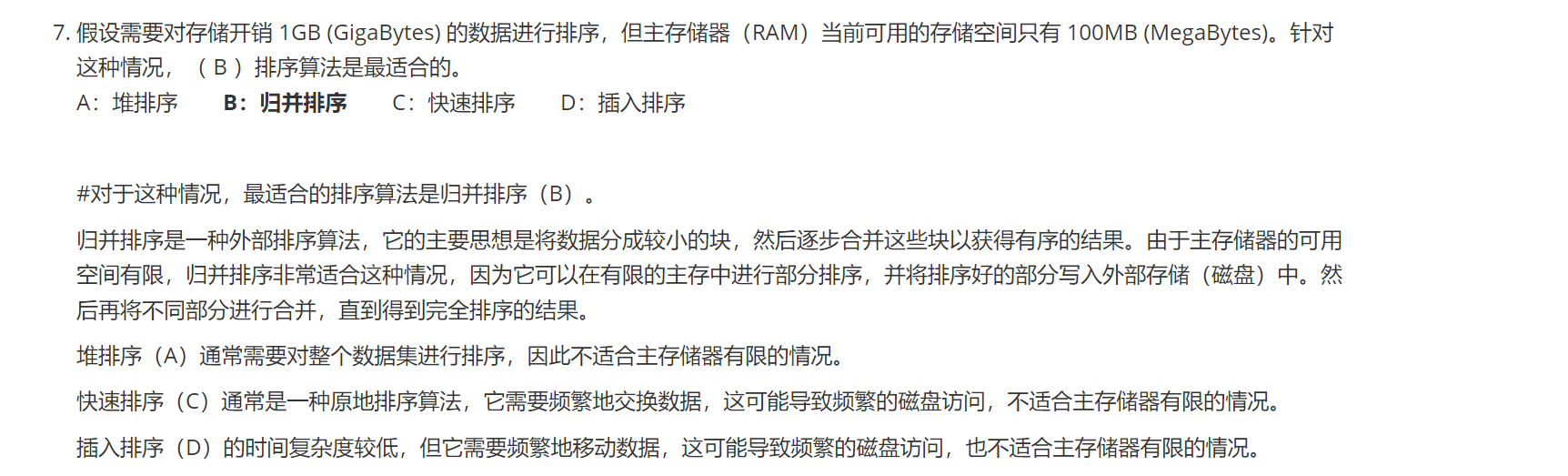
Sort algorithm:

