第 19 卷第 2 期
2003 年 5 月

广西科学院学报
Journal of Guangxi Academy of Sciences

Vol. 19, No. 2
May 2003

二叉树 CreateBTree 算法的改进
Improvement of Algorithm of CreateBTree for Generating Binary Trees

凌国贤
Ling Guoxian

(柳州高新技术开发区管理委员会财政局　柳州　545005)
(Finance Bureau of Management Committee, Liuzhou High-tech Developing District, Liuzhou, 545005)

摘要　在分析二叉树的 CreateBTree 算法的基础上，利用线性探测再散列方法对 CreateBTree 算法的中序遍历序列进行预处理来改进 CreateBTree 算法，使得改进后的 CreateBTree 算法在最差情况下，时间复杂度由 O (N^2) 降为 O(N)。

关键词　二叉树　CreateBTree 算法　线性探测再散列　时间复杂度
中图法分类号　TP301.6

Abstract　An algorithm of CreateBTree is analyzed. Its postorder traversal sequence is pretreated by using linear hashing. The time complexity of the improved algorithm goes down to O (N) from O (N^2).

Key words　binary tree, algorithm of CreateBTree, linear hashing, time complexity

1　CreateBTree 算法

CreateBTree 算法的功能是由已知的前、中序遍历序列递归生成二叉树，该算法功能性的执行步骤如下：

(1) 如果 $N<=0$，则令 T 为空指针（即令该二叉树为空二叉树），退出算法并返回成功标志。

(2) 如果 $N=1$，则生成一个结点，结点数据为前序遍历子序列的第一个数据 prestr[prestr _ startpos]，其左右子树设为空树，并令 T 指向该结点，然后退出算法并返回成功标志。

(3) 如果 $N>1$，则：

(a) 生成一个结点，结点数据为当前前序遍历子序列的第一个数据 prestr [prestr _ startpos]，并令 T 指向该节点；

(b) 在中序遍历子序列 midstr [midstr _ startpos. . midstr _ startpos+N−1] 中查找前序遍历子序列的第一个元素 prestr [prestr _ startpos]，所处的位置 pos（相对于整个中序遍历序列 midstr 的下标）。如果序列数据输入有误，则可能找不到 pos 使得 midstr [pos] =prestr

2003-03-26 收稿，2003-04-18 修回。

[prestr _ startpos] 成立，这种情况下令算法退出并返回失败标志；

(c) 递归调用算法 CreateBTree 生成 T 的左子树并返回标志。递归调用的参数：$T=T$ 的左子树指针，prestr _ startpos=prestr _ startpos+1，midstr _ startpos 不变，N=pos−midstr _ startpos，(即划分前序遍历子序列 prestr [prestr _ startpos+1..pos-midstr _ startpos+prestr _ startpos] 为根 T 的左子树的前序遍历序列，而划分中序遍历子序列 midstr [midstr _ startpos..pos−1] 为根 T 的左子树的中序遍历序列，然后由这 2 个划分出的序列递归生成 T 的左子树)

(d) 如果第 (c) 步递归调用算法后返回的是失败标志，则退出算法并返回失败标志，否则递归调用算法 CreateBTree 生成 T 的右子树并返回标志。该层递归算法结束。递归调用的参数：$T=T$ 的右子树指针，prestr _ startpos=prestr _ startpos+1+(pos-midstr _ startpos)，midstr _ startpos=pos+1，N=midstr _ startpos+N−1-pos。(即划分前序遍历子序列 prestr [prestr _ startpos+1+ (pos-midstr _ startpos) .. prestr _ startpos+N−1] 为根 T 的右子树的前序遍历序列，而划分中序遍历子序列 midstr [pos+1..midstr _ startpos+N−1] 为根 T 的右子树的中序遍历序列，然后由这 2 个划分出的序列递归生成 T 的右子树)

