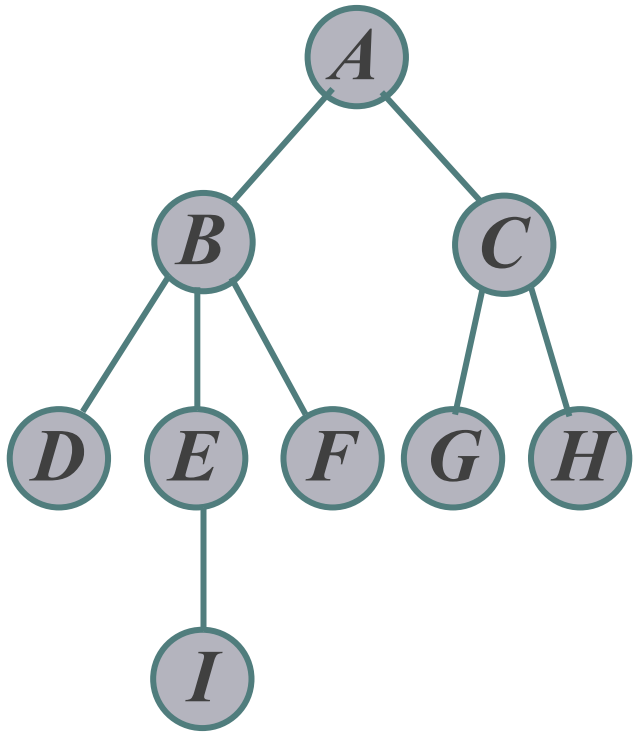
data	firstchild

```
template <typename DataType>
struct CBNode                    //表头结点
{
    DataType data;
    CTNode *firstChild;  //指向孩子链表的头指针
};
```

3. 树的孩子表示法性能分析

如何查找孩子结点？时间性能？

$O(1)$

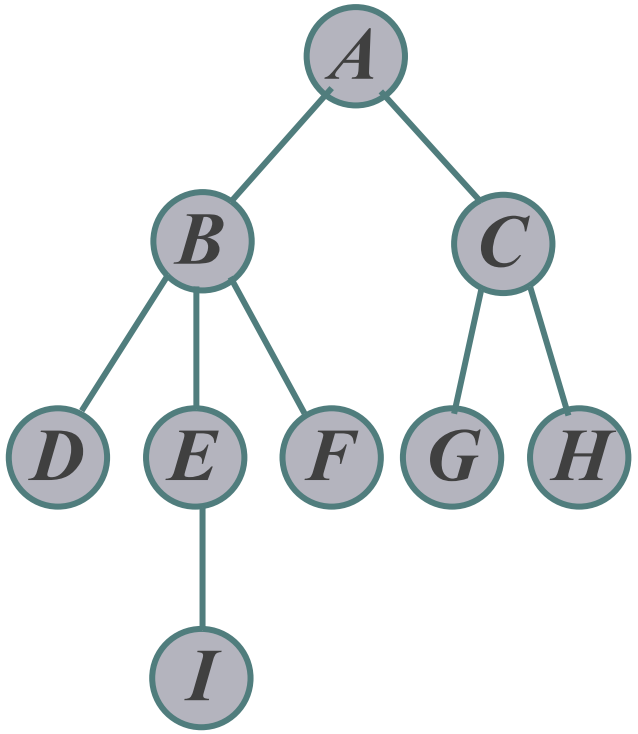


	data	firstchild
0	A	1 → 2 ∧
1	B	3 → 4 → 5 ∧
2	C	6 → 7 ∧
3	D	∧
4	E	8 ∧
5	F	∧
6	G	∧
7	H	∧
8	I	∧

