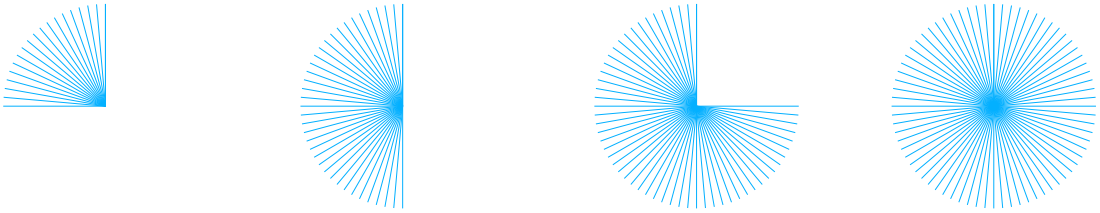
开课吧

计算机硕士

「高频考点」

数据结构



考研

目　录

编写人：田曲申　审核人：于方泽　封面设计：乔宇、刘锦浩、郑悦

勘误信箱：

亲爱的开课吧学员你好，感谢您选择开课吧开启学习之旅。在使用这份学习资料时，希望你能感受到开课吧全体教职人员的用心。经过软件人工三审三校的资料难免有万分之三的概率出现小差错，我们设立勘误激励邮箱，感谢大家和小开共同成长。在使用本资料的同时，如您火眼金睛发现了小错误。可将资料的名称-页码带图反馈到本邮箱：

kaikeba0104@126.com。一经采纳小开感激不尽。

必考点 1——时空复杂度

常见的时间复杂度	级别	典型应用
O(1)	常数级	几条语句
O(logn)	对数级	二分查找
O(n)	线性级	顺序查找
O(nlogn)	线性对数级	归并排序
O(n^2)	平方级	冒泡排序
O(2^n)	指数级	组合问题

必考点 2——顺序表

线性表的顺序存储又称顺序表。它是用一组地址连续的存储单元依次存储线性表中的数据元素，从而使得逻辑上相邻的两个元素在物理位置上也相邻。顺序表的特点是表中元素的逻辑顺序与其物理顺序相同。（顺序表可以简单理解为数组）

顺序表的插入与删除操作主要耗时在对后续元素移动上。

必考点 3——单链表

线性表的链式存储又称单链表，它是指通过一组任意的存储单元来存储线性表中的数据元素。为了建立数据元素之间的线性关系，对每个链表结点，除存放元素自身的信息外，还需要存放一个指向其后继的指针。

单链表的插入、删除与查找操作，主要耗时在查找上。注意链表的插入操作分为头插法和尾插法；同时要注意头结点（虚拟头结点）是否存在的问题。

除了单链表以外，还有双链表，循环链表，静态链表等各种链式数据结构，它们在逻辑结构与物理结构上与单链表相似。在这里还要掌握顺序表与链表的区别。

必考点 4——栈

栈(Stack)是只允许在一端进行插入或删除操作的线性表。首先栈是一种线性表，但限定这种线性表只能在某一端进行插入和删除操作。线性表允许进行插入删除的那一端叫做栈顶，不允许进行插入和删除的另一端叫做栈底。

在栈实现时，物理结构可以选择顺序表（顺序栈），也可以选择链表（链栈）；还有一种两个栈共用同一片内存空间的结构，称为共享栈（对顶栈）。

给定有 n 个入栈元素的序列，出栈序列有 C(2n,n)/(n+1)种（卡特兰数）。

必考点 5——队列

队列(Queue)简称队，也是一种操作受限的线性表，只允许在表的一端进行插入，而在表的另一端进行删除。向队列中插入元素称为入队或进队；删除元素称为出队或离队。这和我们日常生活中的排队是一致的，最早排队的也是最早离队的，其操队头队尾作的特性是先进先出(First In First Out，FIFO)。允许删除的一端叫做队首（队头，front），允许插入的一端叫做队尾(rear/back)。

在队列实现时，物理结构可以选择顺序表（顺序队列，在这里还可实现循环队列），也可以选择链表（链队列）。还有一种特殊的队列——双端队列，它在队首与队尾处均可以执行出队和入队操作。