考虑最坏的情况，就是当所建立的二叉树是左单支树的时候，每次在中序遍历子序列中总要循环比较到最后一个元素，直到中序遍历子序列中只有一个元素为止，又因为该二叉树的结点个数为 N，需要查找 N 次。所以在最坏的情况下的时间复杂度为

$T(N)=\sum_{i=1}^{N} i=\frac{N(N+1)}{2}$，即为 $O(N^2)$。

2 对 CreateBTree 算法的改进

2.1 改进思路

由于该算法的前、中序遍历序列中的数据用字符表示，且串中不能出现相同的字符，因此可以利用线性探测再散列方法来定位某个字符在中序遍历序列中的位置，散列过程中不会发生冲突，将这 N 个字符的地址散列存放到一个地址数组中需 $O(N)$ 时间。每次在递归中用该散列函数以 O(1) 时间查找到要找的字符所在数组的下标，然后递归生成左右子树。由于每次进入函数生成一个结点，共有 N 个结点，也是 $O(N)$ 时间，所以总的时间复杂度为 $O(N)+O(N)=O(N)$ 时间。

2.2 改进方法

(1) 执行生成二叉树的递归算法 CreateBTree 前，首先对中序遍历序列进行预处理，即用线性地址再散列方法将中序遍历序列中的每个字符所处的下标散列存放在中序遍历序列地址数组 midaddr [1..200] 中。

(i) 设中序遍历序列为：Char midstr [1..N]；

(ii) 查找时用的中序遍历序列地址数组：Integer midaddr [1..200]；

(iii) 散列函数设计为 Hash (C) =C 的 ASCⅡ码-'A' 的 ASCⅡ码+1，其中 C 为单个字符。

(iv) 对中序遍历序列每个字符 midstr [i]，执行 midaddr [H (midstr [i])] =i，即是将 midstr [i] 原来的下标 i 记录在 midaddr 中。

(v) 中序遍历序列的预处理过程如下：

Integer H

```
For i=1 to N Do Begin
    H=Hash (midstr [i])    //即 H=midstr [i] 的 ASCⅡ码－'A' 的 ASCⅡ码＋1
    Midaddr [H] =i
Next
```

（2）在 CreateBTree 算法中，将在中序遍历子序列中查找前序遍历子序列的第一个元素所处的位置 i 的那些语句替换成 i =midaddr [H (midstr [i])]，参数 midstr 换成 midaddr 就可以改进上述算法了。

2.3　改进后的 CreateBTree 算法

2.3.1　*算法中的变量及调用说明*

T：Node——二叉树根结点指针；

Prestr：string——前序遍历序列；

Midstr：string——中序遍历序列；

Midaddr：string——中序遍历序列的地址数组；

prestr _ startpos：Integer——前序序列的起始位置；

Midstr _ startpos：Integer——中序序列的起始位置；

N：Integer——前（中）序遍历序列长度。

算法的每一个递归层次所用到的前（中）序遍历序列 prestr 和 midstr 以及中序遍历序列的地址数组 midaddr 都是同一个数组。算法最初调用时的参数 prestr _ startpos=1，midstr _ startpos=1，N=length (prestr)。

2.3.2　*算法 CreateBTree 的功能性执行步骤*

（1）如果 N<=0，则令 T 为空指针（即令该二叉树为空二叉树），退出算法并返回成功标志。

（2）如果 N=1，则生成一个结点，结点数据为前序遍历子序列的第一个数据 prestr[prestr _ startpos]，其左右子树设为空树，并令 T 指向该结点，然后退出算法并返回成功标志。

（3）如果 N>1，则：

（a）生成一个结点，结点数据为当前前序遍历子序列的第一个数据 prestr [prestr _ startpos]，并令 T 指向该节点；

（b）pos=midaddr (Hash (prestr [prestr _ startpos]))；

即是以 O (1) 时间得到前序遍历子序列的第一个元素 prestr [prestr _ startpos] 在中序遍历子序列 midstr [midstr _ startpos. . midstr _ startpos+N−1] 中所处的位置 pos（相对于整个中序遍历序列 midstr 的下标）。如果序列数据输入有误，则可能找不到 pos，使得 midstr [pos] =prestr [prestr _ startpos] 成立，这种情况下令算法退出并返回失败标志。

