抽象数据结构的物理实现

主要内容

- 物理数据结构的概述、定义
- 常见抽象数据的物理实现

概述

物理数据结构：就是研究如何将我们定义好的抽象数据结构存储在我们的计算机内存中，同时，在计算机物理内存中的存储方式要能很好地反应抽象数据的特点

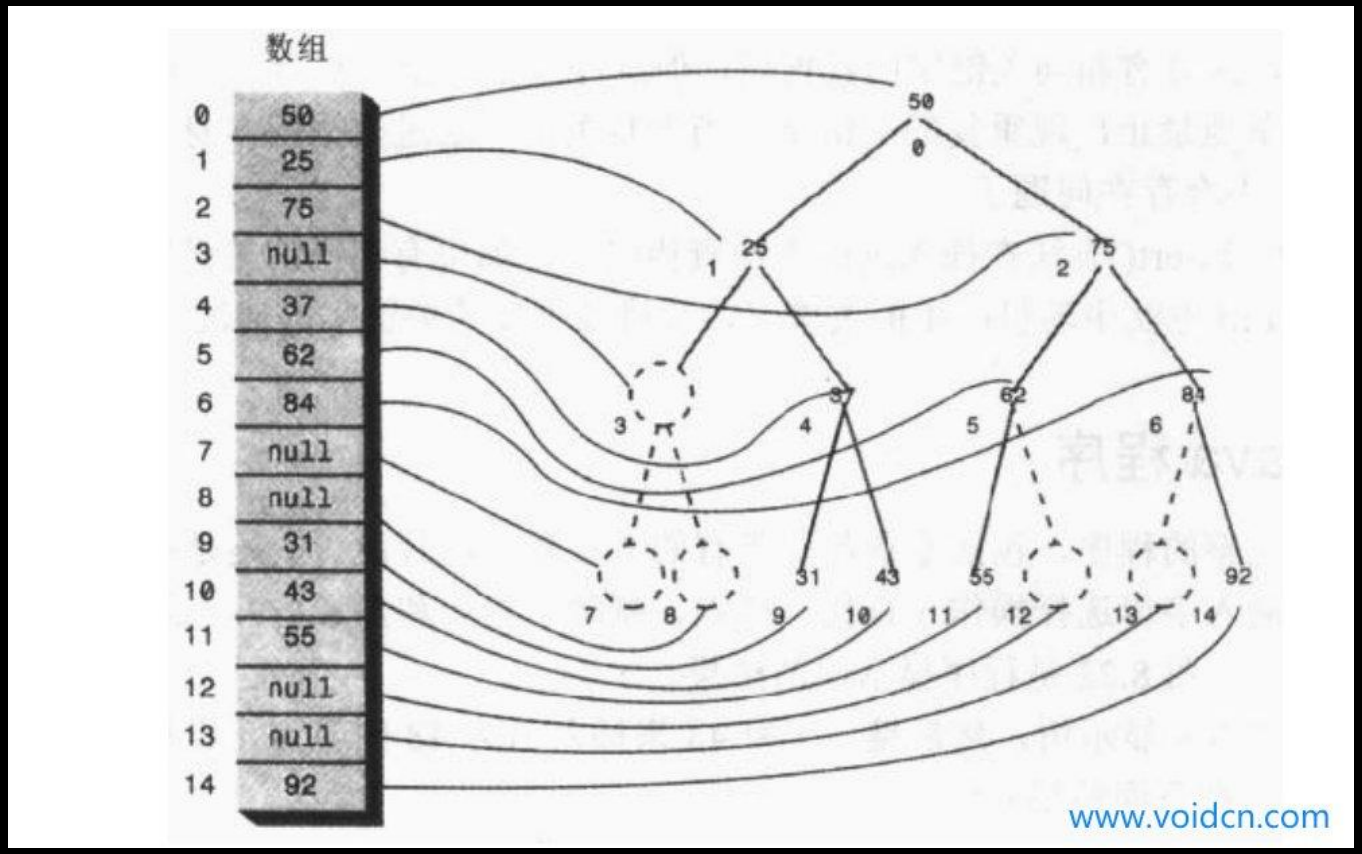
常见的物理存储方式：

* 顺序存储结构——地址连续存储，逻辑地址和物理地址相同（如数组结构）。特点是能根据下标直接访问到，有较快的访问速度，缺点是插入删除时需要改变后续元素的位置，效率低。

* 链式存储结构——地址不一定是连续的，逻辑地址和物理地址不对应，所以存储是需要存储元素的地址（如链表）。特点是插入删除时无需改变其他元素的位置，所以插入删除时有很快的效率，缺点是查找需要重头遍历，时间消耗大。

