第七章 查找

2022年10月2日 星期日 23:30

查找的知识框架

基本概念	• 静态查找，动态查找
线性结构	• 顺序查找 • 折半查找 • 分块查找
树形结构	• 二叉排序树 • 二叉平衡树 • 红黑树 • B树，B+树
散列结构	• 散列表
效率指标	• 平均查找长度

查找的基本概念

查找	• 在数据集合中寻找满足某种条件的数据元素的过程
查找表	• 用于查找的数据集合
关键字	• 数据元素中唯一标识该元素的某个数据项的值
平均查找长度	• 所有查找过程中进行关键字比较次数的平均值

查找的分类

静态查找表	• 只作查找操作的查找表	• 顺序查找 • 二分查找 • 插值查找 • 斐波那契查找 • 线性索引查找
动态查找表	• 表结构本身是在查找过程中动态生成的 • 查找时插入数据元素 • 查找时删除数据元素	• 二叉排序树 • 平衡二叉树 • B树 • 散列表

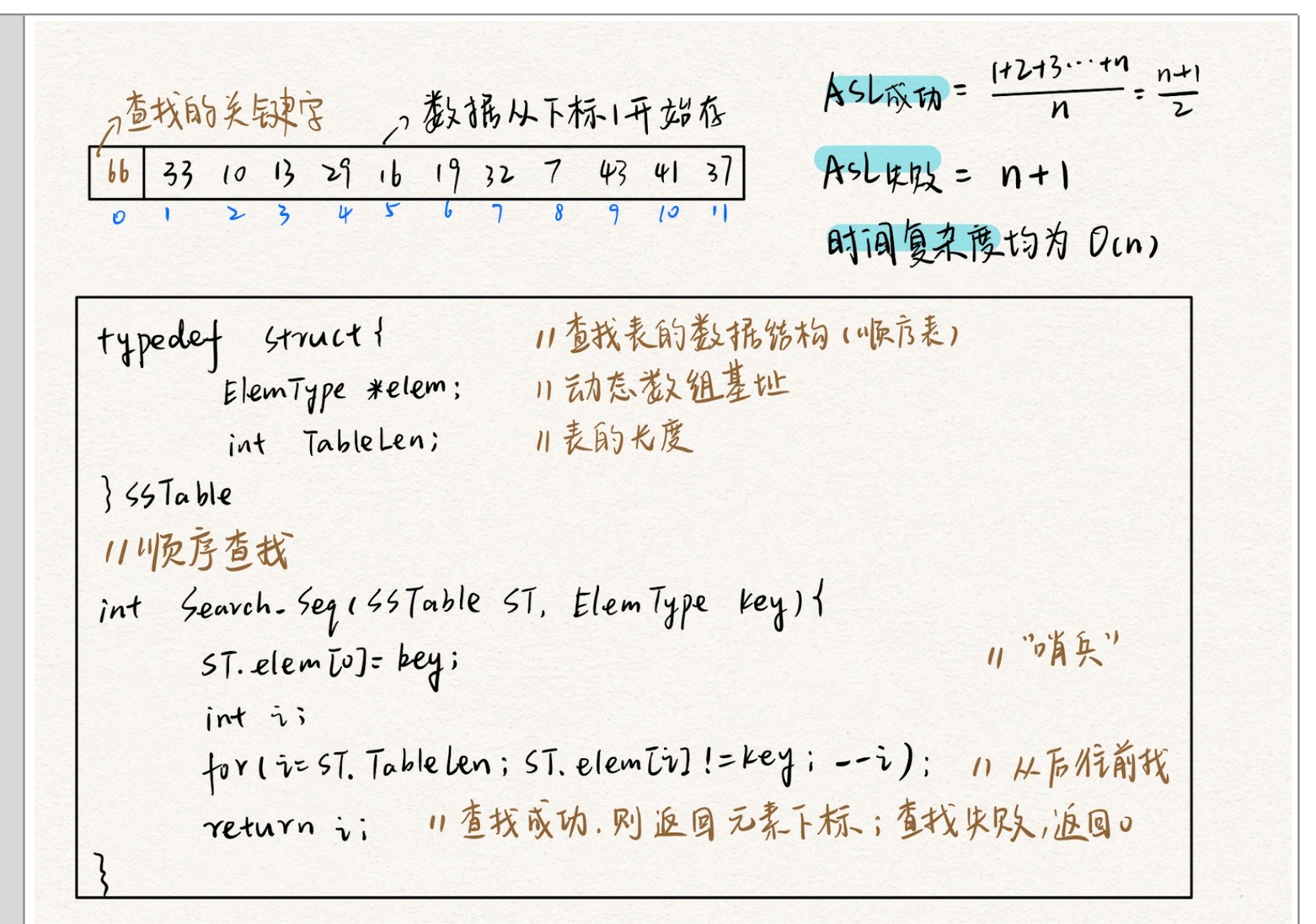
各个查找方法的指标

	时间复杂度	平均查找长度ASL
顺序查找	$O(n)$	
折半查找	$O(\log_2 n)$	
分块查找	$O(\log_2 n) \sim O(n)$	

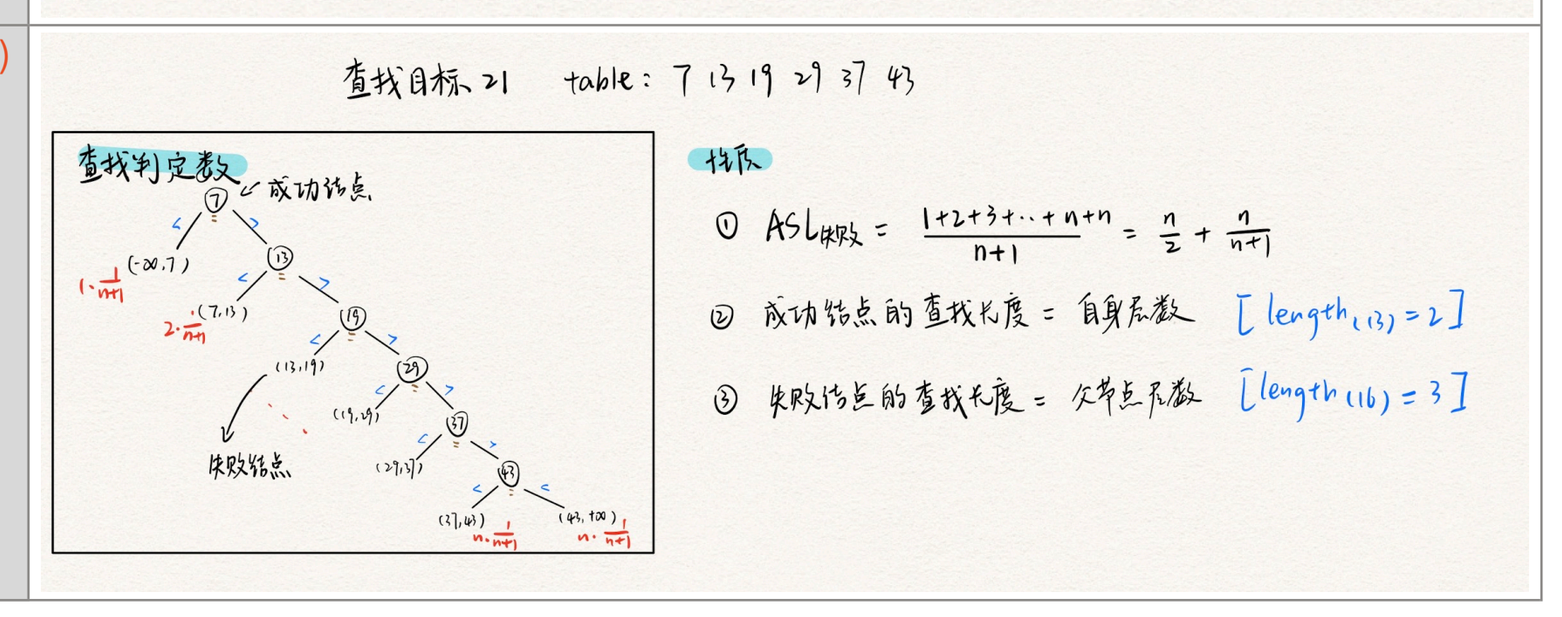
顺序查找、折半查找、分块查找

顺序查找

一般线性表的顺序查找

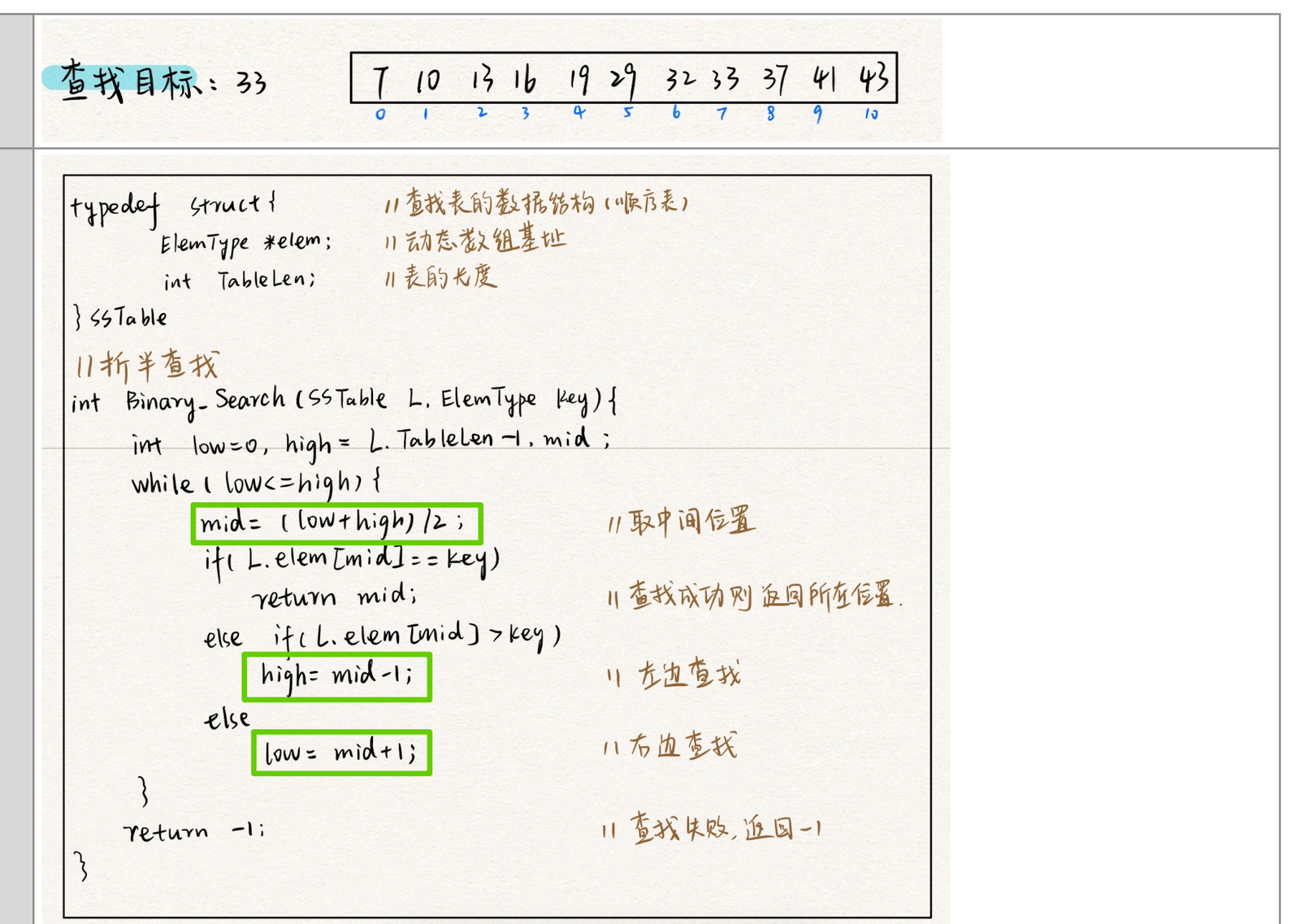


顺序查找的优化(对有序表)