（c）递归调用算法 CreateBTree 生成 T 的左子树并返回标志。递归调用的参数：T=T 的左子树指针，prestr _ startpos=prestr _ startpos+1，midstr _ startpos 不变，N=pos-midstr _ startpos；（即划分前序遍历子序列 prestr [prestr _ startpos+1. . pos-midstr _ startpos+prestr _ startpos] 为根 T 的左子树的前序遍历序列，而划分中序遍历子序列 midstr [midstr _ startpos. . pos-1]为根 T 的左子树的中序遍历序列，然后由这 2 个划分出的序列递归生成 T 的左子树）

（下转第 76 页）

的。

对于已给的任意样本 X，只要知道其性别、综合成绩、毕业论文、党员及学生干部，就可预测其是否就业。

例 1 $X=\{971057,$"女","70～79","良","否","是"$\}$.

由图 1，可知其就业情况为"已"。

该预测模型在学生管理工作统计历届毕业生的近期就业情况，及对下届毕业生就业指导等方面有着现实的意义。

参考文献

1 Jiawei H, Kamber M. 数据挖掘概念与技术. 范 明，孟小峰译. 北京：机械工业出版社，2001. 185～220.
2 Matheus C J, Piatetsky-Shapiro G, McNeil D. Selecting and reporting what is interesting: The KEFIR application to healthcare data. In: Fayyak U M, Piatetsky-Shapiro G, Smyth P, Uthurusamy R. Advances in Knowledge Discovery and Data Mining, Cambridge, MA: AAAI/MIT Press, 1996. 495～516.
3 盛 骤，谢式千，潘承毅编. 概率论与数理统计. 北京：高等教育出版社，1999.

（责任编辑：黎贞崇）

（上接第 71 页）

(d)如果第(c)步递归调用算法后返回的是失败标志，则退出算法并返回失败标志，否则递归调用算法 CreateBTree 生成 T 的右子树并返回标志。该层递归算法结束。递归调用的参数：$T=T$ 的右子树指针，prestr _ startpos＝prestr _ startpos＋1＋(pos-midstr _ startpos)，midstr _ startpos＝pos＋1，N＝midstr _ startpos＋N－1－pos。（即划分前序遍历子序列 prestr[prestr _ startpos＋1＋(pos-midstr _ startpos). . prestr _ startpos＋N－1]为根 T 的右子树的前序遍历序列，而划分中序遍历子序列 midstr[pos＋1. . midstr _ startpos＋N－1]为根 T 的右子树的中序遍历序列，然后由这 2 个划分出的序列递归生成 T 的右子树）

3 结束语

本文改进算法只适用于结点数据为单个字符的二叉树，若希望生成任意数据类型(包括记录类型)结点的二叉树，则可以先对每个结点赋予一个惟一的 ID 号，由该 ID 号代替结点数据，然后选择一个好的散列函数对该 ID 号进行地址散列，也可在最差情况下达到 $O(N)$时间复杂度。

参考文献

1 Corman T H, Leisern C E. Introduction to algorithms. The MIT Press, 1995.
2 Pieprzyh J, Sadeghiyan. Design B of hashing algorithms. Berlin: Springer-verlag, 1993.
3 Baase A, Gelder A V. Computer algorithms: Introduction to design and analysis. Higher Education Press, 2001.
4 娄定俊. 算法分析与设计. 广州：中山大学出版社，1998.
5 严蔚敏，吴伟民. 数据结构(C 语言版). 北京：清华大学出版社，1997.
6 克努特 D E. 计算机程序设计技巧(第一卷 基本算法). 管纪文，苏运霖译. 北京：国防工业出版社，1980. 277.

（责任编辑：黎贞崇）