新型二叉树后序遍历非递归算法*

孙 毅[1,2], 张 丽[3]

(1. 东南大学计算机科学与工程学院, 江苏 南京 210096;
2. 金陵科技学院公共基础课教学部, 江苏 南京 211169; 3. 73841部队自动化站, 江苏 南京 210003)

摘 要: 二叉树遍历的非递归算法相对于递归算法, 减少了函数调用等开销, 具有性能优势。传统的二叉树后序遍历非递归算法, 用栈来模拟递归调用的全过程, 没有算法意义上的改进。由于先序遍历相对后序遍历具有较大的性能优势, 基于两种遍历的逆序关系, 将后序遍历转化为先序遍历, 提出了新型的后序遍历非递归算法。通过理论证明及试验数据的对比分析, 验证了新型算法的高效性。

关键词: 二叉树; 后序遍历; 非递归算法; 栈
中图分类号: TP 311. 12 文献标识码: A 文章编号: 1672-755X(2008)01-0026-04

New Non-recursive Algorithm of Post-traversing Binary Tree

SUN Yi[1,2], ZHANG Li[3]

(1. Southeast University, Nanjing 210096, China; 2. Jinling Institute of Technology, Nanjing 211169, China;
3. Automation Office of 73841 Troop PLA, Nanjing 210003, China)

Abstract: Compared with recursive algorithm for binary tree traversing, non-recursive algorithm reduces expenses of function calls, gains performance advantage. Adopting more-push method, traditional non-recursive post-traversing algorithm simulates the recursive procedure by stack, but it doesn't improve algorithm from essence. Since pre-traversing is efficient than post-traversing, this paper aims the relationship of these tow method, transforms post-traversing to pre-traversing and builds up new non-recursive post-traversing algorithm. Through theoretical proof and comparative analysis on experimental data, efficiency is verified.

Key words: binary tree; post-traversing; non-recursive algorithm; stack

二叉树作为树形数据结构的重要类型, 具有简单的存储结构和相关算法, 在程序设计及应用中使用非常广泛。遍历是二叉树的一种重要运算, 很多其他运算都基于遍历, 可见遍历算法的优劣对二叉树的性能有重要意义。递归算法具备较好的可读性、设计容易等优点, 但由于递归算法引入了多次函数调用的开销, 在性能上与非递归算法有差距。尤其对于程序中频繁执行的部分, 设计非递归算法至关重要。

经典的数据结构教程[1-2]中利用栈模拟函数递归调用, 构造了遍历的非递归算法。其中后序遍历的非递归算法[3-4]复杂度较高, 对于结点多次压栈, 不利于理解, 性能有待于进一步提高。文献[5-6]中增加标记来标识出栈情况, 文献[7]采用了两个栈, 总体上都是对直接用栈模拟递归的微小调整, 没有真正算法意义上的改进。本文从先序遍历和后序遍历的关系入手, 证明了先序遍历与后序遍历的逆序关系, 以先序遍历非递归算法为基础, 提

* 收稿日期: 2007-10-12; 修回日期: 2008-01-15
作者简介: 孙 毅(1978-), 男, 吉林省吉林市人, 硕士研究生, 助教, 研究方向: 网络行为学。

出了新型的二叉树后序遍历非递归算法, 从理论和实验两方面验证了算法的高效性。

1　新型算法及实现

现有非递归遍历算法均为用栈模拟整个递归过程, 从先序遍历与后序遍历递归算法的性能对比可以发现: 后序遍历的栈操作远远多于先序遍历(详细内容见第 2 部分)。如果能将后序遍历转换为先序遍历, 构造出的非递归算法将有突破性性能提升。

传统遍历方式有先序、中序、后序 3 种, 这里是默认先左后右的顺序, 引入以下符号: M- 根结点, L- 左子女, R- 右子女, 则上边 3 种遍历分别为先序(MLR)、中序(LMR) 和后序(LRM)。以图 1 所示二叉树为例, 3 种遍历结果分别为: ABDCE、BDAEC 和 DBECA。为了寻找先序遍历与后序遍历的关系, 不妨引入另外 3 种遍历: 先序(MRL)、中序(RML)、后序(RLM)。其遍历结果为: ACEBD、CEADB 和 ECDBA。表 1 将这种对应关系清楚地展示出来。

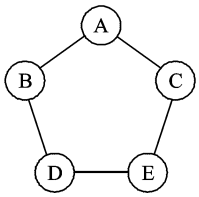

图 1　二叉树

Fig. 1　Binary-tree

表 1　图 1 所示二叉树 6 种遍历结果对比

Table 1　Comparison of six traversing results from binary-tree in figure 1

遍历类型	遍历结果
先序 MLR	ABDCE
后序 RLM	ECDBA
中序 LMR	BDAEC
中序 RML	CEADB
后序 LRM	DBECA
先序 MRL	ACEBD