必考点 6——树的性质

树中的结点数等于所有结点的度数加 1。

度为 m 的树中第 i 层上至多有 m^(i−1)个节点（i≥1）。

高度为 h 的 m 叉树至多有（m^h−1）／（m−1）个节点。

具有 n 个节点的 m 叉树的最小高度为[logm(n(m−1)+1)]。

必考点 7——二叉树的性质

首先要清楚满二叉树与完全二叉树的定义。

非空二叉树上的叶子结点数等于度为 2 的结点数加 1，即 n0（度为 0 的点数量）=n2（度为 2 的点数量）+1；

非空二叉树上第 k 层上至多有 2^(k−1)个节点（k≥1）；

高度为 h 的二叉树至多有 2^h−1 个结点（h≥1);

具有 n 个（n>0）结点的完全二叉树的高度为(log2(n+1))下取整或(log2n)上取整+1。

具有 n 个结点的不同二叉树的数量为 C(2n,n)/(n+1)个（卡特兰数）。

二叉树存储时，可以使用链式存储，也可以使用顺序存储。使用顺序存储时，下标为 x 的结点，左子树存储在下标为 x*2 结点处，右子树存储在下标为 x*2+1 结点处。

必考点 8——二叉树的遍历

先（根）序遍历：先根，再左子树，再右子树；

中（根）序遍历：先左子树，再根，再右子树；

后（根）序遍历：先左子树，再右子树，再根；

层序遍历：按层遍历树

其中三种遍历一般使用递归实现（深搜），也有不实用递归实现的迭代算法，层序遍历一般使用队列实现（广搜）。给定先序与中序或给定中序与后序，可以还原树。

必考点 9——森林与树与二叉树的转换

将树转换为二叉树时，可以使用孩子兄弟表示法：左孩子，右兄弟。在这里，左子树存的是当前结点的第一个子结点，右子树存的是当前结点的下一个兄弟（具有同一个父结点的）结点。

将森林转换为二叉树时，在上述基础上：森林中不同树的根结点互为兄弟。

树有先序遍历和后序遍历，其规则与二叉树的先序遍历和后序遍历一致。先序遍历序列与这棵树相应二叉树的先序序列相同；后序遍历序列与这棵树相应二叉树的中序序列相同。

森林有先序遍历和中序遍历，先序遍历序列为对每一颗树先序遍历连接后的序列；中序遍历序列为对每一颗树后序遍历连接后的序列。

必考点 10——二叉排序树

二叉排序树（也称二叉查找树）或者是一棵空树，或者是具有下列特性的二叉树：

若左子树非空，则左子树上所有结点的值均小于根结点的值。若右子树非空，则右子树上所有结点的值均大于根结点的值。左、右子树也分别是一棵二叉排序树。

根据二叉排序树的定义，左子树结点值＜根结点值＜右子树结点值，所以对二叉排序树进行中序遍历，可以得到一个递增的有序序列。

二叉排序树的查找插入删除操作的平均复杂度为 O(log2n)。二叉排序树在退化为链时，复杂度最坏情况下为 O(n)。

必考点 11——二叉平衡树

为避免树的高度增长过快，降低二叉排序树的性能，规定在插入和删除二叉树结点时，要保证任意结点的左、右子树高度差的绝对值不超过 1，将这样的二叉树称为平衡二叉树(Balanced Binary Tree)，简称平衡树。定义结点左子树与右子树的高度差为该结点的平衡因子，则平衡二叉树结点的平衡因子的值只可能是−1，0 或 1。

因此，平衡二叉树可定义为或者是一棵空树，或者是具有下列性质的二叉树：它的左子树和右子树都是平衡二叉树，且左子树和右子树的高度差的绝对值不超过 1。

在进行插入或删除操作时，可能会导致整棵树不平衡，所以需要进行旋转调整操作（注意这里每次调整的都是最小不平衡子树），大体上分为 4 种情况：

某结点的左子树的左子树导致该结点不平衡：LL 型，直接对当前结点右旋；

某结点的左子树的右子树导致该结点不平衡：LR 型，先对当前结点的左子树左旋，然后对当前结点右旋；

某结点的右子树的右子树导致该结点不平衡：RR 型，直接对当前结点左旋；

某结点的右子树的左子树导致该结点不平衡：RL 型，先对当前结点的右子树右旋，然后对当前结点左旋。

必考点 12——哈夫曼树与哈夫曼编码

在许多应用中，树中结点常常被赋予一个表示某种意义的数值，称为该结点的权。从树的根到任意结点的路径长度（经过的边数）与该结点上权值的乘积，称为该结点的带权路径长度。树中所有叶结点的带权路径长度之和称为该树的带权路径长度。在含有 n 个带权叶结点的二叉树中，其中带权路径长度（WPL）最小的二叉树称为哈夫曼树，也称最优二叉树。

