树表查找

二叉查找树

【基本思想】

二叉查找树是先对待查找的数据进行生成树，确保树的左分支的值小于右分支的值，然后在就行和每个节点的父节点比较大小，查找最适合的范围。 这个算法的查找效率很高，但是如果使用这种查找方法要首先创建树。

二叉查找树（BinarySearch Tree，也叫二叉搜索树，或称二叉排序树Binary Sort Tree）或者是一棵空树，或者是具有下列性质的二叉树：

1）若任意节点的左子树不空，则左子树上所有结点的值均小于它的根结点的值；

2）若任意节点的右子树不空，则右子树上所有结点的值均大于它的根结点的值；

3）任意节点的左、右子树也分别为二叉查找树。

二叉查找树性质：对二叉查找树进行中序遍历，即可得到有序的数列。

【复杂度分析】

它和二分查找一样，插入和查找的时间复杂度均为O(logn)，但是在最坏的情况下仍然会有O(n)的时间复杂度。原因在于插入和删除元素的时候，树没有保持平衡（比如形成的树没有分支）。

2-3查找树（2-3 Tree）

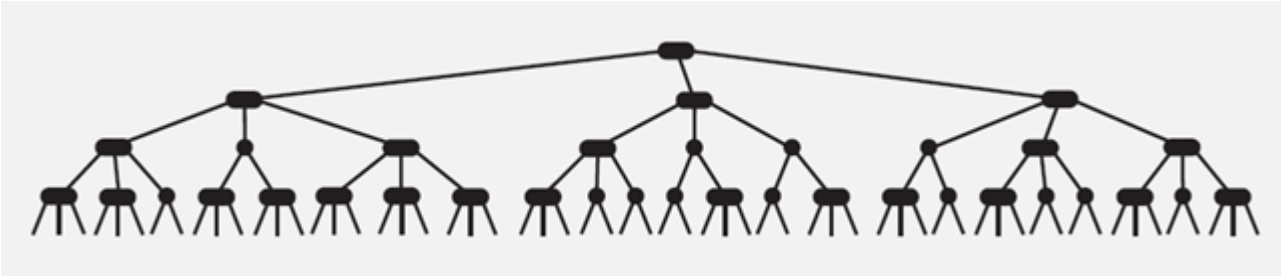
【基本思想】

类似与二分查找树，但父亲结点只能存在空子树，两个子树或者三个子树的情况。存在三个孩子的父亲结点有两个key值来分割三个子树；存在两个孩子的父亲结点有一个key值来分割两个子树，右子树的值总大于左子树的值（或者全部小于）。

【查找树的性质】

1）如果中序遍历2-3查找树，就可以得到排好序的序列；

2）在一个完全平衡的2-3查找树中，根节点到每一个为空节点的距离都相同。（这也是平衡树中“平衡”一词的概念，根节点到叶节点的最长距离对应于查找算法的最坏情况，而平衡树中根节点到叶节点的距离都一样，最坏情况也具有对数复杂度。



【复杂度分析】

2-3树的查找效率与树的高度是息息相关的。

- 在最坏的情况下，也就是所有的节点都是2-node节点，查找效率为lgN
- 在最好的情况下，所有的节点都是3-node节点，查找效率为log3N约等于0.631lgN

红黑树（Red-Black Tree）

2-3查找树能保证在插入元素之后能保持树的平衡状态，最坏情况下即所有的子节点都是2-node，树的高度为lgn，从而保证了最坏情况下的时间复杂度。但是2-3树实现起来比较复杂，于是就有了一种简单实现2-3树的数据结构，即红黑树（Red-Black Tree）。