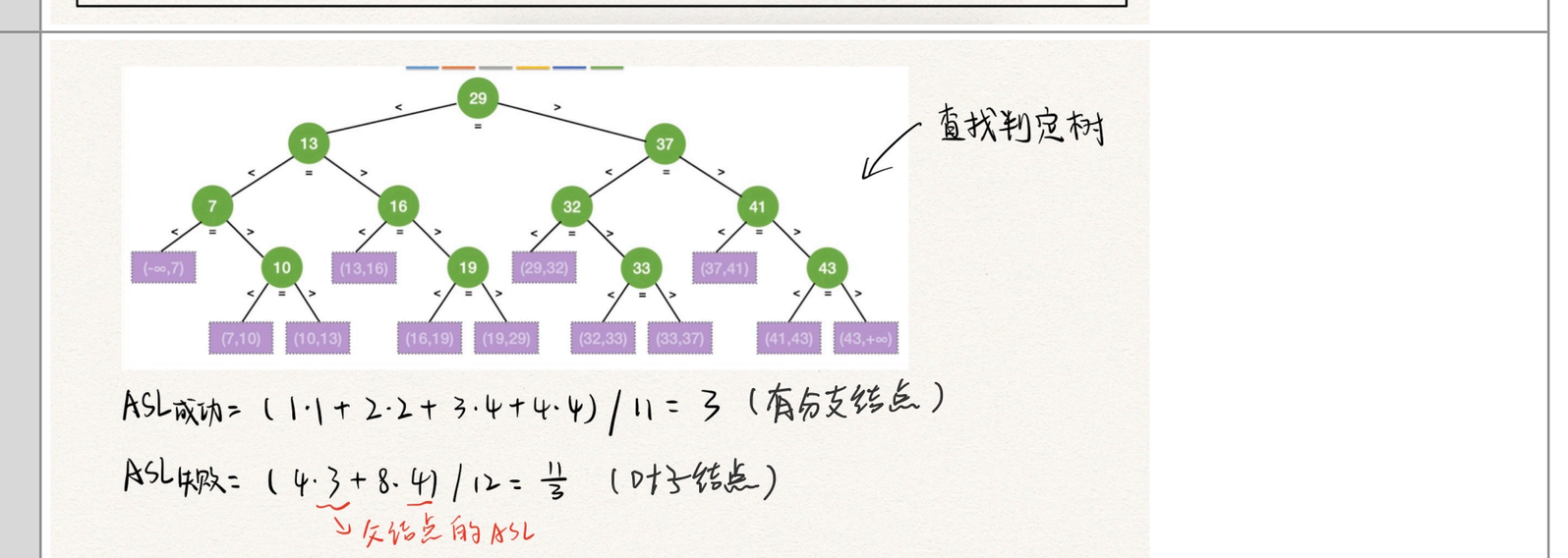


折半查找

查找目标 / 代码



查找效率分析



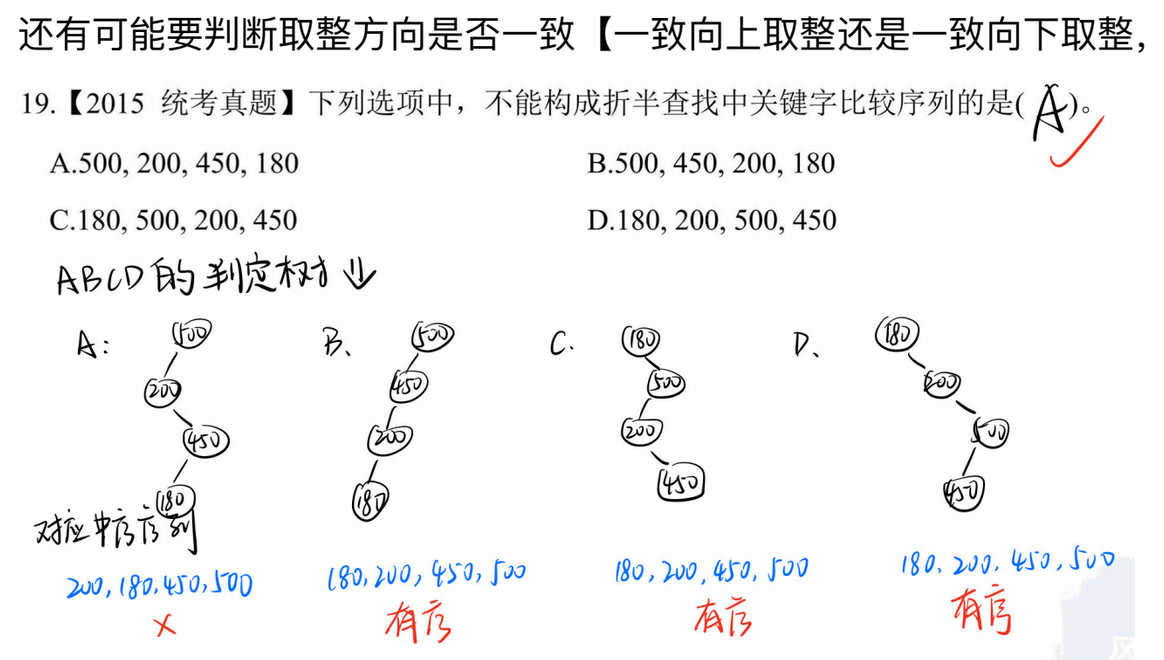
折半查找判定树

- 折半查找树是平衡二叉树（左右子树高度之差不大于1）
- 折半查找判定树可以用二叉排序树来判别，其中序序列是一个有序序列
- 还有可能要判断取整方向是否一致【一致向上取整还是一致向下取整，见17年真题第8题】