3. 树的孩子表示法性能分析

$O(n)$

如何查找双亲结点？时间性能？



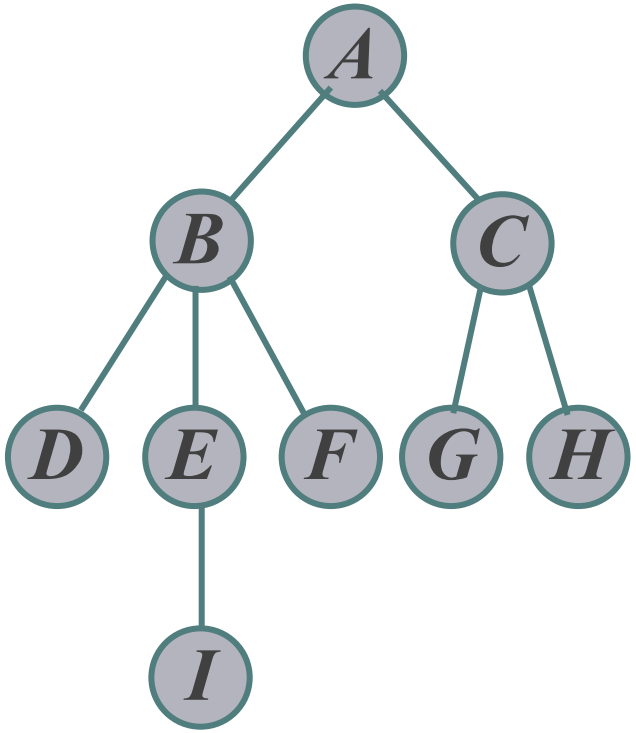
	data	parent	data	firstchild
0	A	−1	A	1
1	B	0	B	3
2	C	0	C	6
3	D	1	D	∧
4	E	1	E	8
5	F	1	F	∧
6	G	2	G	∧
7	H	2	H	∧
8	I	4	I	∧

