证明 必要性显然.

设$\mathscr{B}$是X的一个θ-基,$\mathscr{A}$是X的一个θ-开覆盖.对任意$A\in\mathscr{A}$,因$A\in\theta O(X)$,存在$\mathscr{B}_A\subset\mathscr{B}$,使$A=\bigcup_{A\in\mathscr{B}_A}B$.令$\mathscr{B}_1=\bigcup_{A\in\mathscr{A}}\mathscr{B}_A$,则$\mathscr{B}_1$是$\mathscr{B}$的子族,且,

$$\bigcup_{B\in\mathscr{B}_1}=\bigcup_{B\in\bigcup_{A\in\mathscr{A}}\mathscr{B}_A}B=\bigcup_{A\in\mathscr{A}}(\bigcup_{B\in\mathscr{B}_A}B)=\bigcup_{A\in\mathscr{A}}A=X,$$

即$\mathscr{B}_1$($\mathscr{B}_1\subset\mathscr{B}$)也是$X$的一个$\theta$-开覆盖,从而$\mathscr{B}_1$有可数子覆盖$\mathscr{B}_1^1$.如果$B\in\mathscr{B}_1^1$,则存在$A\in\mathscr{A}$,$B\in\mathscr{B}_A$,因此$B\subset A$,于是对于每一个$B\in\mathscr{B}_1^1$.可以选定某一个$A_B\in\mathscr{A}$,使得$B\subset A_B$,记$\mathscr{A}_1=\{A_B\mid B\in\mathscr{B}_1^1\}$.它是$A$的一个子族,并且

$$\bigcup_{A\in\mathscr{A}_1}A=\bigcup_{B\in\mathscr{B}_1^1}A_B\supset\bigcup_{B\in\mathscr{B}_1^1}B=X,$$

所以$\mathscr{A}_1$是$\mathscr{A}$的一个θ-子覆盖.由于$\mathscr{B}_1^1$是可数的,所以$\mathscr{A}_1$也是可数的.于是θ-覆盖$\mathscr{A}$有一可数子覆盖$\mathscr{A}_1$,这证明X是一个θ-Lindelöf空间.

定理2.7 若X是Lindelöf空间,则X是θ-Lindelöf空间.

由定理1.2直接可得.

定义2.9 若$(X,\mathscr{T})$是一个拓扑空间.如果X的每一个θ-开覆盖有一个有限子覆盖,则称拓扑空间X是一个θ-紧空间.

定理2.8 每一个θ-紧空间都是θ-Lindelöf空间.

定理2.9 X是一个θ-紧空间当且仅当存在拓扑空间X的一个θ-基$\mathscr{B}$,使得X的由$\mathscr{B}$中的元素构成的每一个θ-覆盖有一个有限子覆盖.

证明 由于$\theta O(X)$本身就是一θ-基,故必要性显然.

设$\mathscr{A}$是X的一个θ-开覆盖.对于每一个$A\in\mathscr{A}$,存在$\mathscr{B}$的一个子族$\mathscr{B}_A$,使得$A=\bigcup_{B\in\mathscr{B}_A}B$.令$\mathscr{A}_1=\bigcup_{A\in\mathscr{A}}\mathscr{B}_A$,由于

$$\bigcup_{B\in\mathscr{A}_1}B=\bigcup_{B\in\bigcup_{A\in\mathscr{A}}\mathscr{B}_A}B=\bigcup_{A\in\mathscr{A}}(\bigcup_{B\in\mathscr{B}_A}B)=\bigcup_{A\in\mathscr{A}}A=X,$$

故$\mathscr{A}_1$是一个由$\mathscr{B}$的元素构成的X的一个θ-开覆盖,从而有一个有限子覆盖,设为$B_1,B_2,\cdots,B_n$,对于每一个$B_i,i=1,2,\cdots,n$,由于$B_i\in\mathscr{A}_1$,所以存在$A_i\in\mathscr{A}$,使得$B_i\in\mathscr{B}_{A_i}$,因此$B_i\subset A_i$.于是对于$\mathscr{A}$的有限子族$\{A_1,A_2,\cdots,A_n\}$,有$A_1\cup A_2\cup\cdots\cup A_n\supset B_1\cup B_2\cup\cdots\cup B_n=X$.也就是说$\mathscr{A}$有一个有限子覆盖$\{A_1,A_2,\cdots,A_n\}$,这证明$X$是一个$\theta$-紧空间.