19.【2015 统考真题】下列选项中，不能构成折半查找中关键字比较序列的是(A)。

A.500, 200, 450, 180　　B.500, 450, 200, 180

C.180, 500, 200, 450　　D.180, 200, 500, 450

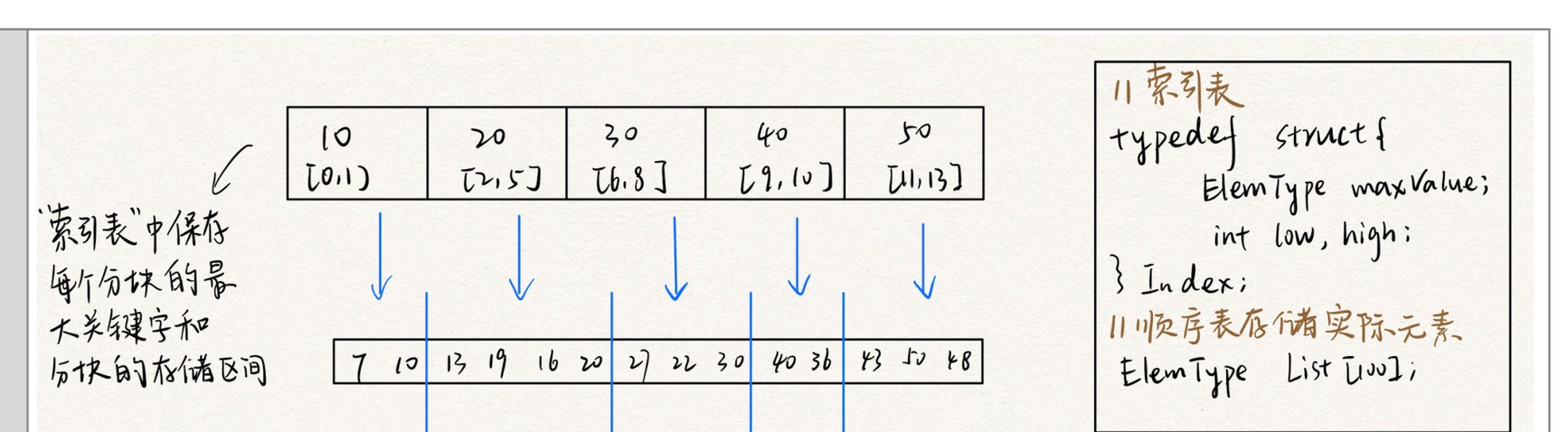


常考结论

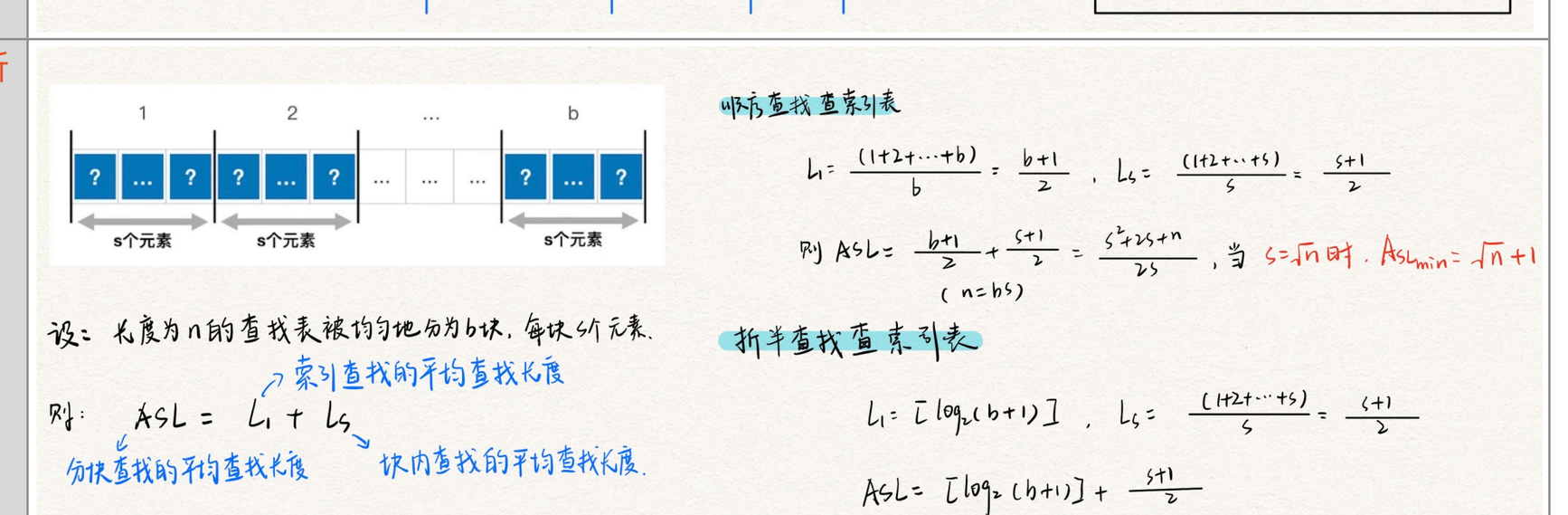
- 树的高度 $h = 向下取整(\log_2 n) + 1 = 向上取整(\log_2 n + 1)$
- 时间复杂度为 $O(\log_2 n)$
- 查找不成功时比较次数最多即为高度h，注意最后一个数也要比较，不要忘记了
- 折半查找和二叉排序树最好的情况时平均查找长度都是 $O(\log_2 n)$
- 二叉排序树最坏的情况形成单支树，查找长度为O(n)

分块查找

定义



效率分析



常考结论

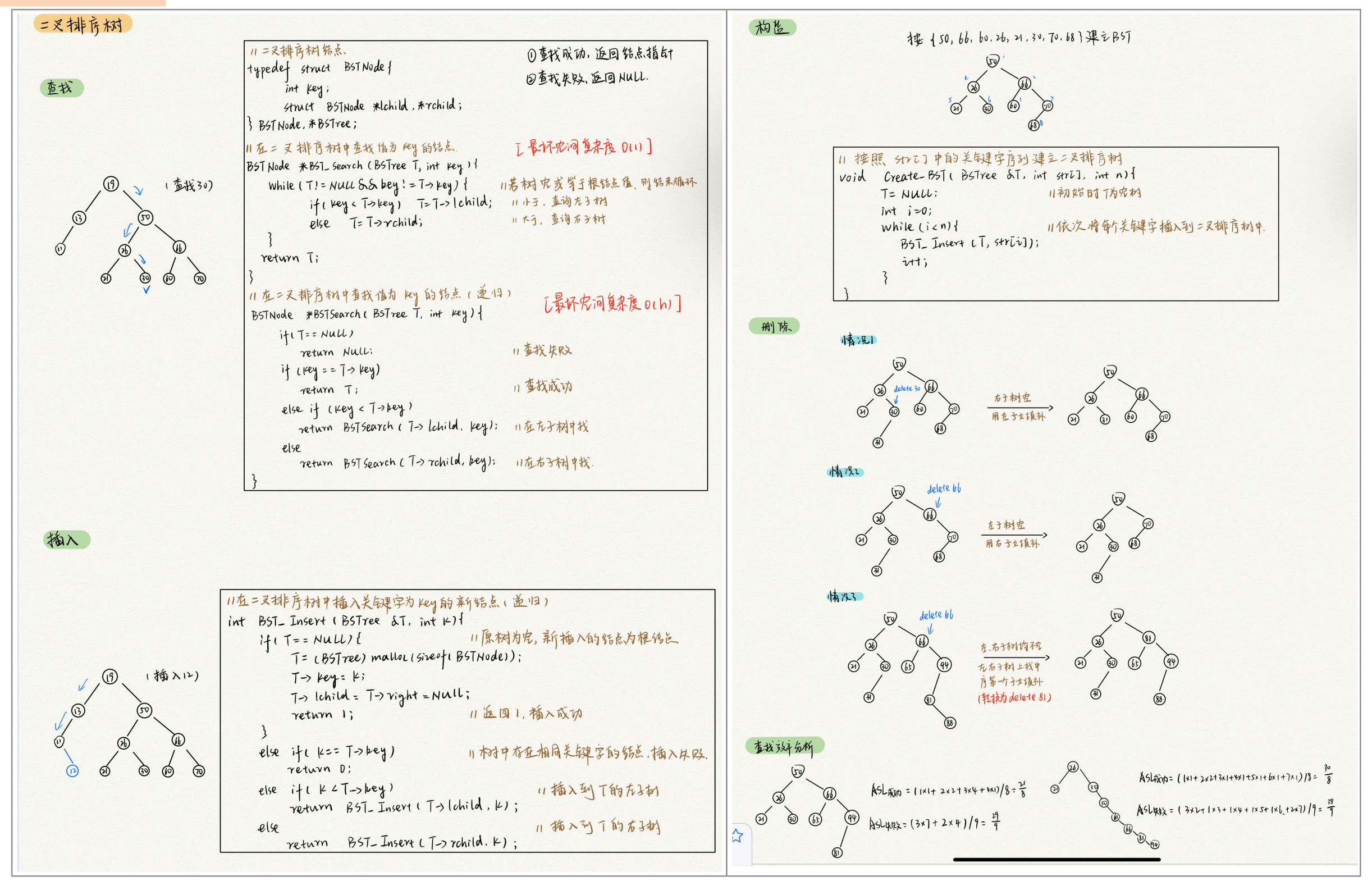
- 顺序查找效率最快时，块数 $= \sqrt{n}$
- 数据分成若干块，每块内数据不必有序，但块间必须有序，但块内最大/最小的数据组成索引块