哈夫曼树的构造方法：初始时，所有元素均作为独立的树根存在，接下来选择树根权值最小的两棵树，将他们合并成一棵树，新树根的权值为两棵子树树根权值之和，不断的重复这一过程，最终只剩下一棵树时，构造完毕。哈夫曼树中不会有度为 1 的结点。

从树根出发，对哈夫曼树的左子树编码 0，右子树编码 1（或反过来），最终到达叶子结点的编码即为哈夫曼编码。使用哈夫曼编码会使得最终总编码长度最短。

必考点 13——图的概念与相关属性

1.图是由有穷“点”集和“边”集组成的集合。在图结构中常常将结点称为顶点，边是顶点的有序偶对。若两个顶点之间存在一条边，则表示这两个顶点具有相邻关系。

2.由有方向的边（有向边）组成的图称为有向图，由不区分方向的边（无向边）组成的图称为无向图。

3.在有向图中，通常将边称为弧，含箭头的一端称为弧头，另一端称为弧尾。

4.在无向图中，与顶点 x 相关的边的条数称为顶点 x 的度；在有向图中，指向顶点 x 的边的条数称为顶点 x 的入度，由顶点 x 发出的边的条数称为顶点 x 的出度。

5.若有向图中有 n 个顶点，则最多有 $n(n-1)$ 条边（图中任意两个顶点都有两条边相连），将具有 $n(n-1)$ 条边的有向图称为有向完全图。若无向图中有 n 个顶点，则最多有 $n(n-1)/2$ 条边（任意两个顶点之间都有一条边），将具有 $n(n-1)/2$ 条边的无向图称为无向完全图（注意这里的前提为没有重边）。

6.在一个图中，路径为相邻顶点序偶所构成的序列。路径长度是指路径上边的累加。

7.序列中顶点不重复出现的路径称为简单路径。

8.若一条路径中第一个顶点和最后一个顶点相同，则这条路径是一条回路，也称为回路或环。

9.图中每条边都可以附有一个对应的数，这种与边相关的数称为权。权可以表示从一个顶点到另一个顶点的距离或者花费的代价。边上带有权的图称为带权图，也称为网。

10.在无向图中，若从顶点 v 到顶点 w 有路径存在，则称 v 和 w 是连通的。若图 G 中任意两个顶点都是连通的，则称图 G 为连通图，否则称为非连通图。无向图中的极大连通子图称为连通分量。

11.在有向图中，若从顶点 v 到顶点 w 和从顶点 v 到顶点 w 之间都有路径，则称这两个顶点是强连通的。若图中任何一对顶点都是强连通的，则称此图为强连通图。有向图中的极大强连通子图称为有向图的强连通分量。

必考点 14——图的存储

邻接矩阵

邻接矩阵是表示顶点之间相邻关系的矩阵 A。

A[i][j]=1 表示顶点 i 与顶点 j 邻接，即它们之间存在边或者弧。A[i][j]=0 表示顶点 i 与顶点 j 不邻接。

邻接矩阵是图的顺序存储结构，由邻接矩阵的行数或列数可知图中的顶点数。对于无向图，邻接矩阵是对称的，矩阵中 1 的个数为图中总边数的两倍，矩阵中第 i 行或第 i 列的元素之和即为顶点 i 的度。对于有向图，矩阵中 1 的个数为图的边数，矩阵中第 i 行的元素之和即为顶点 i 的出度，第 j 列元素之和即为顶点 j 的入度。对于有权图，邻接矩阵中存储边权。

邻接表

邻接表是图的一种链式存储结构。所谓邻接表，是指对图中的每个顶点 x，建立一个单链表，链表中存储以顶点 x 为起点的所有边，每一个链表节点代表一条边。当一个图为稀疏图时，使用邻接矩阵显然要浪费大量的存储空间，而邻接表结合了顺序存储和链式存储方法，大大减少了这种不必要的浪费。使用邻接表存图可以快速的找到所有以顶点 x 为起点的所有边（只需遍历它的链表即可），而邻接矩阵做同样的操作需要遍历一整行元素。与邻接表类似的结构还有邻接多重表。