参考文献

[1] 汪贤华,高汝林.θ-连通空间及其性质[J].北京石油化工学院学报,2005,13(1):61-64.

[2] 钟治初.关于θ-可数紧和θ-子集紧空间[J].淮北煤师院学报,1995,16(3):10-13.

[3] 张广济.θ-紧空间的性质[J].辽宁师范大学学报:自然科学版,1993(2):111-115.

[4] Levine N. scmi-open scts and scmi-continuity in topological spaccs[J]. Amer MathMonthly,1963,70:36-41.

[5] 熊金城.点集拓扑讲义[M].3版.北京:高等教育出版社,2003.

编辑:文心

用栈无标记变量后序遍历二叉树算法

柴宝杰,马弘伟
(牡丹江师范学院,黑龙江 牡丹江 157012)

摘 要:给出一种用栈无标记变量后序遍历二叉树算法,并与常见的用栈加标记变量后序遍历二叉树算法就额外空间和额外栈深等进行分析比较.分析结果显示,无标记变量后序遍历二叉树算法可以节省空间,降低复杂性.

关键词:二叉树;标记变量;后序遍历

[中图分类法]O151 [文献标识码]A [文章编号]1003-6180 (2008) 03-0018-02

遍历二叉树的前序、中序和后序非递归算法中,后序遍历最为复杂.后序遍历二叉树的非递归算法有用栈、逆转链、Robson、Siklóssy等,而对于递归定义的二叉树来说,用栈实现其非递归算法是最自然的.本文将对用栈的后序遍历二叉树算法进行讨论,后面凡用到"遍历算法"字样,均指用栈非递归遍历二叉树算法.

后序遍历二叉树的过程可概括地描述为:a.

收稿日期:2008-03-12

后序遍历根结点的左子树;b. 从左子树返回到根结点;c. 后序遍历根结点的右子树;d. 从右子树返回到根结点;e. 访问根结点.

后序遍历算法的复杂之处在于如何判断是从左子树还是从右子树返回到根结点. 一般的做法是为每一个结点设置一个标记变量,该标记变量可以取两个不同的值,分别表示正在遍历的是该结点的左子树和右子树,在返回时就可以此判断是从哪个子树返回的[1]. 文中将使用这种思想设计的算法称为"一般算法".

研究发现,可以实现不用标记变量的后序遍历二叉树算法. 基本思想是在进入右子树前,先推一个空指针入栈,在退栈时就可以此判断是从左子树或右子树返回的.

1 无标记变量的遍历算法

算法 用栈无标记变量后序遍历二叉树

```
Status PostOrderTraverse2(BiTree T, Status( * Visit)(TElemType e))
{
if(T == NULL)
{
return OK;
}
p = T;
InitStack(S) ;
while(TRUE)
{
while(p)
{
Push(S, p);//向左走到尽头
p = p -> lchild;
}
GetTop(S, p);
while(!p) //从右子树返回,子树根结点退栈并访问之
{
Pop(S,p);//空指针退栈
Pop(S,p);//子树根结点退栈
if(!Visit(p->data))
{
return ERROR;
}
if(StackEmpty(S))
{
return OK;
}
GetTop(S, p);
}
Push(S, NULL);
p = p -> rchild;//向右一步
}//while(TRUE)
return OK;
}// PostOrderTraverse2
```

算法的二叉树结点类型定义如下:

```
typedef struct BiTree{
TElemType data;
struct BiTNode * lchild, * rchild;
}BiTNode, * BiTree;
```

在进入右子树前先推一个空指针入栈. 由于只有进入右子树时才会有空指针入栈,所以返回时就可根据有无空指针判断是从哪个子树返回的,进而做进一步的处理. 因为叶子结点是需要立即访问的结点,而需要访问的结点应该是从右子树返回时栈顶的结点,为了便于循环操作,当叶子结点入栈后推一个空指针入栈.

2 无标记变量的遍历算法与一般算法的比较

主要从两种算法的额外空间以及额外栈深两方面比较.

设二叉树结点的个数为 n,深度为 h,int 类型变量占 2 字节空间.

2.1 额外空间比较

一般算法为每个结点加一个 int 形成员变量,共占用 2 * n 个字节的额外空间,无标记变量的遍历算法没有占用额外空间.