树型查找【重难点】

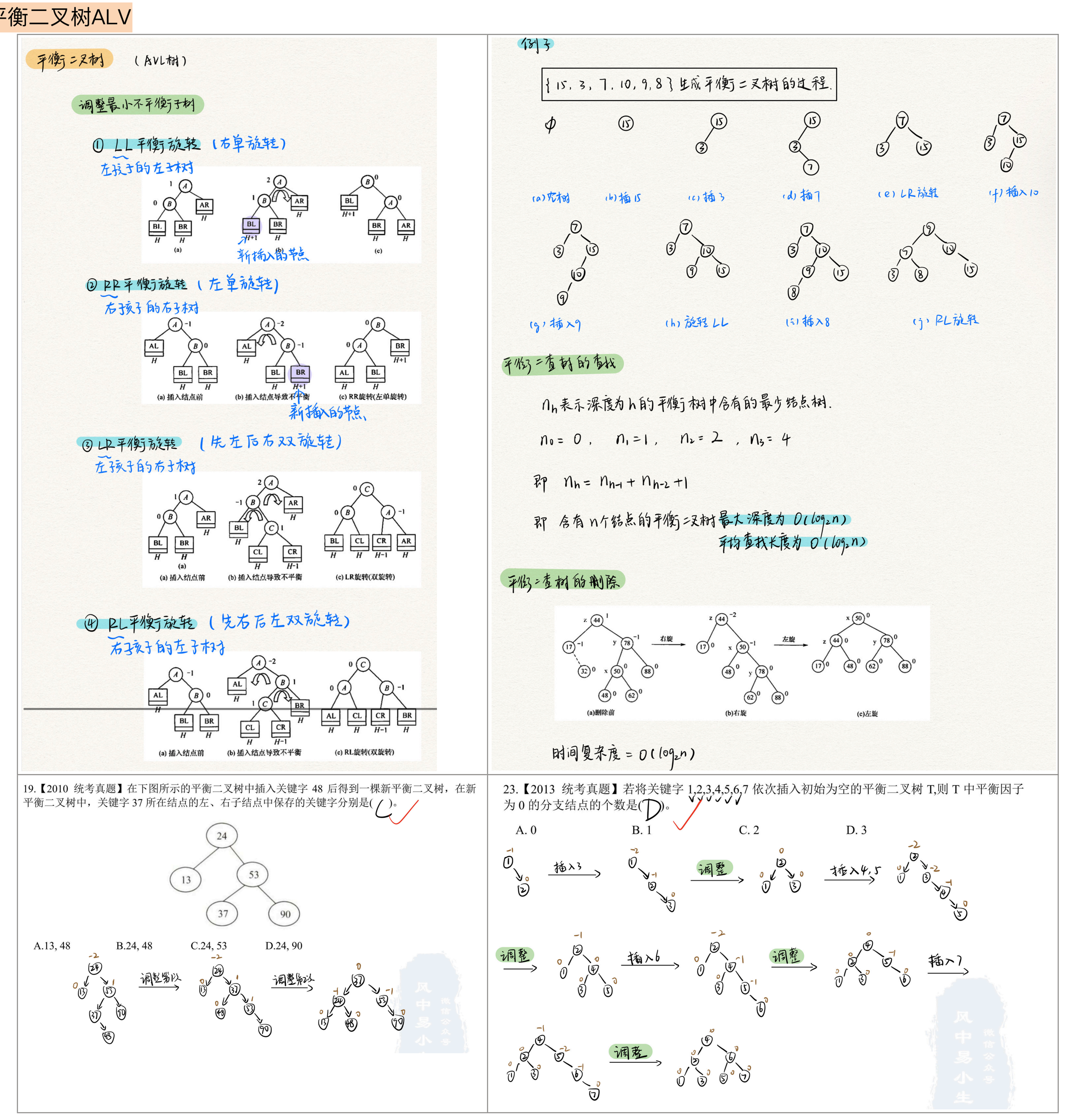
常考结论

二叉排序树	• 当二叉树的叶子节点全部在相邻的两层时，深度最小。理想情况是从第一层到第二层为满二叉树，$h = 向上取整(\log_2 n + 1)$ • 当输入序列是有序序列时，构造的二叉排序树是一支单支树，查找一个关键词最多需要比较n次 • 按中序遍历二叉排序树得到的序列是一个有序序列 • 二叉排序树中，关键字值最大的节点右指针一定为空
平衡二叉树	• 平衡二叉树任意节点的左右子树高度差不超过1（是高度差，不是节点树差！） • 平衡二叉树节点递推公式：$n_0 = 0, n_1 = 1, n_2 = 2, n_h = n_{h-2} + n_{h-1} + 1$ .....(斐波那契数列，n_1是第一层，即根节点) • ALV是高度平衡的二叉树，查找效率最高 • ALV和红黑树查找，删除，插入的平均时间复杂度都相同 • 常考平衡二叉树的四种旋转，凭感觉来就行！
红黑树 (真题没考过)	• 一颗含有n个节点的红黑树的高度至多为 $2\log_2(n+1)$ • 如果一个节点是红色的，则它的父节点和孩子节点都是黑色的 • 从一个结点到其子孙节点的所有路径上包含相同数量的黑节点 • 红黑树是适度平衡的二叉树，查找效率低于平衡二叉树 • 红黑树可能会出现超过2次旋转的操作 • 红黑树的任意节点的左右子树高度之差不超过2倍 • 如果红黑树的所有节点都是黑色的，则它一定是满二叉树