【基本思想】

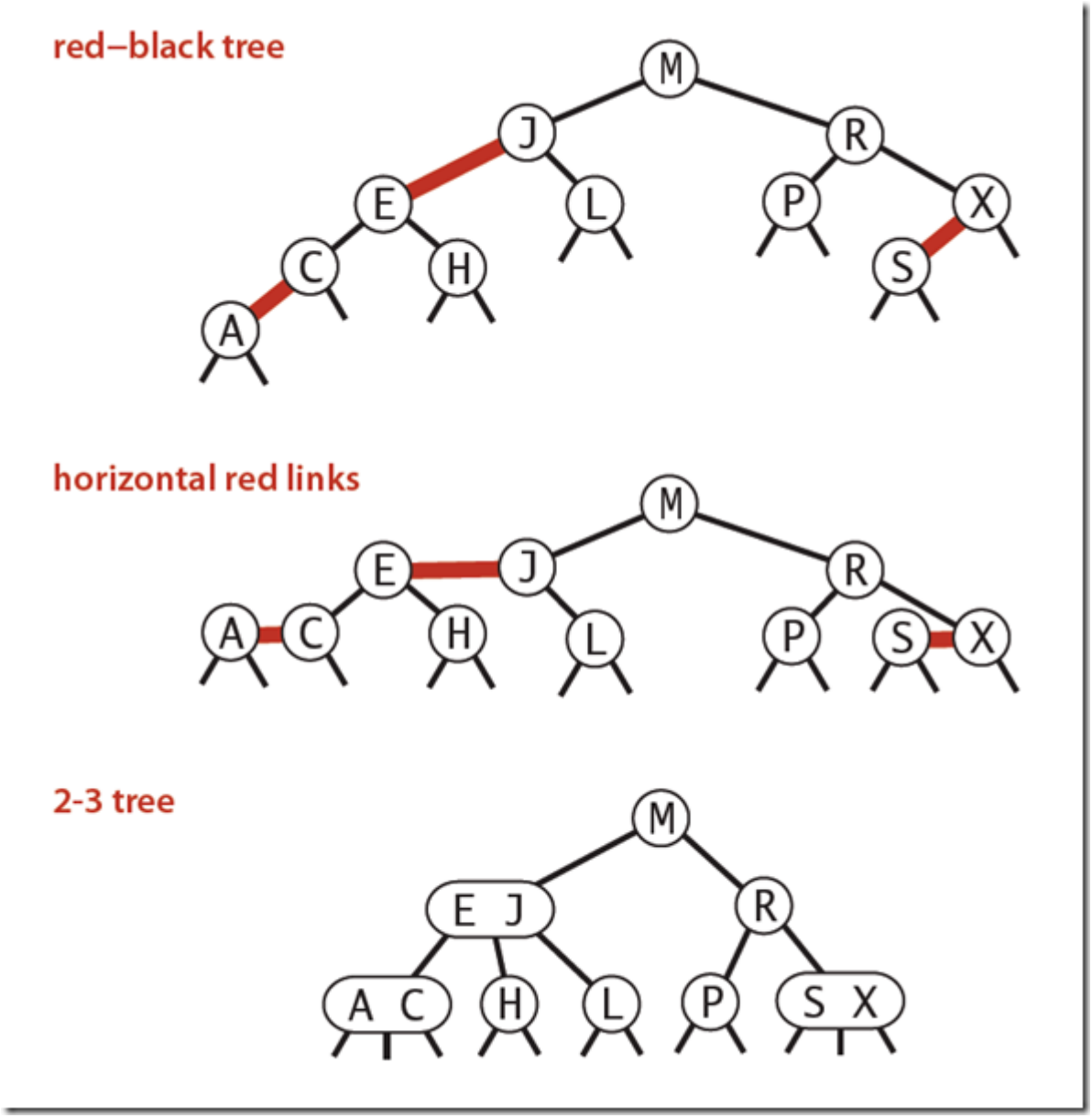
红黑树的思想就是对2-3查找树进行编码，尤其是对2-3查找树中的3-nodes节点添加额外的信息。红黑树中将节点之间的链接分为两种不同类型，红色链接，他用来链接两个2-nodes节点来表示一个3-nodes节点。黑色链接用来链接普通的2-3节点。特别的，使用红色链接的两个2-nodes来表示一个3-nodes节点，并且向左倾斜，即一个2-node是另一个2-node的左子节点。这种做法的好处是查找的时候不用做任何修改，和普通的二叉查找树相同。