2.2 额外栈深比较

一般算法没有额外增加栈的深度,无标记变量的遍历算法使栈的深度有所增加,最坏的情况是增加 h.

设无标记变量的遍历算法使栈的深度增加值为 a,下面两种边缘情况显然成立:空二叉树 a=h=0;只有一个根结点,同时也是叶子结点,所以 a=h=1.

假设存在某二叉树 T,a>h. 记 T 的左右子树的额外栈深分别为 al 和 ar,左右子树的深度分别为 hl 和 hr. 由于进入左子树前没有推空子针入栈,而进入右子树前推一个空指针入栈,又因为左子树遍历完才去遍历右子树,这时左子树的结点和空指针已经从栈中弹出,所以可得出 a = max(al, ar + 1),其中 max(x, y)表示 x 和 y 中较大者. 因为 h = max(hl, hr)+1,a>h,所以 al>hl 或 ar>hr. 对额外栈深大于二叉树深的那棵子二叉树作同样推理,这样下去必然得出这样的结论:叶子结点的额外栈深大于 1 或空二叉树的额外栈深大于 0. 与已知情况矛盾.

两种极端情况:当只有右子树时,a=n=h;

当只有左子树时,a=1,即只有惟一的叶子结点需要推空指针入栈.

两种算法的循环结构相同,循环次数相同,只有个别语句不同,所以时间效率相同.相对来讲,无标记变量的遍历算法没有标记变量,这与前序和中序遍历算法一致,从而减少了算法的复杂性.

3 结束语

无标记变量后序遍历算法可以节省空间,降低复杂性,虽然也会使栈的深度有所增加,但一般情况下,二叉树的深度远远小于结点个数,而在最坏的情况,额外栈深只有二叉树的深度.

参考文献

[1] 许卓群,张乃孝,杨冬青,等.数据结构[M].北京:高等教育出版社,1987.
[2] 严蔚敏,吴伟民.数据结构[M]:C语言版.北京:清华大学出版社,1997.
[3] 张选平,雷咏梅.数据结构[M].北京:机械工业出版社,2002.
[4] (美) Robert L. Kruse, Alex Ryba. Data Structures and Program Design in C++[M]. American:Prentice Hall,2004.

编辑:李志敏

在线考试系统的设计

符维强
(牡丹江联通公司,黑龙江 牡丹江 157000)

[中图分类法]TP4 [文献标识码]A [文章编号]1003-6180(2008)03-0020-02

针对传统考试方式存在的诸多不足,研究开发了一种基于MySQL数据库的在线考试系统.

1 在线考试系统的相关技术

PHP PHP是一种易于学习和使用的服务器端脚本语言,只需要很少的编程知识就能使用PHP建立一个真正交互的WEB站点.PHP是完全免费的,可以从PHP官方站点(http://www.php.net)自由下载.PHP可以编译成具有与许多数据库相连接的函数.PHP与MySQL是现在绝佳的组合.利用PHP语言,可根据用户的要求在web服务器上动态地建立新的主页.

Apache Apache是世界排名第一的Web服务器,它能为网络管理员提供丰富多彩的功能,包括目录索引、目录别名、内容协商、可配置的HTTP错误报告、CGI程序的SetUID执行、子进程资源管理、服务器端图像映射、重写URL、URL拼写检查以及联机手册man等,是创建Web服务器的最佳选择.

MySQL MySQL是一个快速、高效、多线程、多用户的关系型数据库系统,它可以在多种平台上工作,并能与PHP非常完美地结合.MySQL采用标准的结构化查询语句(SQL)对数据库中的数据进行操作,使得存储、更新或者检索数据变得更加容易.

2 数据库系统设计

数据库名称:test

数据库中所涉及的表:user表、exam表、student_exam表.

数据库中表的创建:create database test.

```
Create table user (
ID int(11) auto_increment primary key,
Private_ID int(23),
Name varchar(30),
User_name varchar(30),
Pass_word varchar(20),
Sex varchar(5),
Birth date,
Birth_place text,
Departmet text,
Grade varchar(10),
Major varchar(30),
Tel varchar(13),
Email varchar(20),
User_level int,
Ice_out int
);
create table exam2 (
ID int(5) primary key auto_increment,
```

收稿日期:2007-12-06