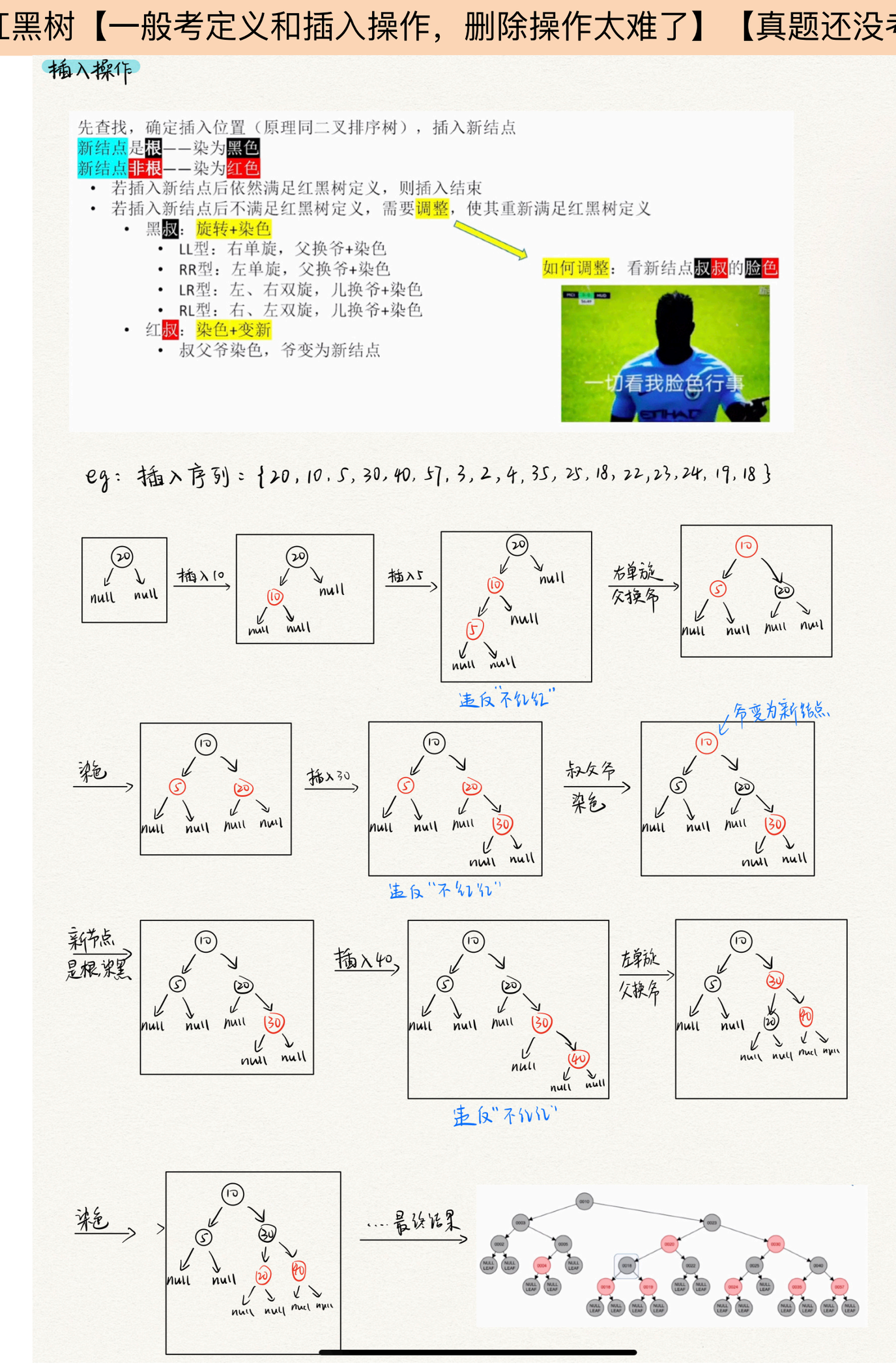
二叉排序树BST



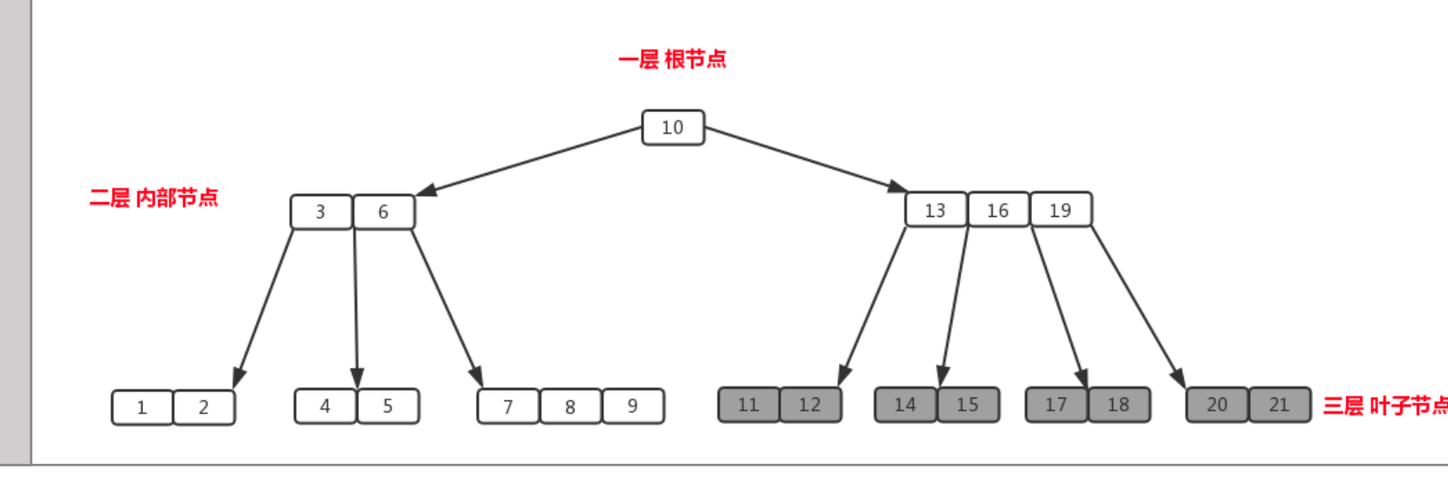
平衡二叉树ALV



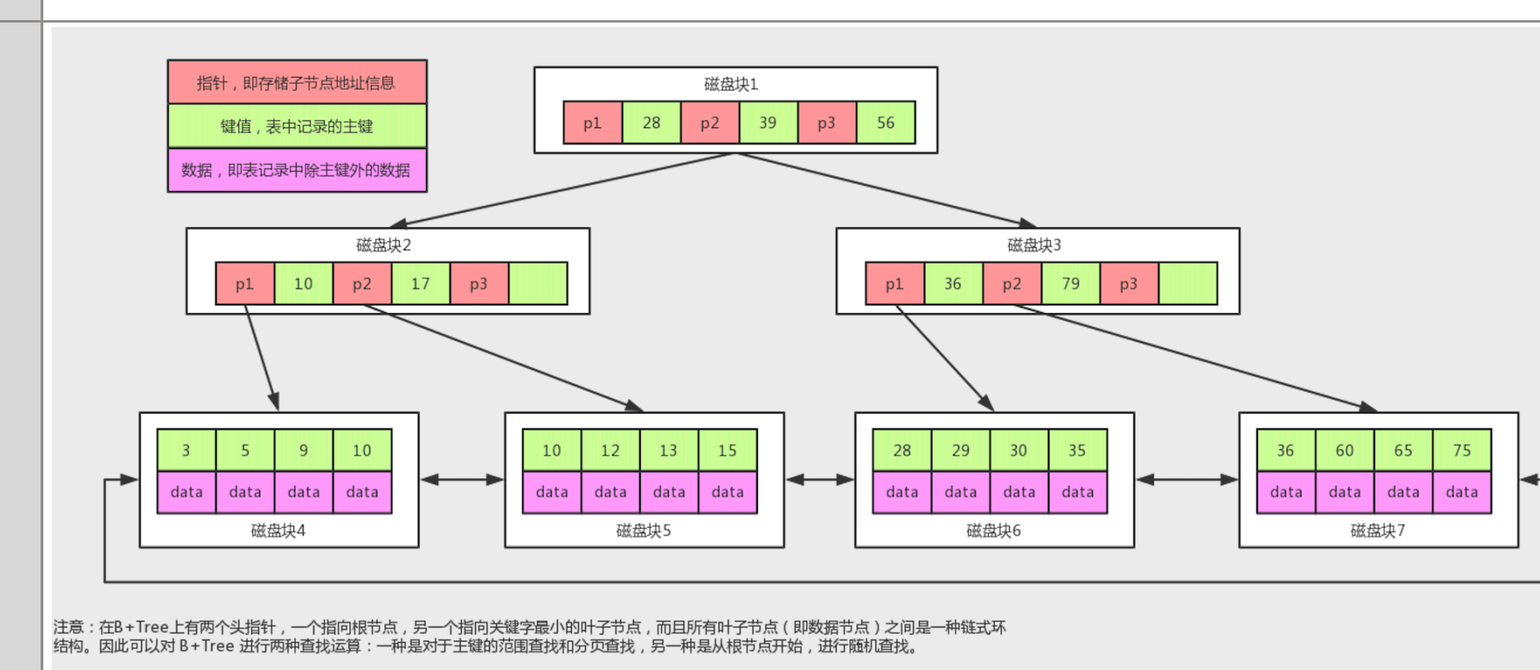
红黑树【一般考定义和插入操作，删除操作太难了】【真题还没考过】



B树(B-tree)

相关概念	• B树是一种自平衡树数据结构，它维护有序数据并允许以对数时间进行搜索，顺序访问，插入和删除 • B树是二叉搜索树的一般化，因为节点可以有两个以上的子节点 • B树非常适合读取和写入相对较大的数据块（如光盘）的存储系统；它通常用于数据库和文件系统
m阶的B树满足的条件	• 每个节点最多只有m个子节点 • 每个非叶子节点(除了根)具有至少⌈ m/2 ⌉个节点 • 根节点最多有m颗子树，至少2个子树 • 具有k个子节点的非叶节点包含k−1个键 • 所有叶节点都在同一层次上
概念图	• B树中一个节点的子节点数目的最大值，用m表示，假如最大值为4，则为4阶，如图 

B+树

B+树的特征	• 有m个子树的中间节点包含有m个元素（B树中是k−1个元素），每个元素不保存数据，只用来索引 • 所有的叶子结点中包含了全部关键字的信息，及指向含有这些关键字记录的指针 • 叶子结点本身依关键字的大小自小而大的顺序链接 • 所有的非终端结点可以看成是索引部分，结点中仅含有其子树根结点中最大（或最小）关键字