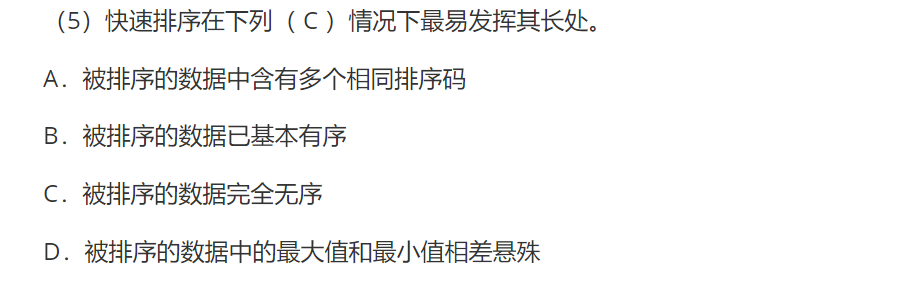
1.十大经典排序算法及动图演示 - 知乎

<https://zhuanlan.zhihu.com/p/449501682>

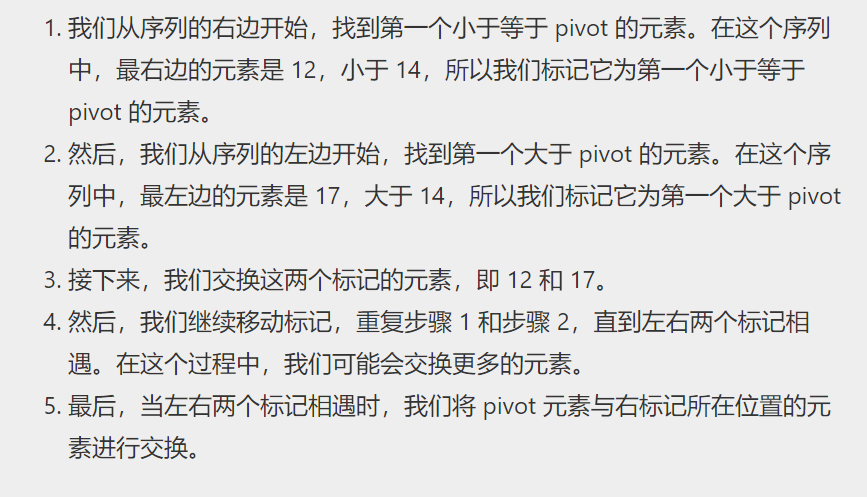
比较类排序只有冒泡、插入、归并稳定。

选择排序、堆排序和归并排序是最好和最坏复杂度相同的。

空间归并最大，但是：



1. 老师课件
2. （1）quicksort双指针



1. mergesort从len=1的子表开始合并，递归先左后右
2. 冒泡排序、选择排序 每次提取最大的到右边，冒泡是不断比较冒上来（同时交换），选择是直接找到最大的
3. Shellsort：注意gapsize=前后2个元素索引差值，不是纯间隔的长度
4. 堆排序是选择排序，堆是完全二叉树，堆排序最适用于partion-sort.

在数据结构中，从逻辑上可以把数据结构分成线性结构和非线性结构。

数据的基本单位是数据元素，数据的最小单位是数据项。

数据对象：是性质相同的数据元素的集合，是数据的子集

数据结构是数据对象，以及存在于该对象的实例合组成实例的数据元素之间的各种联系。这些联系可以通过定义相关的函数来给出。



存储结构：

存储结构描述了数据在计算机内部的实际存储方式。它主要关注数据在计算机内存中的组织形式，以及如何在内存中存储和访问数据。存储结构通常包括以下几种：

顺序存储结构：数据元素在内存中按照一定的顺序依次存储，相邻的元素在内存中也是相邻的，如顺序表、数组，可以随机存取。

链式存储结构：数据元素通过指针相互连接，每个元素在内存中可以不连续存储，如链表。

链接存储的存储结构所占存储空间分两部分，一部分存放结点值，另一部分存放表示结点间关系的指针,单链表数据密度<1

索引存储结构：除了数据元素本身外，还维护一个索引表，索引表中的每个元素指向相应数据元素的位置。

散列存储结构：通过散列函数将数据元素直接映射到内存地址，实现快速的查找。

存储结构与具体的数据结构有关，不同的数据结构可以选择不同的存储结构来实现。例如，数组通常使用顺序存储结构，链表则使用链式存储结构。

逻辑结构：

逻辑结构描述了数据元素之间的逻辑关系，以及对这些关系的操作。它不涉及具体的存储细节，而是关注数据元素之间的关联和组织方式。

逻辑结构通常包括以下几种：

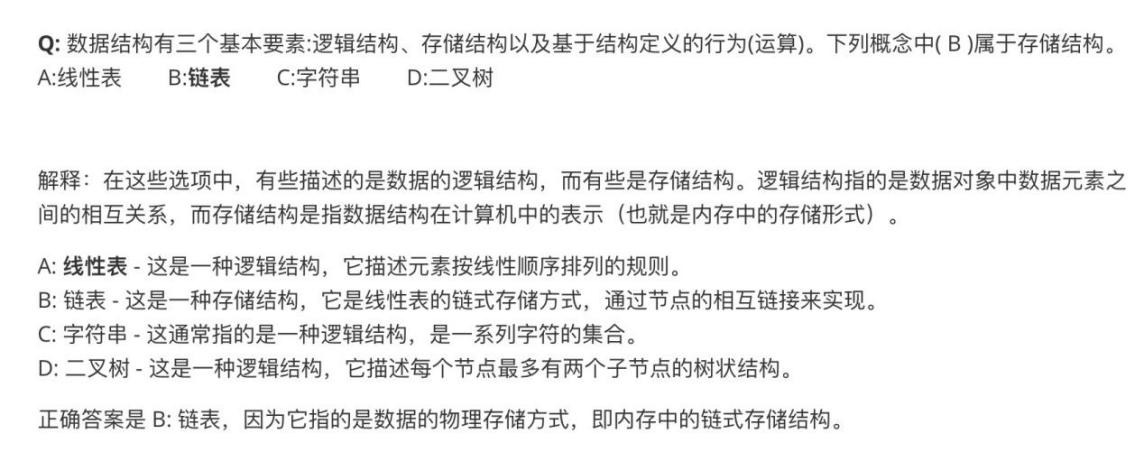
线性结构：数据元素之间存在一对一的关系，可以通过一维的方式排列，例如线性表、栈、队列等。