二叉树的物理实现—数组

用数组的形式来构建二叉树,节点存在数组中,而不是由引用相连,节点在数组中的位置对应它在树中的位置,下标为0 的节点为根节点,下标为1是根的左节点,2为根节点的右节点,依次类推,从左到右的顺序存储树的每一层,包括空节点。如下图:



主要特点：

性质：节点的子节点和父节点可以利用简单的算术计算它们在数组中的索引值

2*index+1

右子节点索引为:

2*index+2

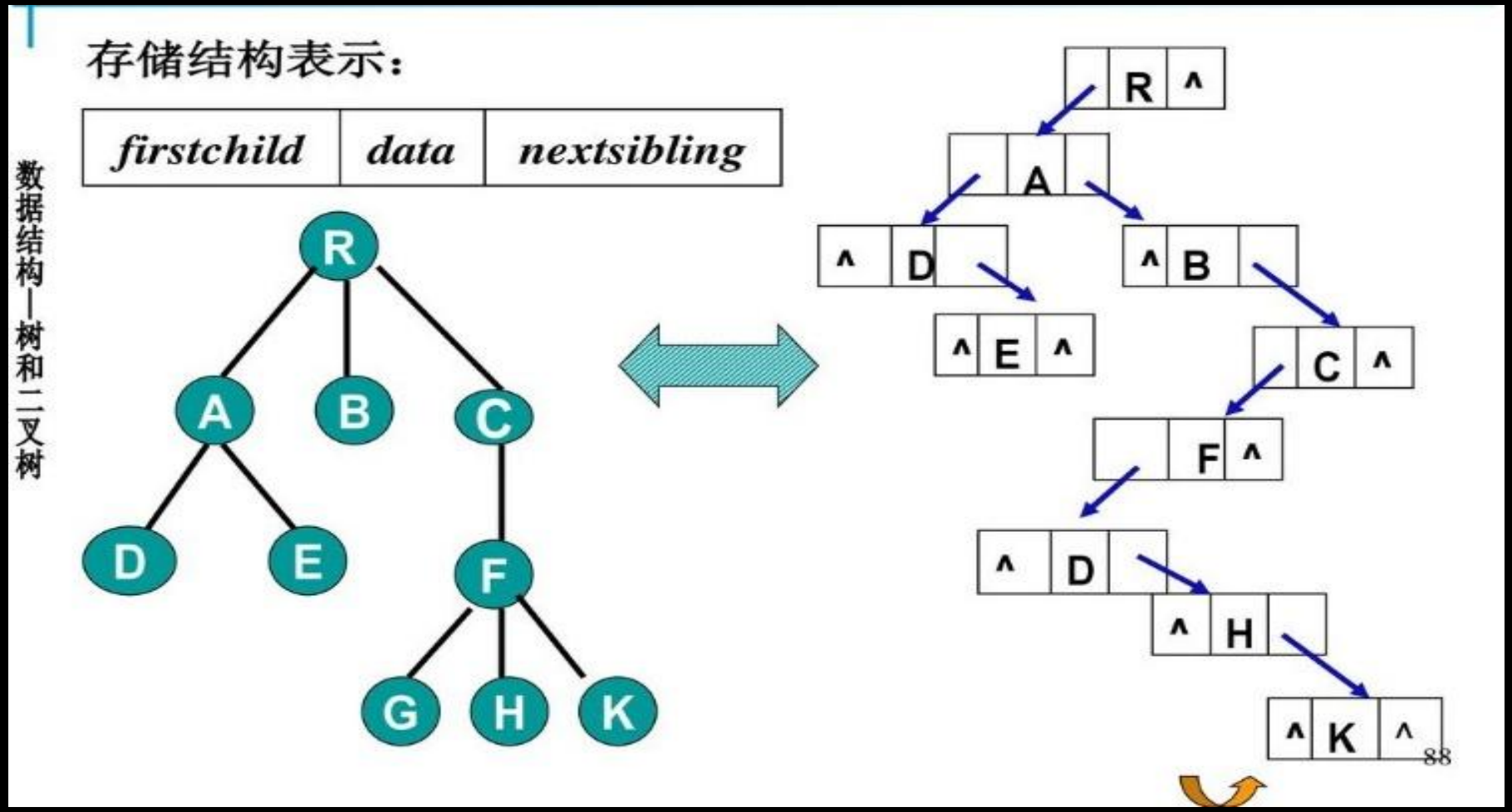
父节点索引为:设某个节点索引值为index,则节点的左子节点索引为:

(index-1)/2

优缺点：大多数情况下用数组表示数不是很有效率,除非是完全二叉树，因为如果是普通的二叉树,特别是有很多空节点的.会有很多空洞,浪费存储空间.用数组表示树,删除节点是很费时费力的.所以用数组表示树适合用于 完全二叉树查找,和插入.