与邻接表相似，还有一种叫做十字链表的结构，不止存储了同起点的所有边，还存储了同终点的所有边。

必考点 15——图的遍历

深度优先遍历

图的深度优先搜索遍历（DFS）类似于二叉树的先序遍历。它的基本思想是：首先访问出发点 x，并将其标记为已访问过；然后选取与 x 邻接的未被访问的任意一个顶点 v，并访问它；再选取与 v 邻接的未被访问的任一顶点并访问，以此重复进行。当一个顶点所有的邻接顶点都被访问过时，则依次退回到最近被访问过的顶点，若该顶点还有其他邻接顶点未被访问，则从这些未被访问的顶点中取一个并重复上述访问过程，直至图中所有顶点都被访问过为止。一般使用递归来实现。

广度优先遍历

图的广度优先搜索遍历（BFS）类似于树的层序遍历。它的基本思想是：首先访问出发点 x，然后选取与 x 邻接的全部顶点 v1...vn 进行访问，再依次访问与 v1...vn 邻接的全部顶点（已经访问过的除外），以此类推，直到所有顶点都被访问过为止。一般使用一个辅助队列来实现。

必考点 16——最短路径算法

Dijkstra 算法

Dijkstra 算法为单源最短路径算法，可以求解某一顶点 x0 到其他各顶点的最短路径。

设有两个顶点集合 S 和 T，集合 S 中存放图中已找到最短路径的顶点，集合 T 存放图中剩余顶点。初始状态时，集合 S 中只包含源点 x0 ，然后不断从集合 T 中选取到顶点 x0 路径长度最短的顶点 xi 并入到集合 S 中。集合 S 每并入一个新的顶点 xi，都要修改顶点 x0 到集合 T 中顶点的最短路径长度值。不断重复此过程，直到集合 T 的顶点全部并入到 S 中为止。在实现过程中，可以使用堆优化。

Floyd 算法

Floyd 算法为多源最短路径算法，可以求解任意顶点到任意顶点的最短路径。

Floyd 一般使用邻接矩阵进行运算，不断尝试经过某顶点 i 中转（先从 j 到 i，再从 i 到 k）去更新从顶点 j 到顶点 k 的路径。

必考点 17——最小生成树

一个连通图的生成树包含图的所有顶点，并且只含尽可能少的边。对于生成树来说，若砍去它的一条边，则会使生成树变成非连通图；若给它增加一条边，则会形成图中的一条回路。对于同一个图来说，生成树不同，每棵树的边权总和不同，在所有生成树中，边权总和最小的生成树，称为最小生成树。它有如下性质：

1．最小生成树的边数为顶点数减 1。

2．最小生成树不是唯一的。

3．最小生成树的边的权值之和总是唯一的。

Prim 算法（以点为基准）

从图中任意取出一个顶点，把它当成一棵树，然后从与这棵树相接的边中选取一条最短（权值最小）的边，并将这条边及其所连接的顶点也并入这棵树中，此时得到了一棵有两个顶点的树。然后从与这棵树相接的边中选取一条最短的边，并将这条边及其所连顶点并入当前树中，得到一棵有 3 个顶点的树。以此类推，直到图中所有顶点都被并入树中为止，此时得到的生成树就是最小生成树。Prim 算法的执行非常类似于寻找图的最短路径的 Dijkstra 算法。

Kruskal 算法（以边为基准）

每次找出候选边中权值最小的边，将该边并入生成树中。重复此过程直到所有顶点都并在一起为止。一般在实现时需要使用并查集记录当前联通关系。Kruskal 算法思路比较简单，在考试中需要求最小生成树时一般都使用此算法。

必考点 18——拓扑排序

AOV 网

活动在顶点上的网（Activity On Vertex network，AOV）是一种可以形象地反映出整个工程中各个活动之间的先后关系的有向图。考试中，只要知道 AOV 网是一种以顶点表示活动、以边表示活动的先后次序且没有回路的有向图即可。因为 AOV 网有实际意义，所以出现回路就代表一项活动以自己为前提，这显

然违背实际。

拓扑排序算法

对一个有向无环图 G 进行拓扑排序，是将 G 中所有顶点排成一个线性序列，使得图中任意一对顶点 u 和 v，若存在由 u 到 v 的路径，则在拓扑排序序列中一定是 u 出现在 v 的前边。求解拓扑排序的过程如下：从有向图中选择一个没有前驱（入度为 0）的顶点输出，删除该顶点，并且删除从该顶点发出的全部边，重复上述两步，直到剩余的图中不存在没有前驱的顶点为止。对于一个图来说，拓扑排序可能不唯一。拓扑排序在实现时可以采用多种不同的方式，如深搜、广搜等。