树形结构：数据元素之间存在一对多的关系，可以用树形方式组织，例如二叉树、堆、哈夫曼树等。

图形结构：数据元素之间存在多对多的关系，例如图、网等。

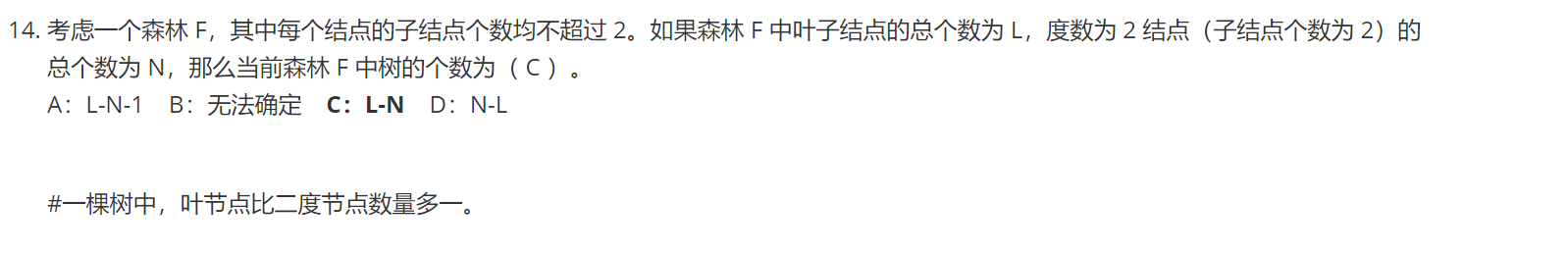
有序表也属于逻辑结构

逻辑结构与数据的抽象关系更为密切，它们描述了数据元素之间的逻辑联系和操作规则，而不关注数据元素的存储方式。例如，队列的逻辑结构就是先进先出（FIFO）的关系，而队列的存储方式可以是数组或链表。



有向图的邻接矩阵、邻接表和逆邻接表\_逆邻接表怎么画-CSDN博客

<https://blog.csdn.net/dyw_666666/article/details/85232559>



1.开放地址法

开放地址法的基本思想是：把记录都存储在散列表数组中，当某一记录关键字 key 的初始散列地址 H0 = H(key)发生冲突时，以 H0 为基础，采取合适方法计算得到另一个地址 H1，如果 H1 仍然发生冲突，以 为基础再求下一个地址 H2，若 H2 仍然冲突，再求得 H3。依次类推，直至 Hk 不发生冲突为止，则 Hk 为该记录在表中的散列地址。

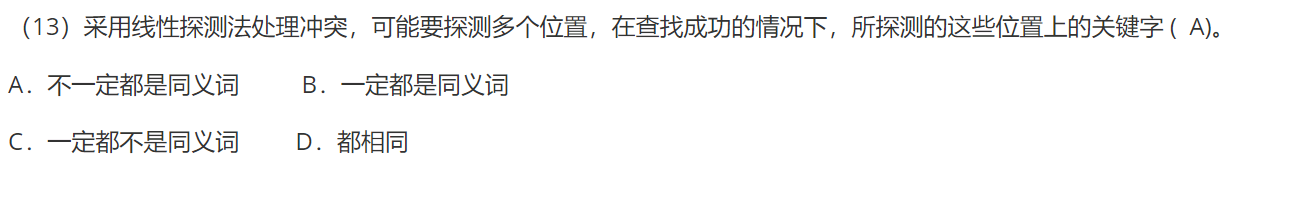
这种方法在寻找 ”下一个” 空的散列地址时，原来的数组空间对所有的元素都是开放的所以称为开放地址法。通常把寻找 “下一个” 空位的过程称为探测，上述方法可用如下公式表示：

$ H{i}=(H(key) +d{i})\%m $ $ i=1,2,…,k(k≤m-l) $

其中，H(key)为散列函数，m 为散列表表长，d为增量序列。根据d取值的不同，可以分为以下3种探测方法。

(1) 线性探测法 $ d\_{i} = 1, 2, 3, …, m-1 $

这种探测方法可以将散列表假想成一个循环表，发生冲突时，从冲突地址的下一单元顺序寻找空单元，如果到最后一个位置也没找到空单元，则回到表头开始继续查找，直到找到一个空位，就把此元素放入此空位中。如果找不到空位，则说明散列表已满，需要进行溢出处理。