概念图	

B树和B+树的对比

	m阶B树	m阶B+树
类比	二叉查找树的进化——>m叉查找树	分块查找的进化——>多级分块查找
关键字与分叉	n个关键字对应n+1个分叉（子树）	n个关键字对应n个分叉
结点包含的信息	所有结点中都包含记录的信息	只有最下层叶子结点才包含记录的信息（可使树更矮）
查找方式	不支持顺序查找。查找成功时，可能停在任何一层结点，查找速度“不稳定”	支持顺序查找。查找成功或失败都会到达最下一层结点，查找速度“稳定”
相同点	除根节点外，最少 $\lceil m/2 \rceil$ 个分叉（确保结点不要太“空”）任何一个结点的子树都要一样高（确保“绝对平衡”）	

散列表

相关概念	• 散列查找一般适用于关键字集合与地址集合之间存在对应关系的情况下的查找 • 散列查找的思想是计算出散列地址来查找，然后比较关键字以确定是否比较成功 • 散列查找的平均查找长度与装填因子有关，与表长无关 • $转填因子 = \frac{表中记录数}{散列表长度}$ • 冲突是不可避免的，与装填因子无关 • 为提高查找效率可以：1. 设计冲突少的散列函数 2. 处理冲突时避免产生堆积现象 【装填因子是固有属性，不可变】 • 产生了堆积即产生了冲突，他对存储效率，散列函数和装填因子没有什么影响，但ASL会随堆积现象而增大
两项基本工作	• 计算位置：构造散列函数确定关键词存储位置 • 解决冲突：应用某种策略解决多个关键词位置相同的问题