必考点 19——关键路径

AOE 网

对于活动在边上的网（Activity On Edge network，AOE）可以和 AOV 网对比着来记忆。

两者的相同点：都是有向无环图。

两者的不同点： AOE 网的边表示活动，边有权值，边代表活动持续时间；顶点表示事件，事件是图中新活动开始或者旧活动结束的标志。 AOV 网的顶点表示活动，边无权值，边代表活动之间的先后关系。

对于一个表示工程的 AOE 网，只存在一个入度为 0 的顶点，称为源点，表示整个工程的开始；也只存在一个出度为 0 的顶点，称为汇点，表示整个工程的结束。

关键路径

在 AOE 网中，从源点到汇点的所有路径中，具有最大路径长度的路径称为关键路径。完成整个工期的最短时间就是关键路径长度所代表的时间。关键路径上的活动称为关键活动。关键路径是个特殊的概念，它既代表了一个最短又代表了一个最长，它是图中的最长路径，又是整个工期所完成的最短时间。

必考点 20——插入类排序

直接插入排序

每趟将一个待排序的关键字按照其值的大小插入到已经排好的部分有序序列的适当位置上，直到所有待排关键字都被插入到有序序列中为止。

折半插入排序

折半插入排序的基本思想和直接插入排序类似，区别是查找插入位置的方法不同，折半插入排序是采用折半查找法来查找插入位置的。

希尔排序

希尔排序又叫作缩小增量排序，其本质还是插入排序，只不过是将待排序列按某种规则分成几组，分别对这几组进行直接插入排序。这个规则的体现就是增量的选取，如果增量为 1，就是直接插入排序。

必考点 21——交换类排序

冒泡排序（起泡排序）

冒泡排序是通过一系列的“交换”动作完成的。首先第一个关键字和第二个关键字比较，如果第一个大，则二者交换，否则不交换；然后第二个关键字和第三个关键字比较，如果第二个大，则二者交换，否则不交换…... 一直按这种方式进行下去，最终最大的那个关键字被交换到了最后，一趟冒泡排序完成。经过多趟这样的排序，最终使整个序列有序。

快速排序

快速排序也是“交换”类的排序，它通过多次划分操作实现排序。以升序为例，其执行流程可以概括为：每一趟选择当前所有子序列中的一个关键字（通常是第一个）作为枢轴，将子序列中比枢轴小的移到枢轴前边，比枢轴大的移到枢轴后边；当本趟所有子序列都被枢轴以上述规则划分完毕后会得到新的一组更短的子序列，它们成为下一趟划分的初始序列集。当所有元素都有序时排序结束。通常用递归实现。

必考点 22——选择类排序

简单选择排序

简单选择排序采用最简单的选择方式，从头至尾顺序扫描序列，找出最小的一个关键字，和第一个关键字交换，接着从剩下的关键字中继续这种选择和交换，最终使序列有序。

堆排序

堆是一种数据结构，可以把堆看成一棵完全二叉树，这棵完全二叉树满足：任何一个非叶结点的值都不大于（或不小于）其左右孩子结点的值。若父亲大孩子小，则这样的堆叫作大顶堆；若父亲小孩子大，则这样的堆叫作小顶堆。根据堆的定义知道，代表堆的这棵完全二叉树的根结点的值是最大（或最小）的，因此将一个无序序列调整为一个堆，就可以找出这个序列的最大（或最小）值，然后将找出的这个值交换到序列的最后（或最前），这样，有序序列关键字增加 1 个，无序序列中关键字减少 1 个，对新的无序序列重复这样的操作，就实现了排序。这就是堆排序的思想。

必考点 23——归并排序与基数排序

归并排序

归并排序可以看作一个分而治之的过程。先将整个序列分为两半，对每一半分别进行归并排序，将得到两个有序序列，然后将这两个序列合并成一个序列即可。一般用递归实现。

基数排序

基数排序的思想是“多关键字排序”。基数排序有两种实现方式：第一种叫作最高位优先，即先按最高位排成若干子序列，再对每个子序列按次高位排序。第二种叫作最低位优先，这种方式不必分成子序列，每次排序全体关键字都参与。最低位可以优先这样进行，不通过比较，而是通过“分配”和“收集 ”。