二叉树的物理实现—二叉链表

二叉树一般使用链式存储结构，由二叉树的定义可知，二叉树的结点由一个数据元素和分别指向其左右孩子的指针构成，即二叉树的链表结点中包含3个域，这种结点结构的二叉树存储结构称之为二叉链表。如下图：



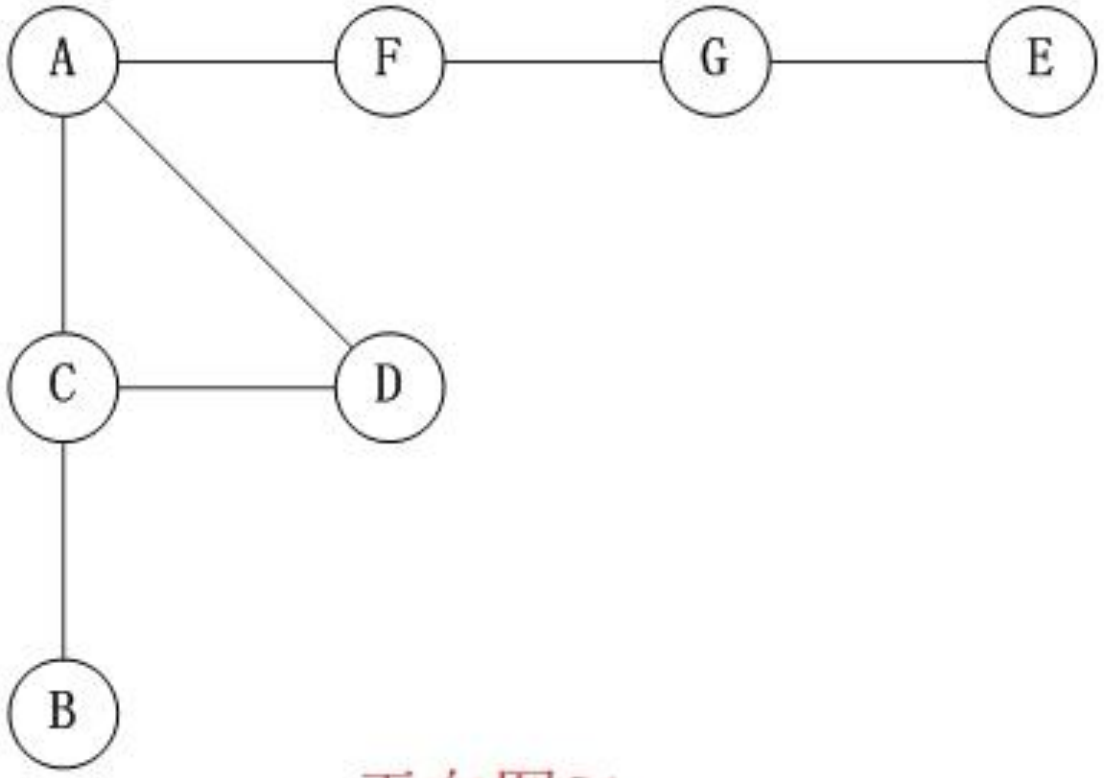
主要特点：

性质：节点与父节点之间的关系通过指针来确定。同时，父节点与子节点之间能有大小的关系，即左子树<父子树<右子树，这样就能提高查找的效率，不用遍历所有的节点。这样的存储结构还能有很大的扩展空间，能建立平衡二叉树、红黑树等等。

优缺点：这种存储方式有很大的灵活性，当树是稀疏树时，不会浪费太多的空间，同时，在添加和删除的操作中，也有很大的时间效率。

图的物理实现—相邻矩阵

图的邻接矩阵存储方式是用两个数组表示图，一个一维数组存储图中的顶点信息，一个二维数组存储图中的边或弧的信息。



无向图G1

	A	B	C	D	E	F	G
A	0	0	1	1	0	1	0
B	0	0	1	0	0	0	0
C	1	1	0	1	0	0	0
D	1	0	1	0	0	0	0
E	0	0	0	0	0	0	1
F	1	0	0	0	0	0	1
G	0	0	0	0	1	1	0