【红黑树的定义】

红黑树是一种具有红色和黑色链接的平衡查找树，同时满足：

- 红色节点向左倾斜
- 一个节点不可能有两个红色链接
- 整个树完全黑色平衡，即从根节点到所以叶子结点的路径上，黑色链接的个数都相同。

下图可以看到红黑树其实是2-3树的另外一种表现形式：如果我们将红色的连线水平绘制，那么他链接的两个2-node节点就是2-3树中的一个3-node节点了。



【红黑树的性质】

整个树完全黑色平衡，即从根节点到所以叶子结点的路径上，黑色链接的个数都相同（2-3树的第2）性质，从根节点到叶子节点的距离都相等）。

【复杂度分析】

最坏的情况就是，红黑树中除了最左侧路径全部是由3-node节点组成，即红黑相间的路径长度是全黑路径长度的2倍。

B树（B Tree）

【基本思想】

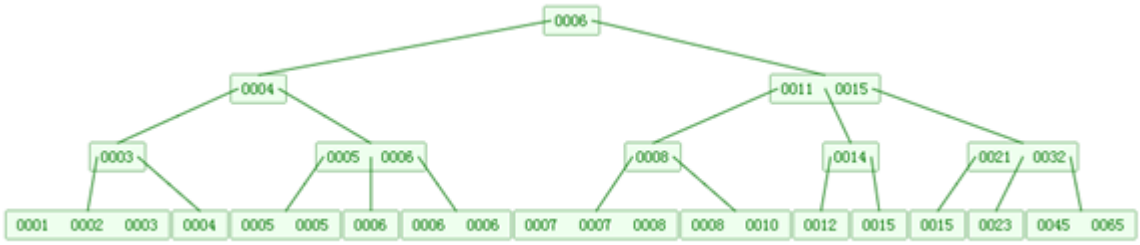
平衡查找树中的2-3树以及其实现红黑树。2-3树种，一个节点最多有2个key，而红黑树则使用染色的方式来标识这两个key。

维基百科对B树的定义为“在计算机科学中，B树（B-tree）是一种树状数据结构，它能够存储数据、对其进行排序并允许以O(log n)的时间复杂度运行进行查找、顺序读取、插入和删除的数据结构。B树，概括来说是一个节点可以拥有多于2个子节点的二叉查找树。与自平衡二叉查找树不同，B树为系统最优化**大块数据的读和写操作**。B-tree算法减少定位记录时所经历的中间过程，从而加快存取速度。普遍运用在**数据库**和**文件系统**。

【B树定义】

- 根节点至少有两个子节点
- 每个节点有M-1个key，并且以升序排列
- 位于M-1和M key的子节点的值位于M-1 和M key对应的Value之间
- 其它节点至少有M/2个子节点

下图是一个M=4 阶的B树:



可以看到B树是2-3树的一种扩展，他允许一个节点有多于2个的元素。B树的插入及平衡化操作和2-3树很相似，这里就不介绍了

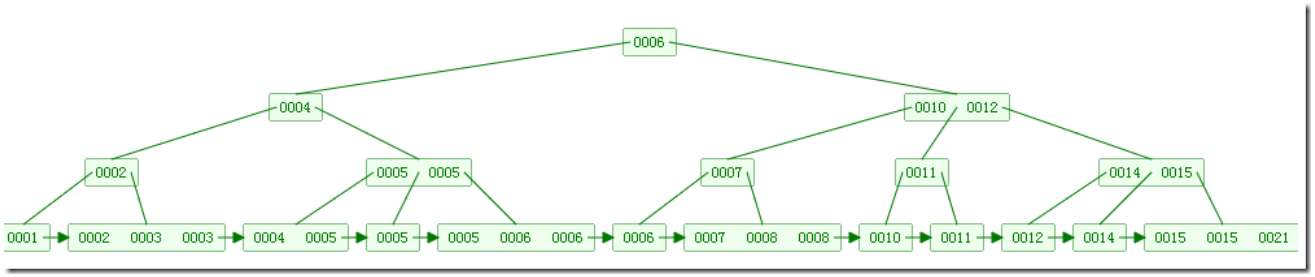
B+树

【B+树定义】

B+树是对B树的一种变形树，它与B树的差异在于：

- 有k个子结点的结点必然有k个关键码；
- 非叶结点仅具有索引作用，跟记录有关的信息均存放在叶结点中。
- 树的所有叶结点构成一个有序链表，可以按照关键码排序的次序遍历全部记录。

如下图，是一个B+树:



B和B+树的区别在于，B+树的非叶子结点只包含导航信息，不包含实际的值，所有的叶子结点和相连的节点使用链表相连，便于区间查找和遍历。

B+ 树的优点在于：

- 由于B+树在内部节点上不好含数据信息，因此在内存页中能够存放更多的key。 数据存放的更加紧密，具有更好的空间局部性。因此访问叶子节点上关联的数据也具有更好的缓存命中率。
- B+树的叶子结点都是相链的，因此对整棵树的便利只需要一次线性遍历叶子结点即可。而且由于数据顺序排列并且相连，所以便于区间查找和搜索。而B树则需要进行每一层的递归遍历。相邻的元素可能在内存中不相

邻，所以缓存命中性没有B+树好。

B树优点在于：

- 由于B树的每一个节点都包含key和value，因此经常访问的元素可能离根节点更近，因此访问也更迅速。

B/B+树常用于文件系统和数据库系统中，它通过对每个节点存储个数的扩展，使得对连续的数据能够进行较快的定位和访问，能够有效减少查找时间，提高存储的空间局部性从而减少**IO**操作。它广泛用于文件系统及数据库中