必考点 24——内部排序的对比

排序算法	时间复杂度			空间复杂度	是否稳定
	最好	平均	最坏		
直接插入排序	O(N)	O(N^2)	O(N^2)	O(1)	是
冒泡排序	O(N)	O(N^2)	O(N^2)	O(1)	是
选择排序	O(N^2)	O(N^2)	O(N^2)	O(1)	否
希尔排序	O(N^k) k∈[1,2]			O(1)	否
快速排序	O(NlogN)	O(NlogN)	O(N^2)	O(logN)	否
堆排序	O(NlogN)	O(NlogN)	O(NlogN)	O(1)	否
归并排序	O(NlogN)	O(NlogN)	O(NlogN)	O(N)	是
基数排序	O(kN)			O(k+N)	是

必考点 25——线性查找

顺序查找

顺序查找法是一种最简单的查找方法。它的基本思路是：从表的一端开始，顺序扫描线性表，依次将扫描到的关键字和给定值 k 比较，若当前扫描的关键字与 k 相等，则查找成功；若扫描结束后，仍未发现关键字等于 k 的记录，则查找失败。由以上可知，顺序查找法对于顺序表和链表都是适用的。平均查找次数为 O(n/2)。

折半查找

折半查找要求线性表是有序的，即表中记录按关键字有序。首先确定查找区间，根据查找的左右区间，求出一个区间的中间值 mid，若 mid 的关键字等于 k，则查找成功，否则根据 mid 的关键字与 k 的大小关系，确定 k 在区间的前半还是后半，进而调整查找区间，并重复上述过程，若查找区间为空，则查找失败。平均查找次数为 O(log2n)。

折半查找的过程可以用二叉树来表示。把当前查找区间中的中间位置上的记录作为树根，左子表和右子表中的记录分别作为根的左子树和右子树，由此得到的二叉树称为描述折半查找的判定树。

必考点 26——B 树与 B+树

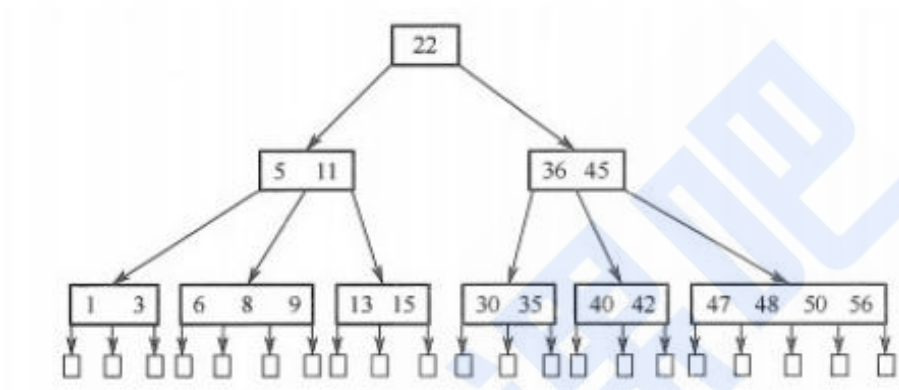
B 树

B 树，又称多路平衡查找树，B 树中所有结点的孩子个数的最大值称为 B 树的阶，通常用 m 表示。一棵 m 阶 B 树或为空树，或为满足如下特性的 m 叉树：

1．树中每个结点至多有 m 棵子树，即至多含有 m−1 个关键字；

2．若根结点不是终端结点，则至少有两棵子树；

3．除根结点外的所有非叶结点至少有 m/2（上取整）棵子树，即至少含有 m/2（上取整）−1 个关键字；



4．所有非叶结点的结构如下：[n,P0,K1,P1,K2,P2.....Kn,Pn]，其中 Ki（1<=i<=n）为节点的关键字，且满足 K1<K2<...Kn；Pi（i=0,1.....n）为指向子树根结点的指针，且指针 Pi−1 所指子树中所有结点的关键字均小于 Ki，Pi 所指子树中所有结点的关键字均大于 Ki−1，n 为结点中关键字的个数；