5.3 树的存储结构

5-3-3　树的孩子兄弟表示法

1. 树的孩子兄弟表示法

- 树的孩子兄弟表示法（二叉链表）：链表中的每个结点包括数据域和分别指向该结点的第一个孩子和右兄弟的指针



某结点的第一个孩子是惟一的

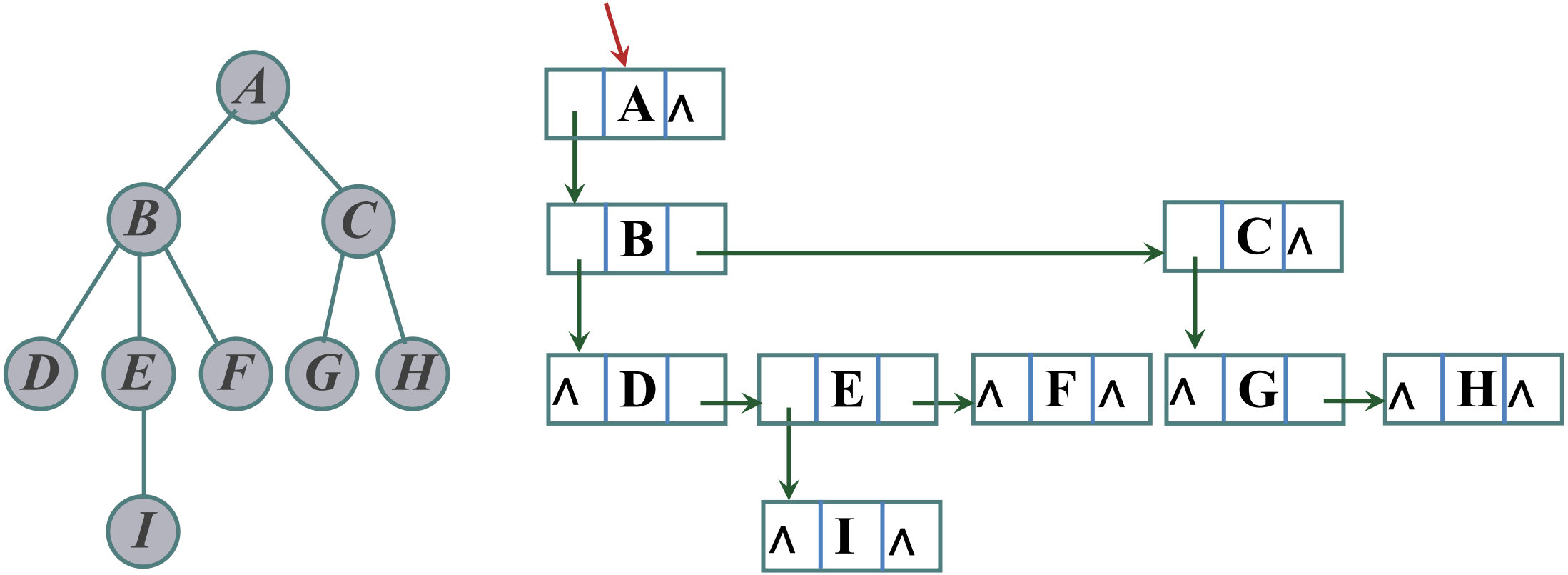
某结点的右兄弟是惟一的

⇩

