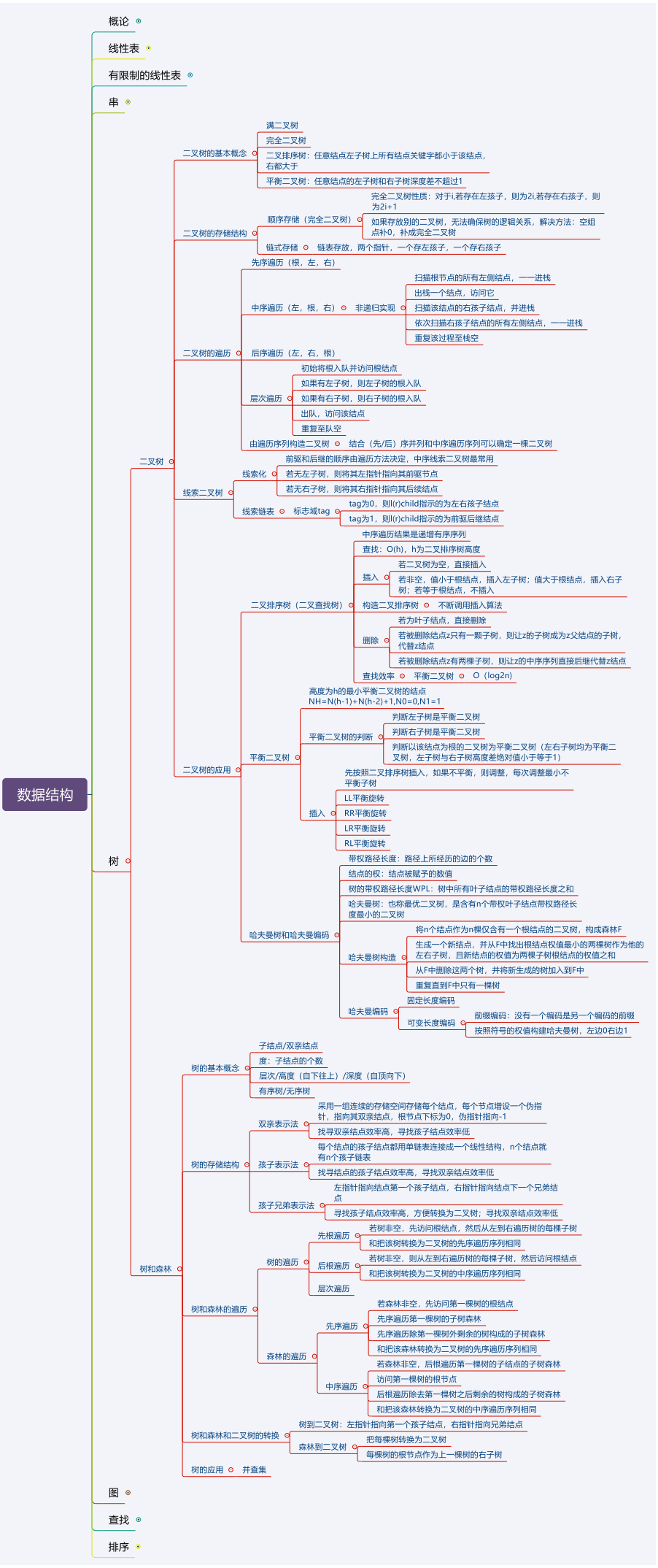
数据结构
概论
线性表
有限制的线性表
串
树
二叉树
二叉树的基本概念
满二叉树
完全二叉树
二叉排序树：任意结点左子树上所有结点关键字都小于该结点，右都大于
平衡二叉树：任意结点的左子树和右子树深度差不超过1
二叉树的存储结构
顺序存储（完全二叉树）
完全二叉树性质：对于i,若存在左孩子，则为2i,若存在右孩子，则为2i+1
如果存放别的二叉树，无法确保树的逻辑关系，解决方法：空姐点补0，补成完全二叉树
链式存储
链表存放，两个指针，一个存左孩子，一个存右孩子
二叉树的遍历
先序遍历（根，左，右）
中序遍历（左，根，右）
非递归实现
扫描根节点的所有左侧结点，一一进栈
出栈一个结点，访问它
扫描该结点的右孩子结点，并进栈
依次扫描右孩子结点的所有左侧结点，一一进栈
重复该过程至栈空
后序遍历（左，右，根）
层次遍历
初始将根入队并访问根结点
如果有左子树，则左子树的根入队
如果有右子树，则右子树的根入队
出队，访问该结点
重复至队空
由遍历序列构造二叉树
结合（先/后）序并列和中序遍历序列可以确定一棵二叉树
线索二叉树
前驱和后继的顺序由遍历方法决定，中序线索二叉树最常用
线索化
若无左子树，则将其左指针指向其前驱节点
若无右子树，则将其右指针指向其后续结点
线索链表
标志域tag
tag为0，则l(r)child指示的为左右孩子结点
tag为1，则l(r)child指示的为前驱后继结点
二叉树的应用
二叉排序树（二叉查找树）
中序遍历结果是递增有序序列
查找：O(h)，h为二叉排序树高度
构造二叉排序树
不断调用插入算法
插入
若二叉树为空，直接插入
若非空，值小于根结点，插入左子树；值大于根结点，插入右子树；若等于根结点，不插入
删除
若为叶子结点，直接删除
若被删除结点z只有一颗子树，则让z的子树成为z父结点的子树，代替z结点
若被删除结点z有两棵子树，则让z的中序序列直接后继代替z结点
查找效率
平衡二叉树
O（log2n）
平衡二叉树
高度为h的最小平衡二叉树的结点NH=N(h-1)+N(h-2)+1,N0=0,N1=1
平衡二叉树的判断
判断左子树是平衡二叉树
判断右子树是平衡二叉树
判断以该结点为根的二叉树为平衡二叉树（左右子树均为平衡二叉树，左子树与右子树高度差绝对值小于等于1）
插入