G1的邻接矩阵

邻接矩阵—有什么好处

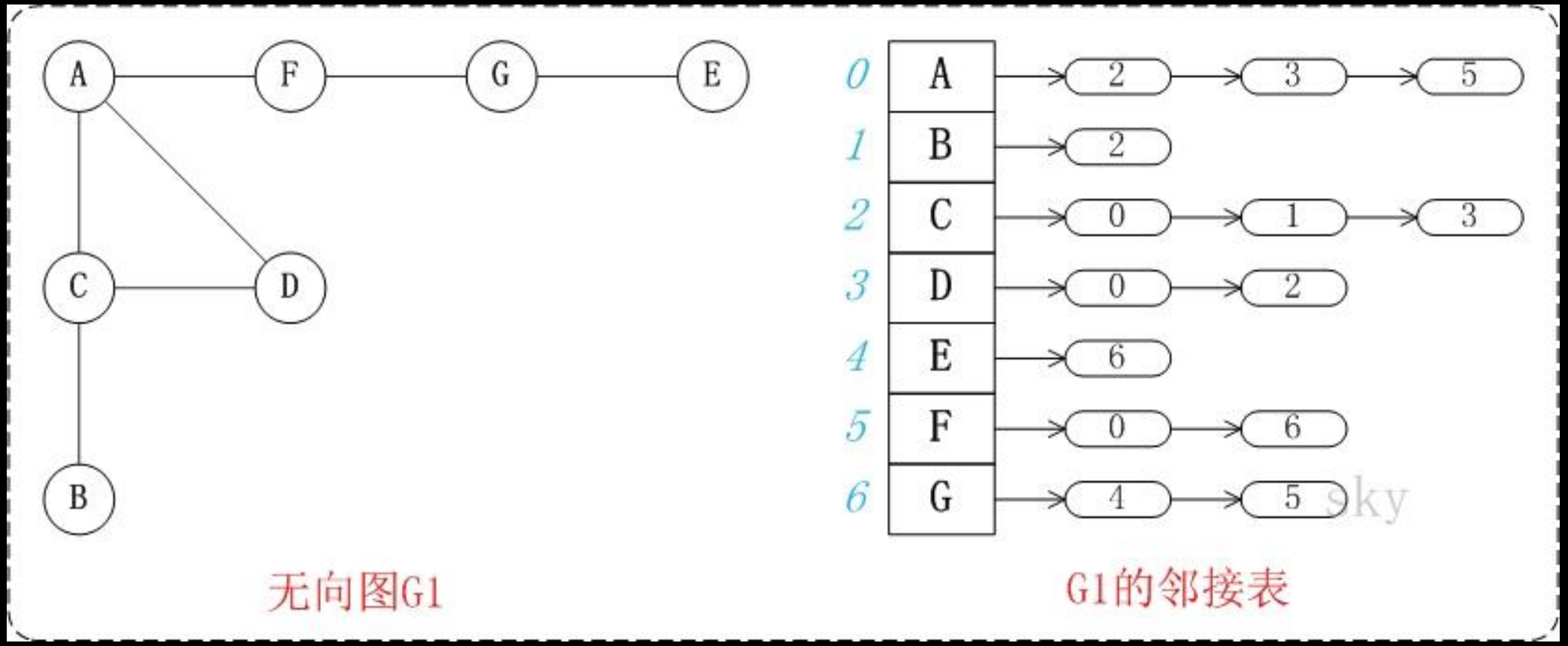
- 直观、简单、好理解
- 方便检查任意一对顶点是否存在边
- 方便找任一顶点的所有“邻接点”
- 方方便计算任一顶点的“度”　对于无向图，对应行或列的非零元素的个数即是该顶点的度；有向图，对应行非零元素的个数是出度，对应列非零元素个数为入度

邻接矩阵—有什么不好

- 浪费空间　对于点多但边很少的稀疏图，有大量的无效元素，不过对稠密图利用率较高
- 浪费时间　统计稀疏图的变数时，要遍历整个数组，时间效率不高

图的物理实现—邻接表

图中的顶点用一个一维数组存储，当然也能用链表存储，但数组更容易读取顶点信息。图中每一个顶点的所有邻接点构成一个线性表，由于邻接点个数不定，用链表存储。



邻接表—有什么好处

- 方便检查任意一对顶点是否存在边
- 方便找任一顶点的所有“邻接点”
- 方方便计算任一顶点的“度”　只需遍历该顶点的边表；但对于有向图不变找到入度是多少
- 对于稀疏图，不浪费空间

邻接表—有什么不好

- 浪费空间　每一条边都存储了两次
- 浪费时间　要遍历整个图时，时间效率不高
- 删除、添加麻烦