由表 1 可见, 6 种遍历的结果分成 3 类, 每类中两种遍历结果排列顺序是互逆的, 即: 先序(MLR) 与后序(RLM)、中序(LMR) 与中序(RML)、后序(LRM) 与先序(MRL) 分别互逆。定理 1 给出了如上事实的完整证明。

定理 1　二叉树 6 种遍历中, 先序(MLR) 与后序(RLM)、中序(LMR) 与中序(RML)、后序(LRM) 与先序(MRL) 3 组的遍历结果排列顺序分别互逆。

证: 不妨只证明后序(LRM) 与先序(MRL) 的遍历结果互逆。

任给一二叉树, 不妨结点数 $n \geqslant 2$, 任取两个结点 $node_i$, $node_j$, 设两结点后序(LRM) 遍历结果的序号为 $post_i$, $post_j$; 先序(MRL) 遍历结果的序号为 pre_i, pre_j。

由于 $n \geqslant 2$, 故 $node_i$, $node_j$ 有最近公共祖先, 记为 a, 共两种情况:

1) $a = node_i$ 或 $a = node_j$;

2) $a \neq node_i$ 且 $a \neq node_j$。

下证在两种情况下均有 $post_i$, $post_j$ 与 pre_i, pre_j 逆序。

1) 不妨 $a = node_i$, 则 $node_j$ 位于左子树或右子树, 均有 $pre_i < pre_j$ 且 $post_i > post_j$;

2) a 是最近公共祖先, 故 $node_i$, $node_j$ 分别位于 a 的左右子树, 不妨 $node_i$ 在左子树, $node_j$ 在右子树, 则 $pre_i < pre_j$ 且 $post_i > post_j$。

故原命题成立。

基于定理 1, 可以建立起新型的二叉树后序(LRM) 遍历算法, 这里用一个数组结构 LRM [ ] 来存储后序(LRM) 遍历的结果。非形式化的算法如下:

```
void post- LRM (bitree root, int & LRM [ ]){
  push(root);
  while(stack is not empty){
    pop(p);
    while(p! = NULL){
    将 p- > data 以逆序形式写入数组
    LRM [ ];
    push(p- > lchild);   //先序(MRL),
    所以左子女入栈
    p= p- > rchild;
    }
  }
}
```

2　算法性能对比分析

下面从理论分析和实验数据两方面对新型算法和传统算法的性能进行对比。由于新型算法的性能耗费本质上与先序(MLR) 遍历的耗费相同,

这里直接对先序(MLR)遍历与后序(LRM)遍历非递归算法的效率进行对比。

2.1 理论对比分析

首先了解一下传统的后序遍历非递归算法,便于对出入栈情况有清楚的了解。算法描述:最后访问根结点,所以对任一结点,先沿左分枝向下搜索,搜到结点则进栈,直到左分枝为空;然后退栈,取出最后入栈的结点 x,此时不访问,而是从该结点的右分枝的根开始,同样方法沿其左分枝处理;处理完结点 x 的右分枝才能访问结点 x。对任一结点都有两次出现在栈顶(不包括刚进栈):第一次在处理完左分枝时,第二次在处理完右分枝时。第一次出栈不访问,目的在于找到右分枝,只有第二次出栈才访问它。为了区别第几次出栈,为结点设置标志位 flag,随结点进出栈,进而完成后序遍历过程。表 2 为图 1 所示二叉树传统后序遍历非递归算法的栈使用情况。根据标志位 flag 取 1、2,指针分别指向左右子女结点。

由于先序遍历的非递归算法访问过根结点后,只需将右子女压栈,然后按同样策略继续访问左子女,依次便可以完成先序遍历的全过程;而后序遍历非递归算法为了访问右子女和根结点,所有的结点都必须进行两次入栈出栈操作。这样显著增加了对栈的访问消耗。表 3 是图 1 二叉树先序(MLR)

表 **2** 图 **1** 所示二叉树后序遍历传统非递归算法栈变化情况

Table 2 Stack variation of traditional non-recursive post-traversing of binary-tree in figure 1

