DOI: 10.13657/j.cnki.gxkxyxb.1998.01.003

第 14卷第 1期 广 西 科 学 院 学 报 Vol. 14, No. 1
1998年 2月 Journal of Guangxi Academy of Sciences February 1998

基于映射和二分树的查找算法及实验研究*

A Dynamic Search Algorithm and Its Experiment Based on Mapping and Binary Trees

钟 诚
Zhong Cheng

(广西大学计算机科学系 南宁 530004)
(Dept. of Computer Sci., Guangxi Univ., Nanning, 530004)

摘要 通过构造散列表和二分树，设计 1个适用于大规模信息处理的快速动态查找算法，分析其执行效率，并给出程序及实验结果。

关键词 映射 二分树 查找算法

Abstract A dynamic searching algorithm applied in large information systems is designed by creating HASH table and some binary search trees, and its experiment is made on IBM Microcomputer.

Key words mapping, binary tree, searching algorithm
中图法分类号 TP 301.6

HASH方法是一种高效的查找技术，它广泛应用于文件系统、数据库系统和编译系统等领域。当不产生冲突时，其查找时间为常数 $O(1)$；而产生冲突时，可用线性探测、二次方法探测、随机探测和内（外）拉链方法解决。对于内（外）拉链方法，若具有某个相同 HASH地址的记录数目为 m，则在此冲突位置上查找时间为 $O(m)$。因为不管开辟的 HASH表有多大，也不管如何构造均匀 HASH函数，冲突的概率几乎不可能为零，尤其是对于大规模信息处理领域和某些关键字特殊的领域，动态查找过程中产生冲突的机会更大，所以当记录数目 N 很大时，m 值也较大，这时查找速度会明显下降[1,2]。为此，本文通过构造 HASH表和二分树，设计一个允许动态地插入、删除的快速查找算法。该算法在没有产生冲突的情形下，其查找时间为常数 $O(1)$；对于产生冲突情形，其平均查找时间为 $O(\log_2 m) = O(\log_2 N/B)$，明显优于内（外）拉链的动态 HASH方法的平均查找时间 $O(m) = O(N/B)$，这里的 N 为问题的规模即记录数目，B 为 HASH表的桶数目。本文给出的查找算法特别适用于大规模信息处理领域以实现快速查找。

1997-06-16收稿。

* 香港王宽诚教育基金会（9312004）和广西大学科研基金（S94314）资助项目。

1 数据结构与算法设计

采用的数据结构由一个具有 B 桶的基本 HASH表及若干个二分树组成，HASH表各项及二分树各结点用于存储记录信息，如图 1所示。

其中,二分树满足其任一棵左子树各结点记录的关键字均小于 (大于)根结点记录的关键字,且其任一棵右子树各结点记录的关键字均大于 (小于)根结点记录的关键字,即二分树应是一棵二分查找树。

HASH表各项和二分树各结点由记录数据和指针域两部分组成。

基于映射和二分树的动态查找过程如下:

(1) 由 HASH函数计算记录关键字 KEY相应的 HASH地址值 BUCKET;

(2)判断 HASH表 BUCKET位置是否为空,若空则将此记录存储于该位置上,转第 (5) 步;否则执行下一步;

(3) 将关键字 KEY与已存储在 HASH表中第 BUCKET桶位置上的记录的关键字作比较,若相等则查找成功,转第 (5) 步;否则执行下一步;