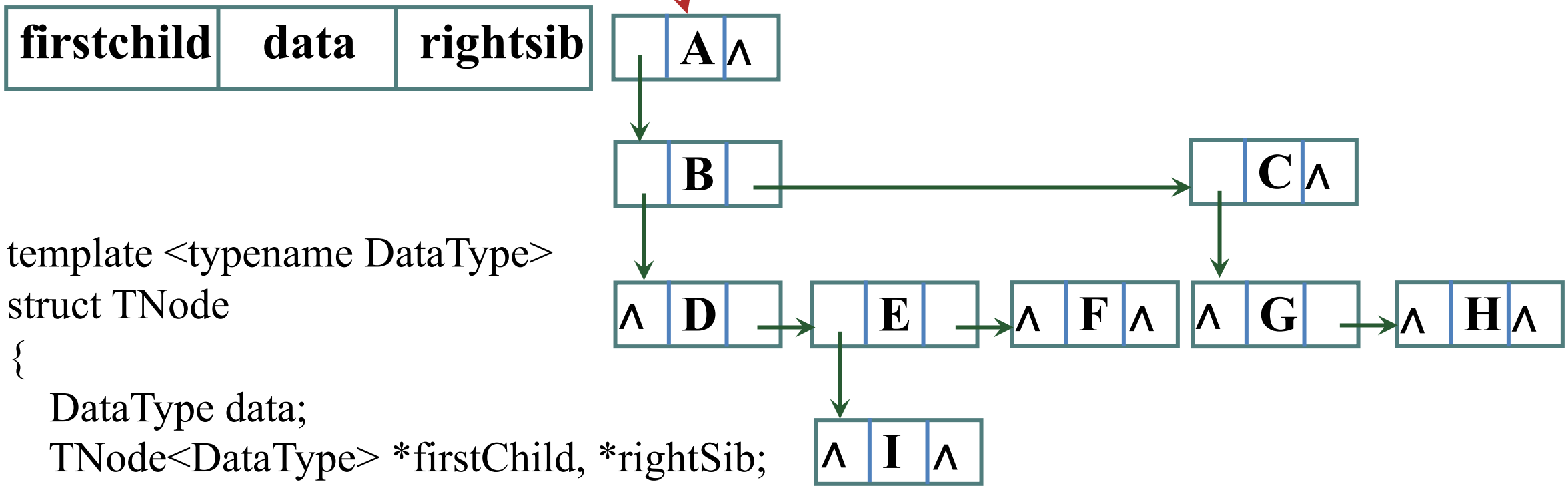
设置两个分别指向该结点的第一个孩子和右兄弟的指针

1. 树的孩子兄弟表示法

- 树的孩子兄弟表示法（二叉链表）：链表中的每个结点包括数据域和分别指向该结点的第一个孩子和右兄弟的指针



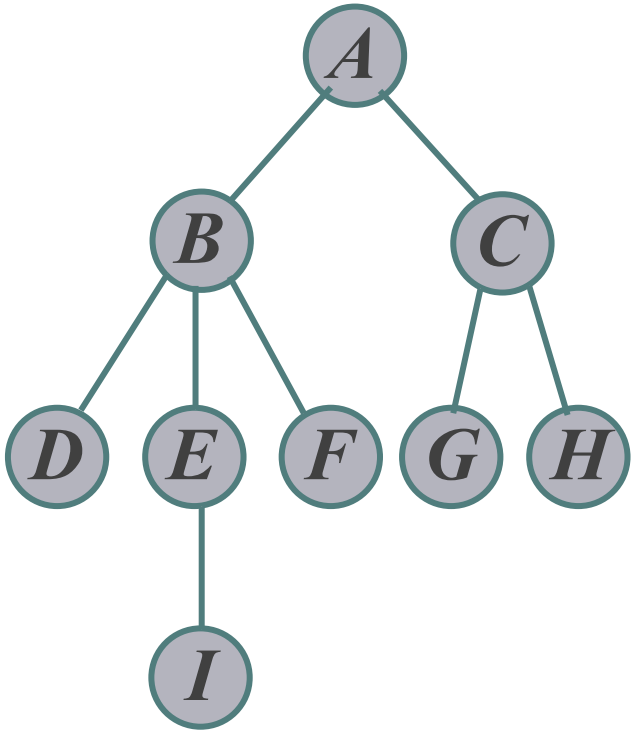
