**[Trees and Heaps](http://www.fcicq.net/wp/?p=36" \o "Permanent link to Trees and Heaps)**

July 30, 2006 8:06 · Filed under [SEO](http://www.fcicq.net/wp/?cat=18), [技术文档](http://www.fcicq.net/wp/?cat=7)

二叉查找树 ( Binary Search tree )

定义：

一种二叉树。它可能是空的，若不是空的，它具有下列特性：

每一个元素有一键值，而且每一元素的键值都不相同，即每一个键值都是唯一的。

在非空的左子树上的键值，必小于在该子树的根节点中的键值。

在非空的右子树上的键值，必大于在该子树的根节点中的键值。

左子树和右子树也都是二叉查找树。  
  
建立二叉查找树：

以第一个建立之元素为根节点。

加入新元素的过程中，由根节点开始做比较，若元素值大于所比较之节点，则往右链结走，反之则往左链结走

搜索：亦如同建立树之原则，所欲找寻的元素比较大，则往右子树搜索，反之则往左子树搜索

删除节点：依搜索的方法找到目标节点后，如果是叶节点，则直接删除，而其父节点与它的链结改为NULL。 如果不是叶节点，则在其左子树找一最大值，或在其右子树找一最小值来取代所欲删除的节点的位置。

红黑树

红黑树是一种自平衡二叉查找树。它是在1972年由Rudolf Bayer发明的，他称之为”对称二叉B树”，它现代的名字是在 Leo J. Guibas 和 Robert Sedgewick于1978年写的一篇论文中获得的。它是复杂的，但它的操作有著良好的最坏情况运行时间，并且在实践中是高效的: 它可以在O(log n)时间内做查找，插入和删除，这里的n 是树中元素的数目。

定义：

红黑树是每个节点都有顏色特性的二叉查找树，顏色的值是红色或黑色之一。除了二叉查找树带有的一般要求，我们对任何有效的红黑树加以如下增补要求:

节点是红色或黑色。

根是黑色。

所有叶子（外部节点）都是黑色。

每个红色节点的两个子节点都是黑色。

从每个叶子到根的所有路径都包含相同数目的黑色节点。

哈夫曼编码(Huffman Coding)

定义：

又称最优二叉树，是一种带权路径长度最短的二叉树。带权路径长度最小的二叉树，经常应用于数据压缩。所谓树的带权路径长度，就是树中所有的叶结点的权值乘上其到根结点的路径长度（若根结点为0层，叶结点到根结点的路径长度为叶结点的层数）。树的带权路径长度记为WPL=(W1\*L1+W2\*L2+W3\*L3+…+Wn\*Ln)，N个权值Wi(i=1,2,…n)构成一棵有N个叶结点的二叉树，相应的叶结点的路径长度为Li(i=1,2,…n)。可以证明哈夫曼树的WPL是最小的。

从树中一个结点到另一个结点之间的构成这两个结点之间的路径，路径上的分支数目称做路径长度。树的路径长度是从树根到每一结点的路径长度之和。结点的带权路径长度为从该结点到树根之间的路径长度与结点上权的乘积。

将资料建构成霍夫曼树：

对资料中出现过的每一元素各自产生一外部节点，并予外部节点该元素之出现频率。

令 L 是所有外部节点所成之集合。产生一个新节点 N 。

令 N 为 L1 和 L2 的父节点，L1 和 L2 是 L 中出现频率最低的两个节点。

令 N 节点的出现频率等于 L1 和 L2 的出现频率总和。

由 L 中删除 L1 和 L2 ，并将 N 加入 L 中，重复步骤上一个动作，直到 | L | = 1 。

标示树中各节点的左子树链结为 0 ，右子树链结为 1 。

二叉树追踪 ( traversing a tree )

意义：

即遍历树中的每一个节点，且每个节点恰好被寻访一次。

中序遍历

前序遍历

后序遍历

：树根→左子树→右子树

：左子树→树根→右子树

：左子树→右子树→树根

M-Tree

Multi-Way Search Tree ( m-路搜索树 )

一个m路搜索树是一个所有节点之degree均小于或等于的树；可为空或满足下列特性：

(1)T之 其中为指向subtree之pointer，而则为key value.,.

﹝註﹞：最多有个subtree，最多有个key value .

(2), .

(3)之所有key value均小于，而大于，.

(4)中所有key value均大于，而中者则小于.

(5)都是way search tree .

B-Tree

定义

设一个B-Tree是一个m路搜索树，它可能是空的，或满足下列特性：

root :最少两个key value

除了root和 failure node，其他节点至少有个key value .

﹝注﹞：最多有个key value.