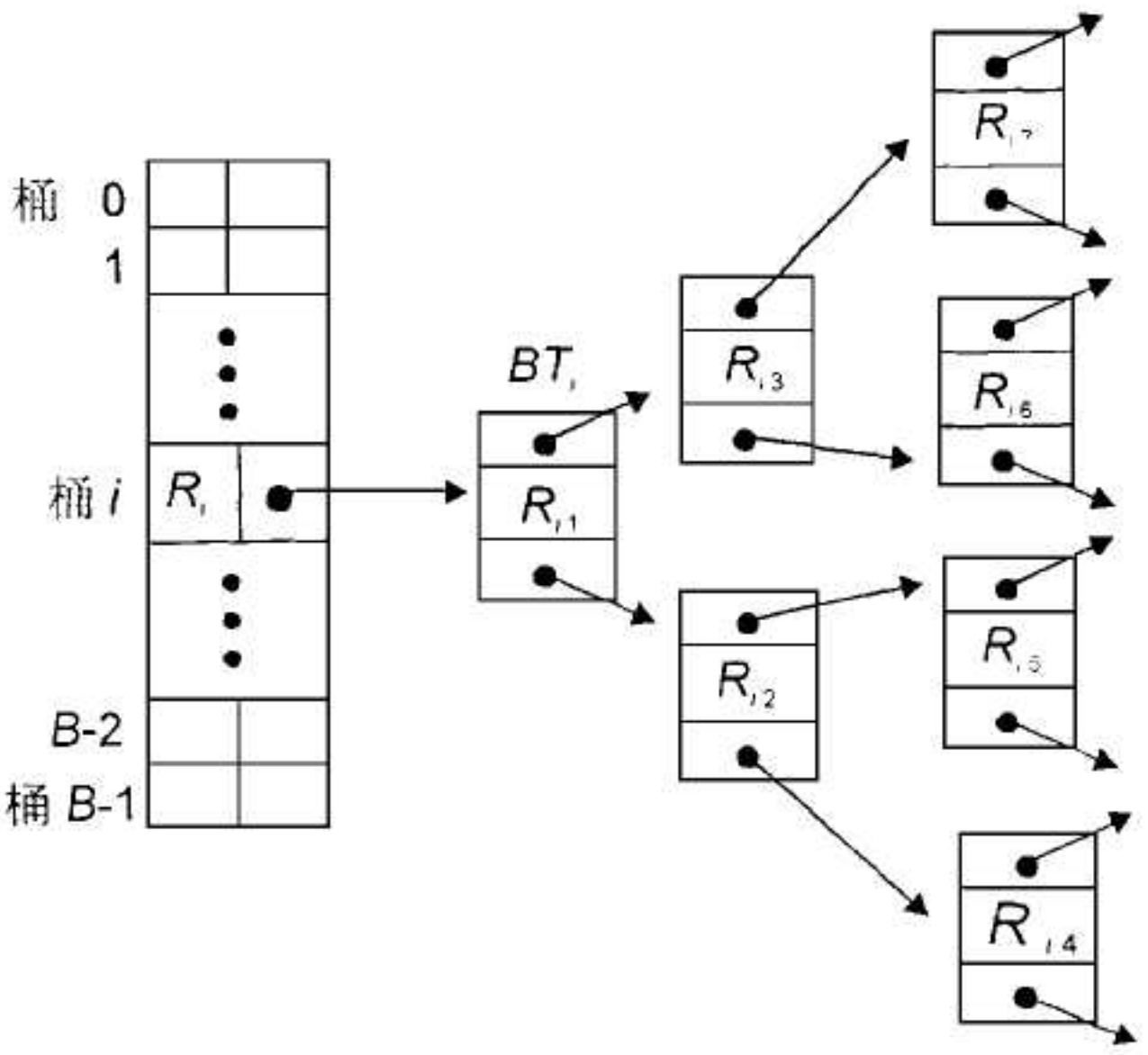

图 1 基于 HASH表和二分树的动态查找结构

BT_i为在桶 i位置产生冲突的二分树,R_i和 R_{ij}等为记录数据,B 为 HASH表桶数

(4) 读取 HASH表第 BUCKET桶位置中的指针域值，该指针指向二分树根结点，由此在二分树中进行动态查找，若查找成功，则转第 (5) 步；否则在二分树中插入此记录;

(5) 算法结束。

2 算法复杂性分析

设所开辟的 HASH表的桶数为 B,问题的规模即记录数为 N。

对于不产生冲突情形,本算法一次探测成功,查找时间为常数 $O(1)$

对于产生冲突情形，考虑如下两种可能:

(1) 记录关键字均匀分布

此时，记录被均匀地映射存储到 HASH表各桶位置，并且在各桶位置上产生冲突的可能性相同,在各桶位置产生冲突的记录平均数目 $m = (N - B)/B = N/B - 1$,而在此位置上生成引出的二分查找树的高度为 $\lceil \log_2 m \rceil + 1 = \lceil \log_2 (N/B - 1) \rceil + 1$,因此,其平均查找时间为 $O\lceil \log_2 m \rceil = O(\log_2 N/B)$;而对应的内 (外)拉链 HASH方法的查找时间为 $O(m) = O(N/B)$。

(2) 记录关键字非均匀分布

关键字却很少就属于非均匀分布情形。此时，HASH表可能有大量桶位置空着；而在某个桶位置却频繁产生冲突，最坏的情形为所有记录的关键字均被映射存储到同一个桶位置，这样在该位置上生成一棵高度为〔$\log_2 N$〕+ 1的二分查找树，其查找时间为 $O(\log_2 N)$，优于内(外)拉链的 HASH方法的查找时间 $O(N)$[1]。

3 实验结果讨论

为了验证本算法的正确性和有效性，我们采用 Turbo PASCAL 6.0分别编制本文设计的算法及外拉链 HASH方法相应的程序，基于 IBM PC386SX/33兼容微机，在 DOS系统支持下，通过由程序自动产生数据的方法，分别对这两种查找算法进行连续测试各组实验数据，运行结果如表 1所示。

表 1　两种动态查找算法运行时间比较

数据组号	冲突桶数	记录总数	冲突桶位置上记录数	外拉链 HASH查找算法	本文算法
1	5	5000	桶 1：4：1000桶 5：1000	6 s 70 ms	11 ms
2	5	8000	桶 1：4：1000桶 5：4000	26 s 94 ms	16 ms
3	5	10000	桶 1：4：1000桶 5：6000	52 s 79 ms	22 ms
4	5	12000	桶 1：4：1000桶 5：8000	89 s 59 ms	22 ms
5	1	20000	桶 1：20000	164 s 88 ms	92 s 72 ms

由实验结果可知，本文算法的运行速度比外拉链 HASH查找方法快得多。它适用于数据记录量较大、映射之后产生冲突的应用场合，尤其适用于记录数据存储于外存的场合。

参考文献

1　Knuth D E. The art of computer programming: sorting and searching, Addison-Wesley Publishing Company, 1973. 3.

2　周建钦等. 排序及查找理论与算法. 北京：科学出版社，1993.

3　钟　诚. 任意两序列公共元素的并行查找. 微电子学与计算机，1993，9.

4　钟　诚. 分级快速排序方法研究的研究. 计算机工程与应用，1997，4.