2. 树的孩子兄弟表示法—存储结构

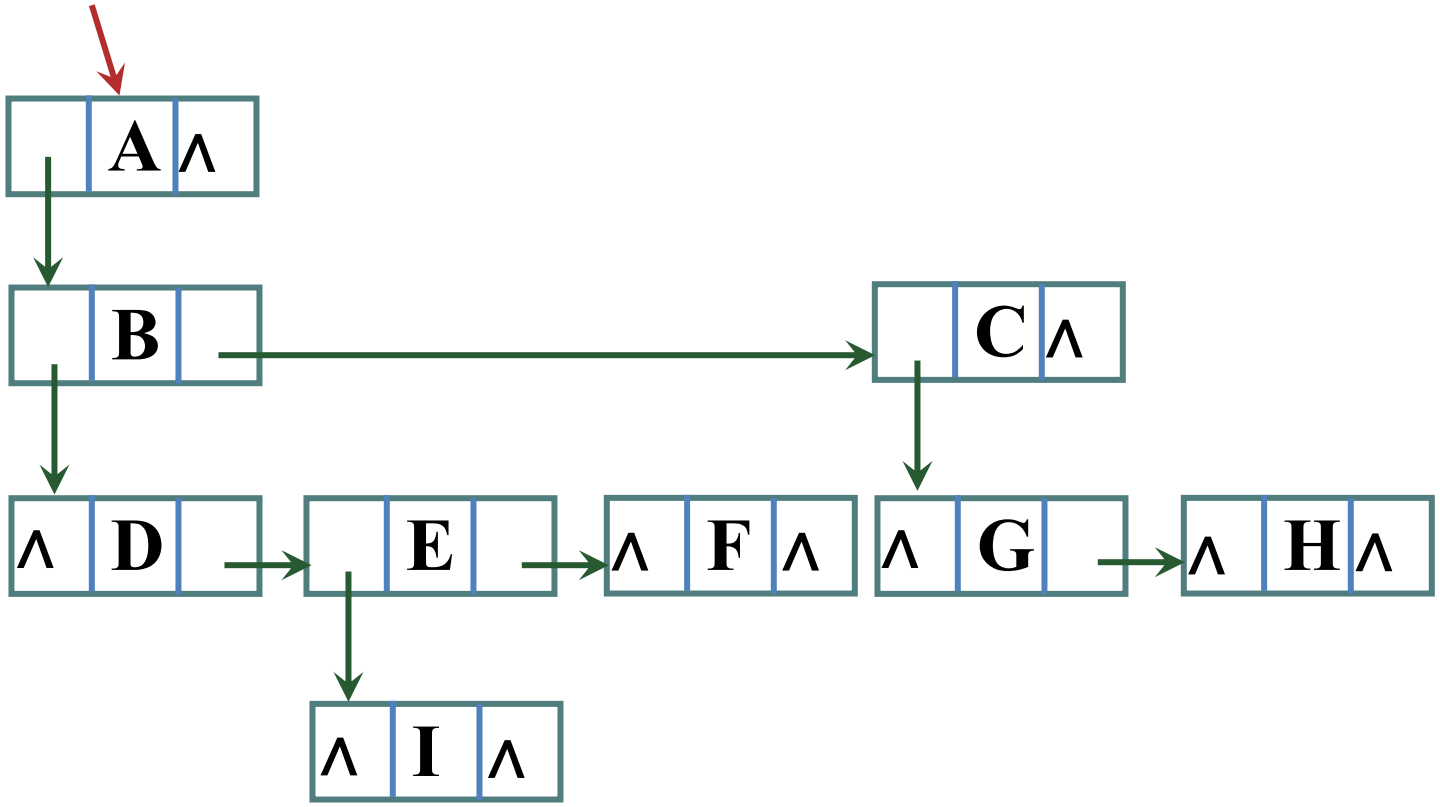



```
template <typename DataType>
struct TNode
{
    DataType data;
    TNode<DataType> *firstChild, *rightSib;
};
```