先按照二叉排序树插入，如果不平衡，则调整，每次调整最小不平衡子树
LL平衡旋转
RR平衡旋转
LR平衡旋转
RL平衡旋转
哈夫曼树和哈夫曼编码
带权路径长度：路径上所经历的边的个数
结点的权：结点被赋予的数值
树的带权路径长度WPL：树中所有叶子结点的带权路径长度之和
哈夫曼树：也称最优二叉树，是含有n个带权叶子结点带权路径长度最小的二叉树
哈夫曼树构造
将n个结点作为n棵仅含有一个根结点的二叉树，构成森林F
生成一个新结点，并从F中找出根结点权值最小的两棵树作为他的左右子树，且新结点的权值为两棵子树根结点的权值之和
从F中删除这两个树，并将新生成的树加入到F中
重复直到F中只有一棵树
哈夫曼编码
固定长度编码
可变长度编码
前缀编码：没有一个编码是另一个编码的前缀
按照符号的权值构建哈夫曼树，左边0右边1
树和森林
树的基本概念
子结点/双亲结点
度：子结点的个数
层次/高度（自下往上）/深度（自顶向下）
有序树/无序树
树的存储结构
双亲表示法
采用一组连续的存储空间存储每个结点，每个节点增设一个伪指针，指向其双亲结点，根节点下标为0，伪指针指向-1
找寻双亲结点效率高，寻找孩子结点效率低
孩子表示法
每个结点的孩子结点都用单链表连接成一个线性结构，n个结点就有n个孩子链表
找寻结点的孩子结点效率高，寻找双亲结点效率低
孩子兄弟表示法
左指针指向结点第一个孩子结点，右指针指向结点下一个兄弟结点
寻找孩子结点效率高，方便转换为二叉树；寻找双亲结点效率低
树和森林的遍历
树的遍历
先根遍历
若树非空，先访问根结点，然后从左到右遍历树的每棵子树
和把该树转换为二叉树的先序遍历序列相同
后根遍历
若树非空，则从左到右遍历树的每棵子树，然后访问根结点
和把该树转换为二叉树的中序遍历序列相同
层次遍历
森林的遍历
先序遍历
若森林非空，先访问第一棵树的根结点
先序遍历第一棵树的子树森林
先序遍历除第一棵树外剩余的树构成的子树森林
和把该森林转换为二叉树的先序遍历序列相同
中序遍历
若森林非空，后根遍历第一棵树的子结点的子树森林
访问第一棵树的根节点
后根遍历除去第一棵树之后剩余的树构成的子树森林
和把该森林转换为二叉树的中序遍历序列相同
树和森林和二叉树的转换
树到二叉树：左指针指向第一个孩子结点，右指针指向兄弟结点
森林到二叉树
把每棵树转换为二叉树
每棵树的根节点作为上一棵树的右子树
树的应用
并查集
图
查找
排序