计算位置

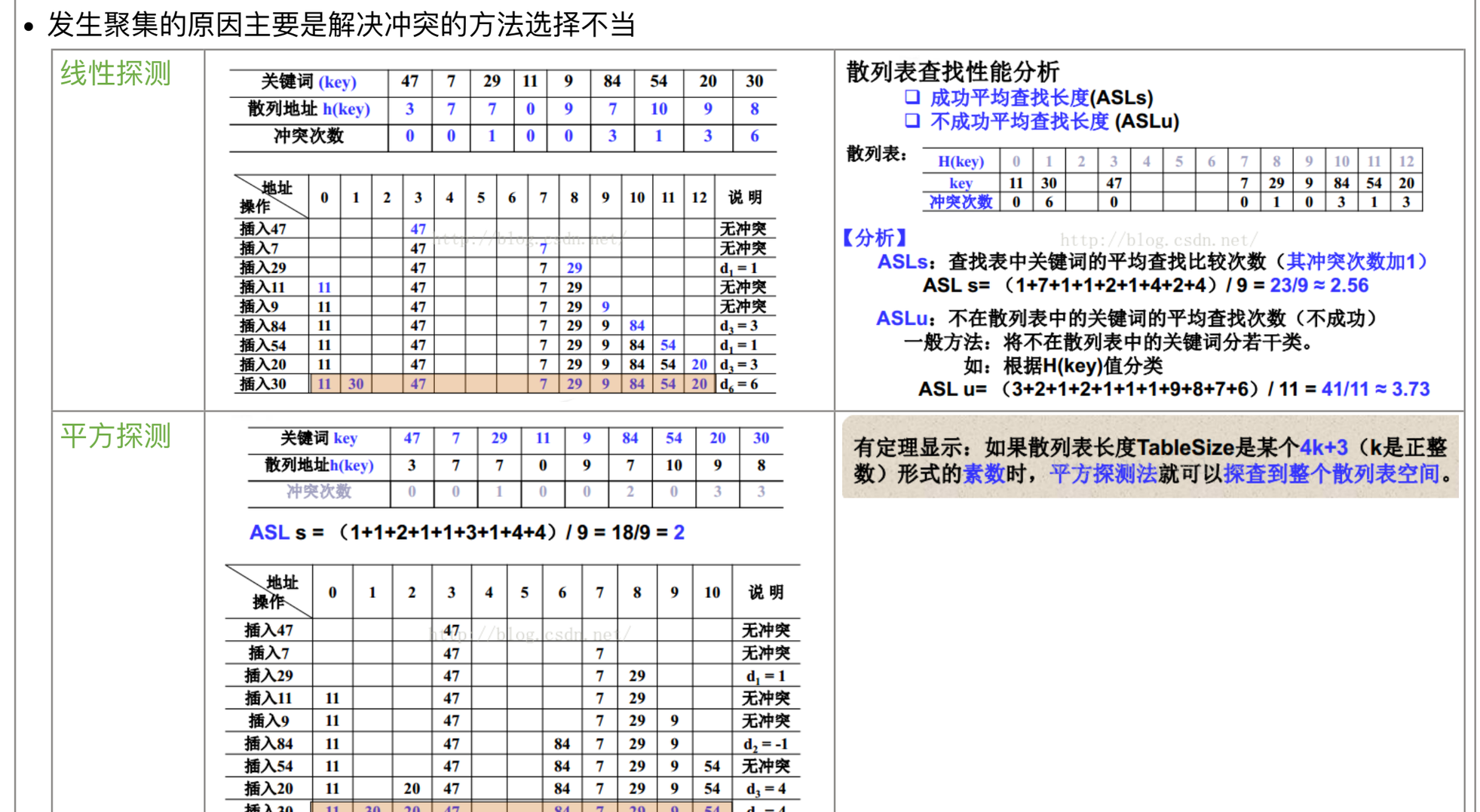
除留余数法	• $h(key) = key\%p$
直接定址法	• $h(key) = a \times key + b$【线性函数】
平方取中法	• key取平方再取中间几位
数字分析法	• 分析数字关键字在各位上的变化情况 • 取比较随机的位作为散列地址 • 如电话号码、身份证号码某几位会比较随机

解决冲突

开放地址法

- 一旦产生了冲突，就按某种规则去寻找另一空地址
- 发生聚集的原因主要是解决冲突的方法选择不当

线性探测 / 平方探测

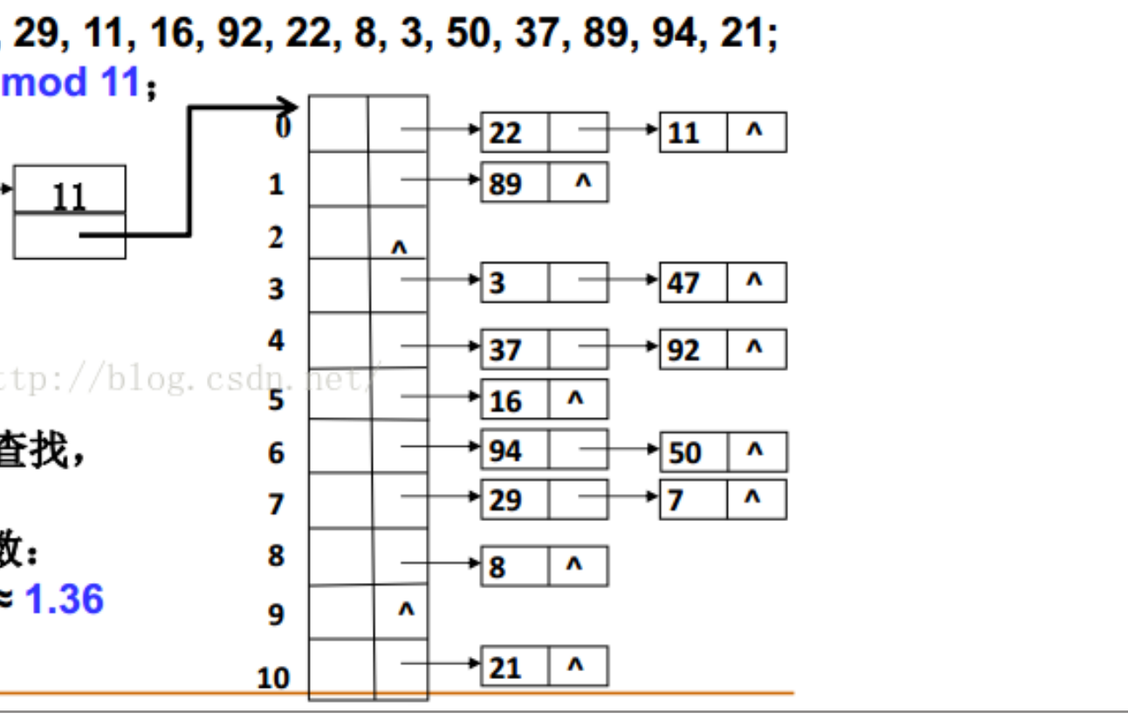


链地址法

- 将相应位置上冲突的所有关键词存储在同一个单链表中

【例】设关键字序列为 47, 7, 29, 11, 16, 92, 22, 8, 3, 50, 37, 89, 94, 21;
散列函数取为：h(key) = key mod 11;
用分离链接法处理冲突。

```
struct HashTbl {
  int TableSize;
  List TheLists;
}H;
```



➤表中有9个结点只需1次查找，
➤5个结点需要2次查找，
➤查找成功的平均查找次数：
ASL s=（9+5*2）/ 14 ≈ 1.36