(2) 二次探测法 di =1^2, -1^2, 2^2,-2^2,3^2,.…, +k^2,-k^2 ​

(3)伪随机探测法

$ d\_{i} = 伪随机数序列 $ 例如，散列表的长度为 11，散列函数 $ H(key)=key\%11$，假设表中已填有关键字分别为 17、60、29 的记录，如图7.29(a)所示。现有第四个记录，其关键字为38，由散列函数得到散列地址为 5，产生冲突。

若用线性探测法处理时，得到下一个地址6，仍冲突；再求下一个地址7，仍冲突；直到散列地址为8的位置为“空”时为止，处理冲突的过程结束，38填入散列表中序号为8的位置，如图2(b)所示。

若用二次探测法，散列地址5冲突后，得到下一个地址6，仍冲突；再求得下一个地址 4，无冲突，38填入序号为4的位置，如图 2(c)所示。

若用伪随机探测法，假设产生的伪随机数为9，则计算下一个散列地址为(5+9)%11=3，所以38 填入序号为3 的位置，如图 2(d)所示。

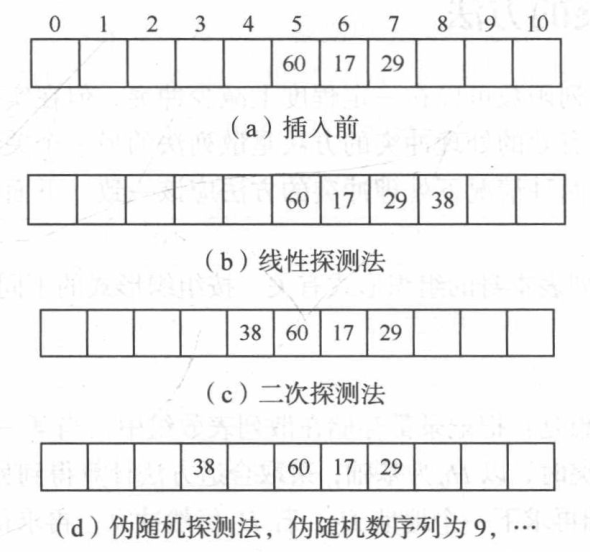


图 2 用开放地址法处理冲突时，关键字为38的记录插入前后的散列表

从上述线性探测法处理的过程中可以看到一个现象:当表中

$ i, i+1, i+2 $

位置上已填有记录时，下一个散列地址为

$ i、i+1、i+2、i+3 $

的记录都将填入i+3的位置，这种在处理冲突过程中发生的两个第一个散列地址不同的记录争夺同一个后继散列地址的现象称作“二次聚集”(或称作“堆积”)，即在处理同义词的冲突过程中又添加了非同义词的冲突。

可以看出，上述三种处理方法各有优缺点。线性探测法的优点是：只要散列表未填满，总能找到一个不发生冲突的地址。缺点是：会产生“二次聚集”现象。而二次探测法和伪随机探测法的优点是：可以避免“二次聚集”现象。缺点也很显然：不能保证一定找到不发生冲突的地址。

2.链地址法

链地址法的基本思想是：把具有相同散列地址的记录放在同一个单链表中，称为同义词链表。有 m个散列地址就有m 个单链表，同时用数组

$ HT[0…m-1] $

存放各个链表的头指针，凡是散列地址为i的记录都以结点方式插入到以 HT[]为头结点的单链表中。

【例】 已知一组关键字为 (19 , 14 , 23 , 1，68，20 , 84 , 27 , 55 , 11 , 10 , 79)，设散列函数

$ H(key)=key\%13 $

用链地址法处理冲突，试构造这组关键字的散列表。

由散列函数 H(key)=key %13 得知散列地址的值域为 0~12，故整个散列表有 13 个单链表组成，用数组 HT[0..12]存放各个链表的头指针。如散列地址均为1的同义词 14、1、27、79 构成一个单链表，链表的头指针保存在 HT[1]中，同理，可以构造其他几个单链表，整个散列表的结构如图 3 所示。