5．所有的叶结点都出现在同一层次上，并且不带信息（可以视为外部结点或类似于折半查找判定树的查找失败结点，实际上这些结点不存在，指向这些结点的指针为空）。

B 树是所有结点的平衡因子均等于 0 的多路平衡查找树。B 树中的大部分操作所需的磁盘存取次数与 B 树的高度成正比。

B+树

B+树是应数据库所需而出现的一种 B 树的变形树。一棵 m 阶的 B+树需满足下列条件:

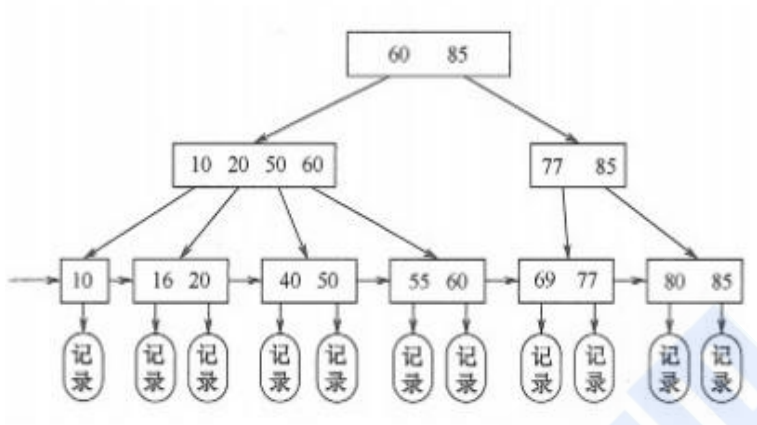
1．每个分支结点最多有 m 棵子树（孩子结点）。

2．非叶根结点至少有两棵子树，其他每个分支结点至少有 m/2（上取整）棵子树。

3．结点的子树个数与关键字个数相等。

4．所有叶结点包含全部关键字及指向相应记录的指针，叶结点中将关键字按大小顺序排列，并且相邻叶结点按大小顺序相互链接起来。

5．所有分支结点（可视为索引的索引）中仅包含它的各个子结点（即下一级的索引块）中关键字的最大值及指向其子结点的指针。



m 阶的 B+树与 m 阶的 B 树的主要差异

1．在 B+树中，具有 n 个关键字的结点只含有 n 棵子树，即每个关键字对应一棵子树；而在 B 树中，具有 n 个关键字的结点含有 n+1 棵子树。

2．在 B+树中，每个结点（非根内部结点）的关键字个数 n 的范围是 m/2（上取整）<=n<=m（根节点：1<=n<=m)；在 B 树中，每个结点 (非根内部结点) 的关键字个数 n 的范围是 m/2 (上取整) −1<=n<=m（根节点：1<=n<=m−1）。

3．在 B+树中，叶结点包含信息，所有非叶结点仅起索引作用，非叶结点中的每个索引项只含有对应子树的最大关键字和指向该子树的指针，不含有该关键字对应记录的存储地址。

4．在 B+树中，叶结点包含了全部关键字，即在非叶结点中出现的关键字也会出现在叶结点中；而在 B 树中，叶结点包含的关键字和其他结点包含的关键字是不重复的。

必考点 27——散列表（哈希表）：

对于考研要求范围内的散列（Hash）表这部分，要记住这么一句话：根据给定的关键字来计算出关键字在表中的地址。这句话就是贯穿于整个散列表的思想，也是 Hash 表和之前讲过的其他查找表的不同之处。在其他的查找表中，关键字的地址跟关键字之间不存在确定的关系；而在 Hash 表中，关键字和关键字的地址是有确定关系的。这种关系可以用 Hash 函数 H 来表示。例如，关键字为 key，则 H(key）称为 Hash 地址，就是 key 在查找表中的地址。

考试中最常用的 Hash 函数构造方法：

1．直接定址法。取关键字或关键字的某个线性函数为 Hash 地址。

2．除留余数法。取关键字被某个不大于 Hash 表表长 m 的数 p 除后所得的余数为 Hash 地址。

考试中最常用的 Hash 冲突处理方法：

1．开放地址法。包含最常见的线性探测法、再散列法。

2．拉链法。将同一哈希值的元素存储在链表中。

开课吧

使 命

让每个人都能公平和便利地获取优质教育服务，并实现可持续职业成长

愿 景

打造全球顶尖的人才科技公司，创办一所国际知名大学，成为令人尊敬的企业

价值观

关心并热爱、只为赋能人才、极致敢为、坦诚开放、拥抱变化、始终创业