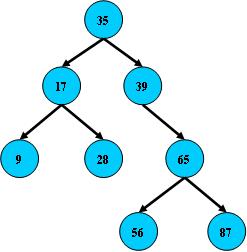
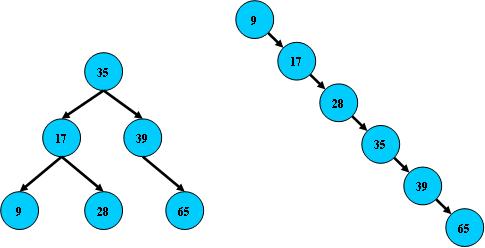
Atitit btree 搜索原理



       B树的搜索，从根结点开始，如果查询的关键字与结点的关键字相等，那么就命中；否则，如果查询关键字比结点关键字小，就进入左儿子；如果比结点关键字大，就进入右儿子；如果左儿子或右儿子的指针为空，则报告找不到相应的关键字；

但B树在经过多次插入与删除后，有可能导致不同的结构：



   右边也是一个B树，但它的搜索性能已经是线性的了；同样的关键字集合有可能导致不同的树结构索引；所以，使用B树还要考虑尽可能让B树保持左图的结构，和避免右图的结构，也就是所谓的“平衡”问题；

       实际使用的B树都是在原B树的基础上加上平衡[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "https://blog.csdn.net/andyzhaojianhui/article/details/_blank)，即“平衡二叉树”；如何保持B树结点分布均匀的平衡算法是平衡二叉树的关键；平衡算法是一种在B树中插入和删除结点的策略；

****小结****

       B树：二叉树，每个结点只存储一个关键字，等于则命中，小于走左结点，大于走右结点；

       B-树：多路搜索树，每个结点存储M/2到M个关键字，非叶子结点存储指向关键字范围的子结点；

       所有关键字在整颗树中出现，且只出现一次，非叶子结点可以命中；

       B+树：在B-树基础上，为叶子结点增加链表指针，所有关键字都在叶子结点中出现，非叶子结点作为叶子结点的索引；B+树总是到叶子结点才命中；

       B\*树：在B+树基础上，为非叶子结点也增加链表指针，将结点的最低利用率从1/2提高到2/3；

BTree,B-Tree,B+Tree,B\_Tree都是什么 - andyzhaojianhui的专栏 - CSDN博客.mhtml