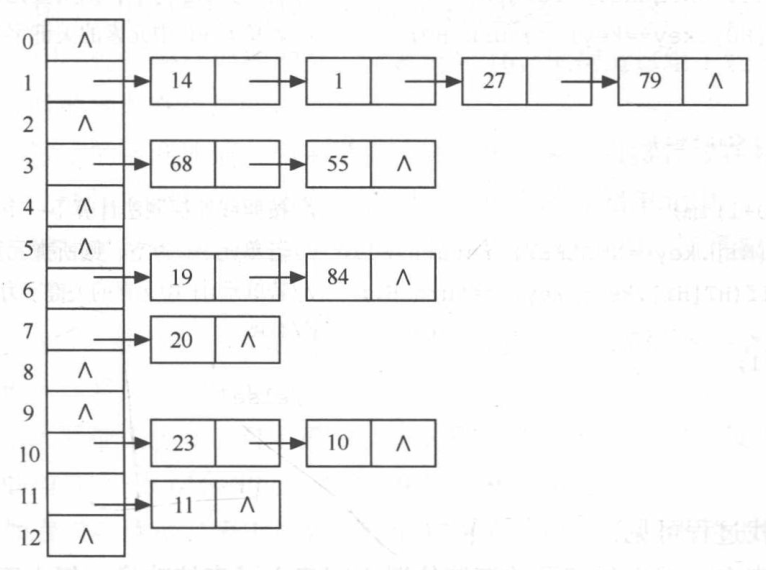
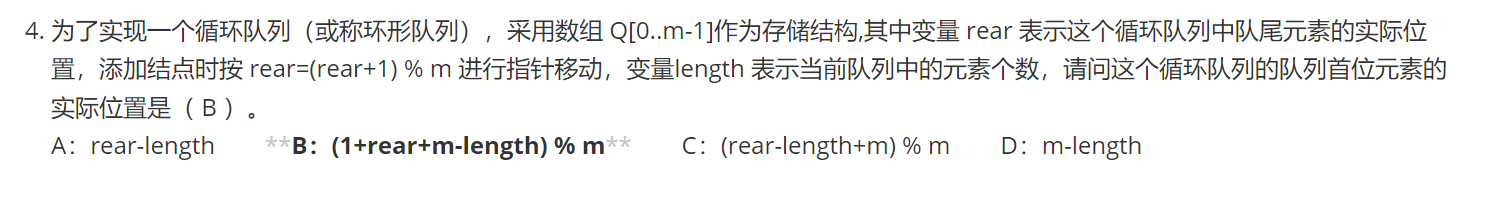
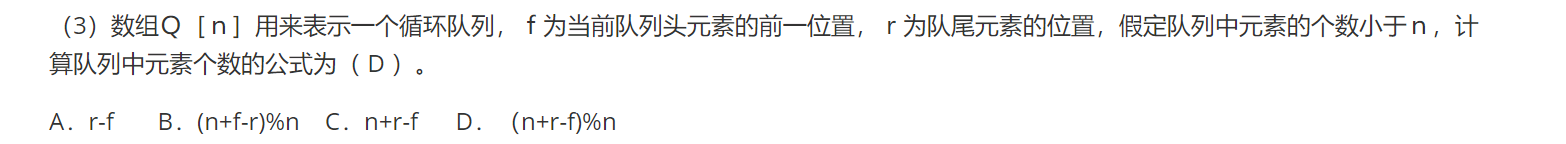