序号	栈操作	栈内元素	当前指针	访问元素	说明
1	初始化	#	A		调用开始
2	A 入栈 1	A	B		指向 A 的左结点 B
3	B 入栈 1	AB	NULL		指向 B 的左结点 NULL
4	B 出栈 1	A	B		
5	B 入栈 2	AB	D		指向 B 的右结点 D
6	D 入栈 1	ABD	NULL		指向 D 的左结点 NULL
7	D 出栈 1	AB	D		
8	D 入栈 2	ABD	NULL		指向 D 的右结点 NULL
9	D 出栈 2	AB	NULL	D	
10	B 出栈 2	A	NULL	B	
11	A 出栈 1	#	A		
12	A 入栈 2	A	C		指向 A 的右结点 C
13	C 入栈 1	AC	E		指向 C 的左结点 E
14	E 入栈 1	ACE	NULL		指向 E 的左结点 NULL
15	E 出栈 1	AC	E		
16	E 入栈 2	ACE	NULL		指向 E 的右结点 NULL
17	E 出栈 2	AC	NULL	E	
18	C 出栈 1	A	C		
19	C 入栈 2	AC	NULL		赘向 C 的右结点 NULL
20	C 出栈 2	A	NULL	C	
21	A 出栈 2	#	NULL	A	调用结束

表 **3** 图 **1** 所示二叉树先序遍历非递归算法栈变化情况

Table 3 Stack variation of non-recursive pre-traversing of binary-tree in figure 1

序号	栈操作	栈内元素	当前指针	访问元素	说明
1	初始化	#			调用开始
2	A 入栈 A 出栈	#	A		预入栈
3	C 进栈	C	B	A	指向 A 的左结点 B
4	D 进栈	CD	NULL	B	指向 B 的左结点 NULL
5	D 出栈	C	NULL	D	指向 D 的左结点 NULL
6	C 出栈	#	E	C	指向 C 的左结点 E
7		#	NULL	E	指向 E 的左结点 NULL 结束

遍历非递归算法栈使用情况。通过对比可见: 先序(MLR)遍历仅仅对右结点压栈, 加上预入栈共有 3 次进栈 3 次出栈; 后序遍历对每一个结点都有两次出入栈, 共计 10 次进栈 10 次出栈。所以先序遍历在栈操作的消耗上, 远小于后序遍历。这从理论上证明了新型算法的高效性。

2.2　实验数据对比

实验中, 以整型结点为例, 通过不断扩大结点个数, 对新型算法与传统算法的时间消耗进行对比, 由于时间消耗被多因素影响而变动, 对于每个结点数 N, 以 10 次实验的均值作为表 4 的实验结果数据。图 2 指明新型算法比较传统算法效率相对提高 40% 左右。

3　结　语

本文基于先序遍历与后序遍历的逆序关系, 构造出新型的后序遍历非递归算法, 通过理论分析及实验数据两方面证明了算法的高效性, 为二叉树后序遍历算法的理解及性能提高提供了新思路。

表 **4**　算法性能对比

Table 4　Comparison of algorithms in performance

结点数 N/1	新型算法耗时/μs	传统算法耗时/μs
10^2	3	5
10^3	24	44
10^4	227	381
10^5	2 303	4 170
10^6	24 004	38 087
10^7	240 630	393 298

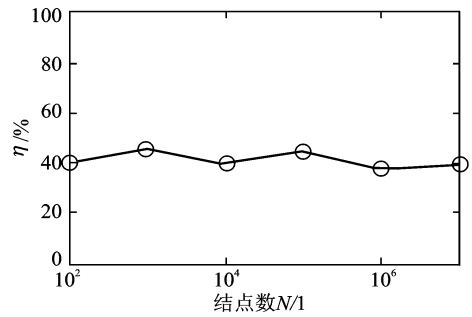

图 **2**　新型算法效率相对提高率

Fig. 2　Increment in performance of new algorithm

参考文献:

[1] 严蔚敏, 吴伟民. 数据结构[M]. 北京: 清华大学出版社, 1997: 128- 131

[2] 殷人昆, 陶永雷. 数据结构(用面向对象方法与 C++ 描述)[M]. 北京: 清华大学出版社, 1999: 172- 182

[3] 孟林, 李忠. 递归算法的非递归化研究[J]. 计算机科学, 2001, 28(08): 96- 98

[4] 朱振元, 朱承. 递归算法的非递归化实现[J]. 小型微型计算机系统, 2003, 24(03): 567- 570

[5] 胡圣荣, 周霭如. 数据结构教程与题解(用 C/C++ 描述)[M]. 北京: 北京大学出版社, 2003: 98- 105

[6] 陈朋. 后序遍历二叉树的递归和非递归算法[J]. 安庆师范学院学报: 自然科学版, 2005, 11(02): 106- 107

[7] 尹德辉, 孟林, 李忠. 二叉树后序遍历的非递归化算法讨论[J]. 西南民族大学学报: 自然科学版, 2003, 29(05): 537- 538