3. 树的孩子兄弟表示法—性能分析

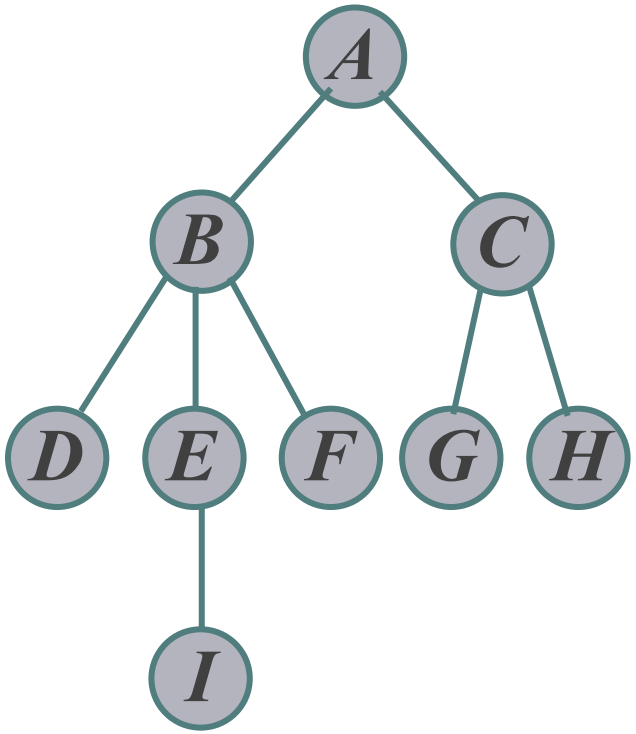
如何查找兄弟结点？时间性能？ $O(1)$

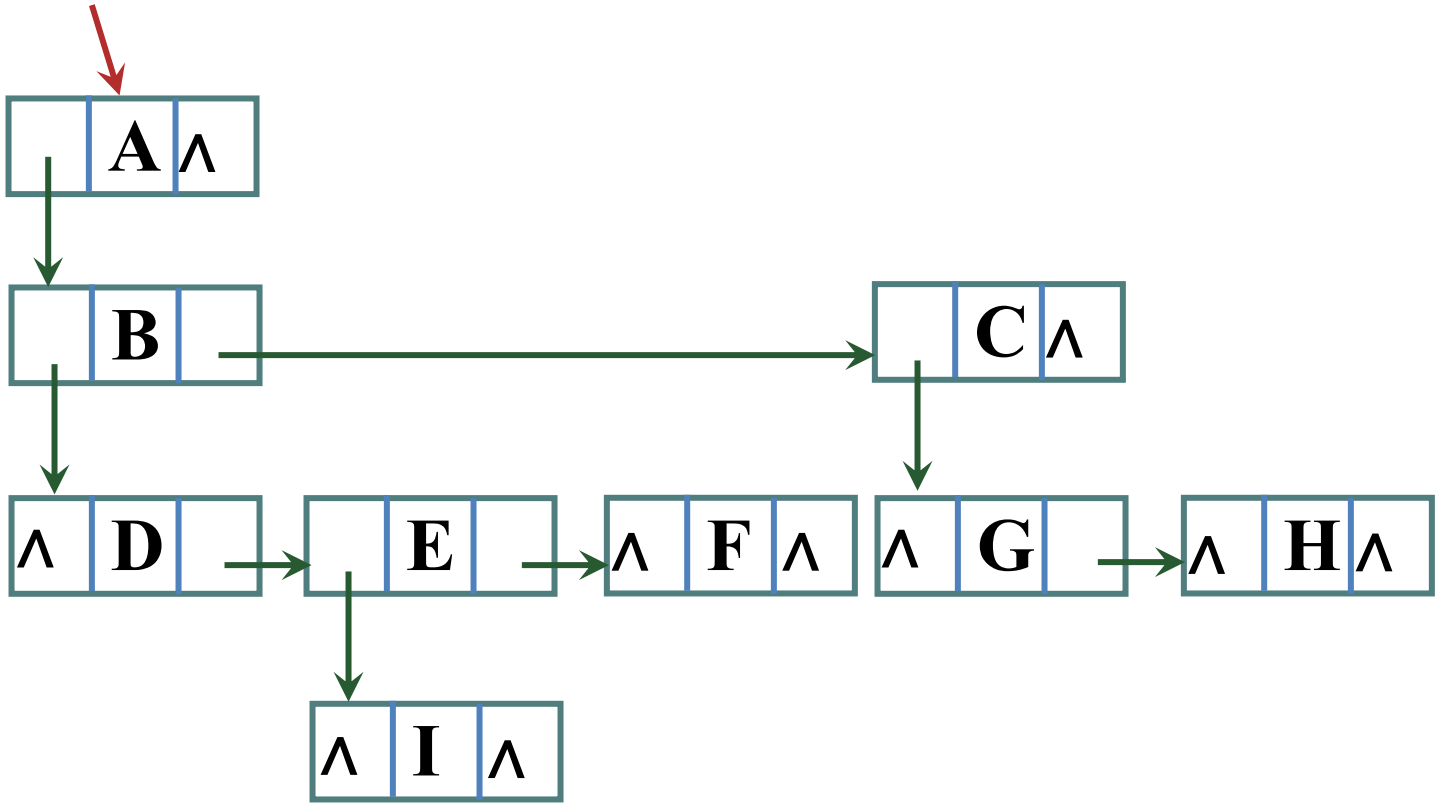




3. 树的孩子兄弟表示法—性能分析

如何查找孩子结点？时间性能？ $O(n)$ ⇨ 已知该结点指针：$O(1)$