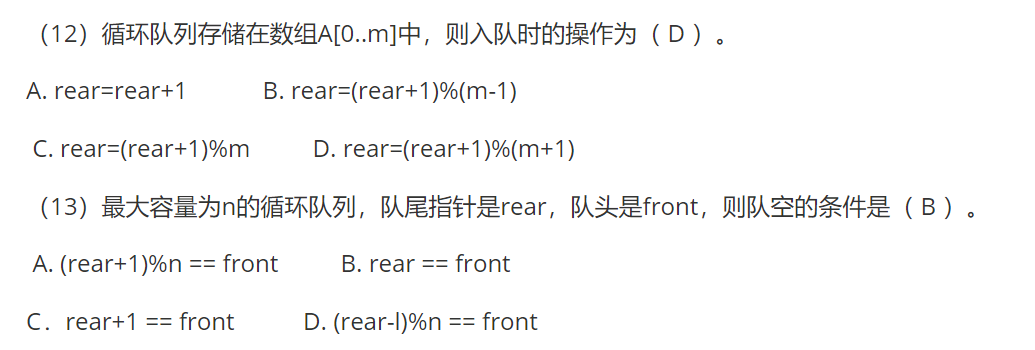
图 3 用链地址法处理冲突时的散列表

这种构造方法在具体实现时，依次计算各个关键字的散列地址，然后根据散列地址将关键字插入到相应的链表中。

有 n（n>=2）个顶点的有向强连通图最少有 \_ \_ \_ \_ \_ 条边。n

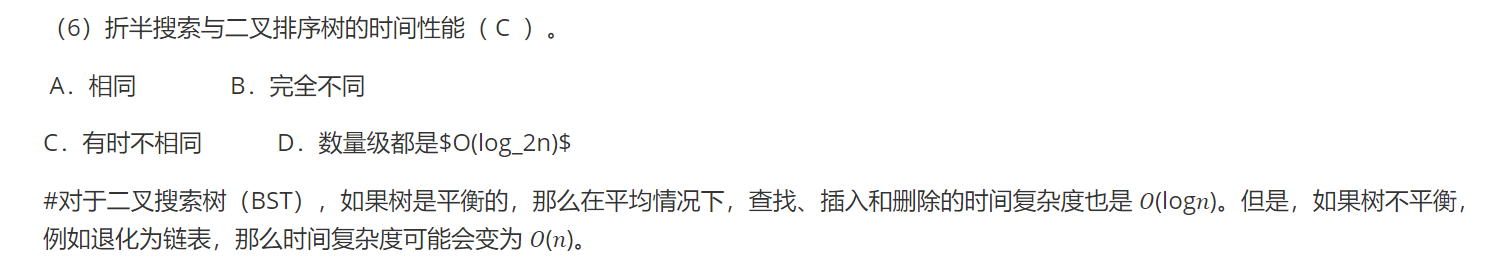
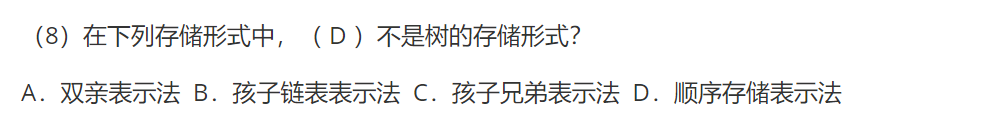
**循环队列：https://zhuanlan.zhihu.com/p/663842362?utm\_id=0**

1. 
2. 



1. 一个递归算法必须包括终止条件和递归部分
2. 回溯算法的核心思想是:从根结点出发,沿着某条路径向前搜索,当搜索到某一结点发现不满足要求时,就返回(回溯)到上一结点,继续尝试其他路径。回溯算法不需要使用队列来保存路径,而是通过递归调用、返回上一层等方式来实现状态的保存和恢复。

3.串是一种特殊的线性表，其特殊性体现在数据元素是一个字符,串是字符的有限序列,模式匹配是串的一种重要运算,串既可以采用顺序存储，也可以采用链式存储

1. 
2. 
3. 二叉树高度=层数-1，无向图节点度数=相连边数
4. prim适合稠密图，kruskal不能有负权边

森林到二叉树的转换：森林是一组不相交的树。将森林转换为二叉树的方法通常称为“孩子-兄弟表示法”：每个节点的第一个孩子作为该节点的左孩子。每个节点的兄弟作为该节点的右孩子。二叉树中右指针域为空的节点，我们需要确定在这种转换中，二叉树中右指针域为空的节点的数量。

分析过程

非终端结点的定义：非终端节点是指在原始森林中有孩子的节点。

转换后的二叉树：在二叉树中，只有那些没有右孩子的节点，其右指针域为空。

如果一个节点没有兄弟节点，那么在二叉树中这个节点的右指针域为空。

在孩子-兄弟表示法中，以下几类节点会导致右指针为空：

每棵树的根节点（因为它们没有兄弟）。每个非终端节点的最右侧的孩子（因为它们没有右兄弟）。右指针为空的节点数量。每棵树的根节点的右指针域为空。每个非终端节点的最后一个孩子的右指针域为空。考虑所有叶子节点，这些节点也没有右孩子。

由于根节点的右指针域为空，而森林的根节点数就是树的数量，每个树至少有一个根节点。如果我们设 n 是非终端节点的数量，那么根节点（第一层）的数量加上叶子节点的数量会影响右指针为空的节点总数。

关键点

对于森林转换成的二叉树：

如果 F 中有 n 个非终端节点，每个非终端节点的每个最后一个孩子节点的右指针域为空。

每棵树的根节点的右指针域为空，这些根节点数通常为 n+1（根节点数 = 非终端节点数 + 1）。

因此，二叉树中右指针域为空的节点数为 n+1

结论

如果 F 中有 n个非终端结点，则由 F变换得的二叉树 B中右指针域为空的节点数为：

n+1

DFS和BFS树：BFS遍历树和DFS遍历树\_bfs树和dfs树-CSDN博客

https://blog.csdn.net/z13192905903/article/details/103306846