所有failure node都在同一阶层，亦即从树根到任一个树叶所经过的路径长度均相同

1.2-3树可能是空的树，或者满足下列特性

(1)每一内部节点可能是2-node或3-node。

(2)左边资料衡小于右边资料

(3)所有的外部节点都在同一阶层上

2.高度为 (即外部节点在阶层上) 的2-3树中，元素个数介于到。反之，有个元素的2-3树之高度介于和之间。

2-3 Tree

同B树一样，2-3-4 树是有序的: 每个元素必须大于或等于它左边的和它的左子树中的任何其他元素。每个儿子因此成为了由它的左和右元素界定的一个区间。2-3-4 树是红黑树的一种等同，这意味著它们是等价的资料结构。换句话说，对于每个 2-3-4 树，都存在著至少一个数据元素是相同次序的红黑树。在 2-3-4 树上的插入和删除操作也等价于在红黑树中的顏色翻转和旋转。这使得它成为理解红黑树背后的逻辑的重要工具

2-3-4 树

Balanced 2-3-4 tree (以下简称 234 tree) 是一种特殊的 4-ary search tree, 它的设计使它永远保持在 balanced 的状态, 如此可以确保所有搜索/新增/删除一笔资料所需的时间必为 O(lg n).

Heap

１、最大堆 ( max heap ) 为一种也是最大树的完整二叉树。

一个 Complete Binary Tree，每一个节点的内容不小于其子节点的内容。

２、最小堆 ( min heap ) 为一种也是最小树的完整二叉树。

一个 Complete Binary Tree，每一个节点的内容不大于其子节点的内容。

特性：

插入新节点、删除节点：时间复杂度为O(log N)

优先队列（Priority Queue）优先顺序高者先出来

最小-最大堆 ( min-max heap ) 是一个完整二叉树，若它不是空的，则每一元素有一个栏位，称为键值 (key) 。此树交替的阶层分别为最小阶层 ( min level ) 和最大阶层 ( max level ) ，根则位在最小阶层。

３、最小-最大堆 ( Min – Max Heaps )

定义

Deap 是一个完整二叉树，它是空的或满足下列特性：

根不包含元素。

左子树为最小堆(min-heap)。

右子树为最大堆(max-heap)。

如果右子树不是空的，则令 i 为左子树中任一节点。令 j 为右子树中相对的节点。如果这样的节点 j 不存在，则令 j 为右子树中与 i 的父节点相对节点。在节点 i 中的键值必须小于或等于节点 j 中的键值。

４、双向堆 ( DEAPS )

５、Fibonacci heap(简称F-heap)

F-heap跟一般heap不太一样，感觉像是广义的heap，他有特殊的资料结构来表示每个node，使得执行效能比一般heap快了许多，原本heap在insert，delete min，decrease key，所需花费的时间复杂度都是O(log n)。 F-heap的特殊资料结构，除了在delete min or delete上面，依旧需要O(log n)，其余的所需花的时间都是O(1)。若用F-heap来implement Dijkstra’s algoritm，一开始要对n个nodes初始化，要做n次的insert；接下来进到n次迴圈，有两部分动作要执行，第一部分在求目前距离s最近的点这部分，也就是做delete min，花费的时间是O(n log n)；另一部分是修改node的key值，也就是做decrease key，花费的时间是O(m)；所以总共所需的时间复杂度是O(m + n log n)。

高度平衡树 ( AVL Tree )

定义：

空的树为高度平衡树。如果 T 是非空二叉树，TL 和 TR 为其左、右子树，则 T 为高度平衡树，若且为若：

TL 和 TR 为高度平衡树。| hL – hR | <= 1，其中 hL 和 hR 分别为 TL 和 TR 的度。

平衡因子(balance factor)

节点的平衡因子是它的右子树的高度减去它的左子树的高度hL – hR。带有平衡因子 1、0 或 -1 的节点被认为是平衡的。带有平衡因子 -2 或 2 的节点被认为是不平衡的，并需要重新平衡这个树。平衡因子可以直接存储在每个节点中，或从可能存储在节点中的子树高度计算出来。

线索二叉树 (threaded binary tree)

线索二叉树(保留遍历时结点在任一序列的前驱和后继的信息)：若结点有左子树，则其lchild域指示其左孩子，否则令lchild域指示其前驱；若结点有右子树，则其rchild域指示其右孩子，否则令rchild指示其后继。还需在结点结构中增加两个标誌域LTag和RTag。LTag=0时，lchild域指示结点的左孩子，LTag=1时，lchild域指示结点的前驱；RTag=0时，rchild域指示结点的右孩子，RTag=1时，rchild域指示结点的后继。以这种结点结构构成的二叉链表作为二叉树的存储结构，叫做线索链表，其中指向结点前驱和后继的指针叫做线索，加上线索的二叉树称为线索二叉树。对二叉树以某种次序遍历使其变为线索二叉树的过程叫做线索化。若对二叉树进行中序遍歷，则所得的线索二叉树称为中序线索二叉树，线索链表称为为中序线索链表。

一个n节点的二叉树，共有2n个链结指标，但实际上只用了n-1个，而有n+1个空指标(NULL)。

引线二叉树(Thread binary tree)：充分利用这些空指标导引到其他的节点，以帮助快速搜索。

作法：把二叉树以中序追踪法将所有节点排列好，再将节点的空指标指向其左、右相邻节点即可