原书第2版

2010年7月第1版第15次印刷

第三部分 数据结构

第11章 散列表

11.3 散列函数

11.3.1 除法散列法

可以选作的值常常是与的整数次幂不太接近的质数.

11.3.2 乘法散列法

11.3.3 全域散列

设为有限的一组散列函数，它将给定的关键字域映射到中.这样的一个函数组称为是全域的，如果对每一对不同的关键字，满足的散列函数的个数至多为.换言之，如果从中随机地选择一个散列函数，当关键字时，两者发生碰撞的概率不大于，这也正好是从集合中随机地独立选择和时发生碰撞的概率.

11.4 开放寻址法

第12章 二叉查找树

12.1 二叉查找树

设是二叉查找树中的一个结点.如果是的左子树的一个结点，则. 如果是的右子树的一个结点，则.

定理12.1 如果是一棵包含个结点的子树的根，则调用INORDER-TREE-WALK()过程的时间为.

证明

用表示在一棵包含个结点的子树的根，则调用INORDER-TREE-WALK()过程的时间，则有

设，那么

练习

12.1-2

In a heap, a node’s key is both of its children’s keys. In a binary search tree, a node’s key is its left child’s key, but its right child’s key.

The heap property, unlike the binary-search-tree property, doesn’t help print the nodes in sorted order because it doesn’t tell which subtree of a node contains the element to print before that node. In a heap, the largest element smaller than the node could be in either subree.

Note that it the heap property could be used to print the keys in sorted order in time, we would have an -time algorithm for sorting, because building the heap takes only time.

12.2 查询二叉查找树

定理12.2 对一棵高度为的二叉查找树，动态集合操作SEARCH, MINIMUM, MAXIMUM, SUCCESSOR和PREDECESSOR等的运行时间均为.

12.3 插入和删除

定理12.3 对于高度为的二叉查找树，动态集合操作INSERT和DELETE的运行时间为.

练习

12.3-2

Worst case: , occurs when a linear chain of nodes results from the repeated TREE-INSERT operations.

Best case: , occurs when a binary tree of height results from the repeated TREE-INSERT operations.

12.4 随机构造的二叉查找树

二叉查找树的各基本操作的运行时间都是.

思考题

12-4 不同二叉树的数目

设表示包含个结点的不同的二叉树的数目，那么，当时

设为生成函数，那么

泰勒展开式

于是

注意斯特林公式

第13章 红黑树

13.1 红黑树的性质

We shall regard these NIL's as being pointers to external nodes (leaves) of the binary search tree and the normal, key-bearing nodes as being internal nodes of the tree(内结点).

一棵二叉查找树如果满足下面的红黑性质，则为一棵红黑树：

1. 每个结点或是红的，或是黑的.
2. 根结点是黑的.
3. 每个叶结点(NIL)是黑的.
4. 如果一个结点是红的，则它的两个儿子都是黑的.
5. 对每个结点，从该结点到其子孙结点的所有路径上包含相同数目的黑结点.

采用一个哨兵代表NIL. 哨兵的域为BLACK，其它域可以设置成任意允许的值.所有指向NIL的指针都被替换成指向哨兵的指针.

采用哨兵代替所有的NIL：所有的叶子以及根部的父结点.

从某个结点出发(不包括该结点)到达一个叶结点的任意一条路径上，黑色结点的个数为该结点的黑高度，记为.红黑树的黑高度定义为根结点的黑高度.

引理13.1 一棵有个内结点的红黑树高度至多为.

证明

先证明以某一结点为根的子树至少包含个内结点.对的高度使用归纳法.若的高度为，那么必为，以为根的子树包含个内结点.若为一高度为正的内结点，并且有两个子女(注意也算).每个儿子的黑高度为或，根据归纳假设，每个儿子为根的子树至少包含个内结点，于是以为根的子树至少包含个内结点.

设树的高度为，根据性质4)，从根结点到叶结点的任意一条简单路径上，至少有一半结点是黑色的，根的黑高度至少为，于是

练习

13.1-5

Every path contains black nodes. The longest path from to a descent leaf has length at least . The longest path contains nodes and at least half the nodes on the path are black, so

13.1-6

根据引理13.1的证明过程，至少有个内结点.

最多的时候，各层红黑间隔，有个内结点.

13.2 旋转

旋转前，旋转后.

练习

13.2-2

若，则无法右旋.若，则无法左旋.可能的旋转数等于边数.

13.2-4

With at most right rotations, we can convert any binary search tree into one that is just a right-going chain.

Let us define the right spine as the root and all descendants of the root that are reachable by following only right pointers from the root. A binary search tree that is just a right-going chain has all nodes in the right spine.

As long as the tree is not just a right spine, repeatedly find some node on the right spine that has a non-leaf left child and then perform a right rotation on .

This rotation increases the number of nodes in the right spine by . Any binary search tree starts out at least one node-the root-in the right spine. At most, rotations are needed to put all nodes in the right spine, so that the tree consists of a single right-going chain.

We could perform this sequence in reverse-turning each right rotation into its inverse left rotation.

至多需要次旋转.

13.2-5

Right-going chain不能通过right rotation转成其它二叉查找树.

设的左子树比的左子树多个结点，先通过次right rotation使得两者左子树结点数目相同，设左右子树分别有个结点，那么

13.3 插入