字典树的数据结构

字典树的核心思想

字典树的基本性质

**二叉搜索树 Binary Search Tree**

二叉查找树、二叉排序树、一颗空树

若左子树不为空， 则左子树上的所有的结点的值均小于它的根结点 的值；

若右子树不为空，则右子树上所有结点的值均大于它的根结点的值。

左右子树也分别为二叉排序树。

具有链表的快速插入与删除。

数组快速查找的优势。

文件系统、数据库系统一般会采用这种数据结构进行高效率的排序与检索操作。

常用链表数据结构表示：

一个节点就是一个对象： key、卫星数据、lchild左孩子、rchild右孩子、parent双亲（父结点）。 孩子结点或父结点不存在， 则相应属性的值为空null.

高效

1. 插入一个数值

2. 查询是否包含某个数值

3. 删除某个数值

复杂度 O（logn）

算法实现：

1. 查找： 递归查找是否存在Key

2. 插入： 原树中不存在key, 插入key返回true， 否则返回false。

3. 构造： 循环的插入操作。

4. 删除

1. 叶子节点： 直接删除，不影响原树。
2. 仅仅有左或右子树的节点：节点删除后，将它的左子树或右子树整个移动到删除节点的位置就可以， 子承父业。
3. 既有左子树又有右子树的节点：找到需要删除的节点P的直接前驱或者直接后继s,用s来替换节点P， 然后再删除节点s。

**字典树 Trie树**

*空间换时间*

*利用字符串的公共前缀来降低查询时间的开销以达到提高效率的目的。*

单词查找树或者键树、哈希树的变种

典型应用： 统计、排序和保存大量的字符串， 但不仅限于字符串。

案列： 搜索引擎系统用于文本词频统计。

优点：

利用字符串的公共前缀来减少查询时间，最大限度地减少无谓的字符串比较，查询效率比哈希树高。

性质：

1. 结点本身不存完整单词。
2. 从根结点到某一结点， 路径上经过的字符链接起来，为该结点对应的字符串。
3. 每个结点的所有子结点路径代表的字符都不相同。

操作：

查找

插入

删除-少见

方法

1. 从根节点开始一次搜索

2. 取得要查找关键字的第一个字母，并根据该字母选择对应的子树并转到该子树继续进行检索。

3. 在相应的子树上， 取得要查找关键字的第二个字母，并进一步选择对应的子树进行搜索

4. 迭代过程

5. 在某个节点处，关键词的所有字母已被取出， 则读取附在该结点的信息， 即完成查找。

应用

串的快速检索

"串"排序