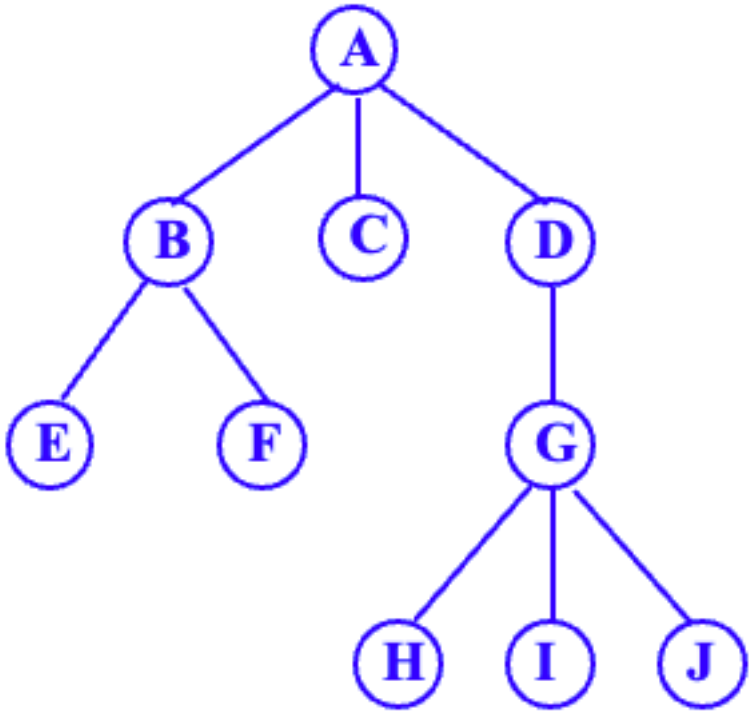


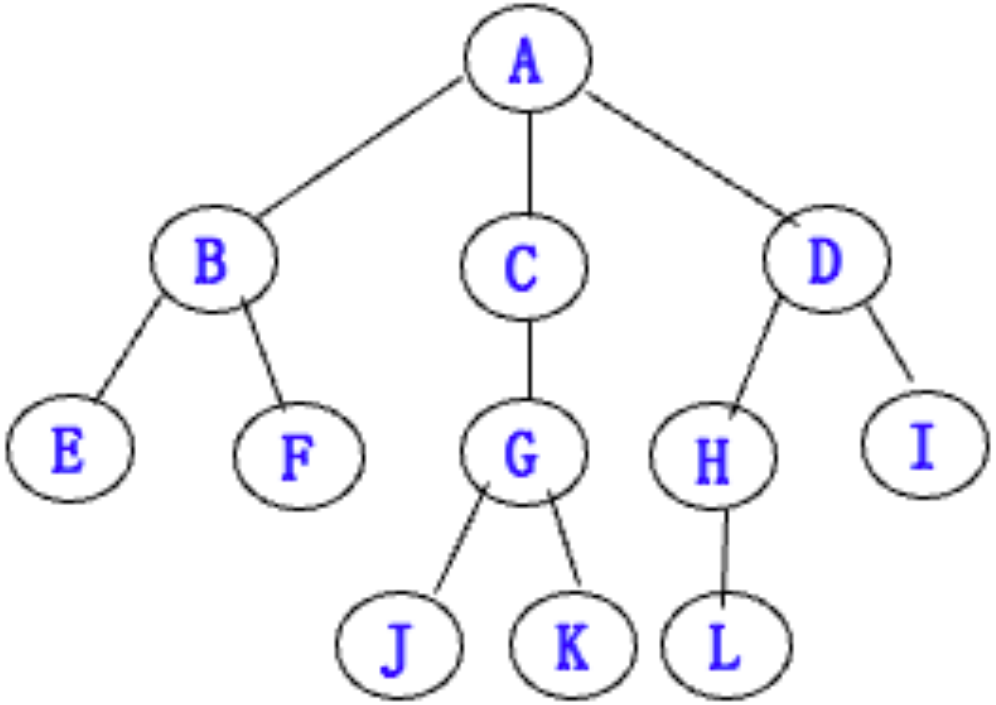
小结

1. 掌握树的定义（递归）
2. 理解树的抽象数据类型定义
3. 熟练掌握树的遍历方法（前序、后序、层次）
4. 掌握树的存储结构（双亲表示/孩子表示/孩子兄弟表示）

作业

1. 写出如下图中树的先序遍历、后序遍历、层次遍历序





Thank You !

Q & A