构造哈夫曼树算法的实现

侯永莉

(内蒙古科技大学包头师范学院 信息科学与技术学院,内蒙古 包头 014030)

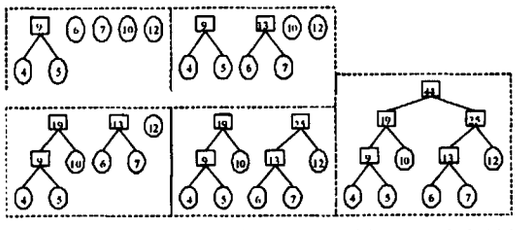
[摘 要] 树型结构是数据结构中最为重要的一种类型,在计算机科学中的应用很广。哈夫曼树又称最优二叉树,是带权路径长度最小的二叉树,可以用来构造最优编码,用于信息传输、数据压缩等方面,是一种应用广泛的二叉树。本文介绍了采用静态三叉链表存储哈夫曼树的构造算法思想与实现,并对算法做了必要的分析。

[关键词] 哈夫曼;算法;数据结构;静态三叉链表

1、构造算法的实现

哈夫曼树是由n个带权叶子结点构成的所有二叉树中带权路径长度最短的二叉树。根据定义必须使权值越大的叶子结点越靠近根结点,而权值越小的叶子结点越远离根结点,这样计算树的带权路径长度时,自然会具有最小带权路径长度,这种生成算法就是一种典型的贪心算法。

1.1 构造哈夫曼树的算法思想。(1)初始化:由给定的n个权值{w1,w2,…,wn}构造n棵只有一个根结点的二叉树,从而得到一个二叉树的集合(森林)F={T_1,T_2,…,T_n};(2)选取与合并:在F中选取根结点的权值最小的两棵二叉树分别作左、右子树构造一棵新的二叉树,这棵新的二叉树的根结点的主值为其左、右子树根结点的权值之和;(3)删除与并入:在集合F中删除作为左、右子树的两棵二叉树,并将新的二叉树加入到集合F中;(4)重复(2)、(3)两步,当集合F中只剩下一棵二叉树时,这棵二叉树便是哈夫曼树;给定一组权值{4,5,6,7,10,12},用上述方法构造一棵哈夫曼树如下图所示:



1.2 算法实现。哈夫曼树是一种二叉树,由于哈夫曼树中没有度为一的结点,因此一棵有n个叶子结点的哈夫曼树共有2n-1个结点。本算法采用静态三叉链表作为哈夫曼树的存储结构,用一个大小为2n-1的一维数组存放哈夫曼树的各个结点,每个结点包含双亲和孩子结点的信息,结点类型定义如下:

```
typedef struct
{
int weight;      /*结点的权值*/
int parent;      /*双亲的下标*/
int Lchild;      /*左孩子结点的下标*/
int Rchild;      /*右孩子结点的下标*/
}HTNode, HuffmanTree[M+1];
```

用ht[]数组存放哈夫曼树,从当前结点集合(森林)ht[1]~ht[i-1]中选择两棵根(即ht[].parant=0)且权值weight最小的树,利用形参指针*x和*y返回两个根结点的下标,选择算法如下:

```
void select(HuffmanTree ht, int n, int *x, int *y)
{
int j, min1=100, min2=100;
for(j=1;j<=n;j++)
if((ht[j].parent==0)&&(ht[j].weight<min1)) /*找到第一个权值最小的根结点*/
{
min1=ht[j].weight;
*x=j;    /*返回此结点的位置*/
}
for(j=1;j<=n;j++)
if(j!=*x && ht[j].weight<min2 && ht[j].parent==0) /*找到第二个权值最小的根结点*/
{
min2=ht[j].weight;
*y=j;    /*返回此结点的位置*/
}
}
```

ht[]数组中n个叶子结点存放在ht[1]~ht[n]中,初始化只有weight域值,将所有2n-1个结点的parent、Lchild、Rchild域置为初值0。每个非叶子结点ht[i](存放在ht[n+1]~ht[2n-1]中):调用select(ht,i-1,&s1,&s2)从ht[1]~ht[i-1]中找出根结点最小的两个结点ht[s1]和ht[s2],将它们作为ht[i]的左右子树,同时修改ht[s1]和ht[s2]的双亲parent域的值为i,并且ht[i].weight=ht[s1].weight+ht[s2].weight。如此反复,直到所有2n-1个非叶子结点处理完毕。构造哈夫曼树的算法如下:

```
void CrtHuffmanTree(HuffmanTree ht, int w[], int n)
{  int i, m, s1, s2;
for(i=1;i<=n;i++) /*初始化n个叶结点*/
{
ht[i].weight=w[i];
ht[i].parent=0;
ht[i].Lchild=0;
ht[i].Rchild=0;
}
m=2*n-1;
for(i=n+1;i<=m;i++) /*初始化后n-1个非叶结点*/
{
ht[i].weight=0;
ht[i].parent=0;
ht[i].Lchild=0;
ht[i].Rchild=0;
}
for(i=1;i<=m;i++) /*打印哈夫曼树的初态*/
printf("%d\t%d\t%d\t%d\n", ht[i].weight, ht[i].parent, ht[i].Lchild, ht[i].Rchild);
for(i=n+1;i<=m;i++) /*构造哈夫曼树,创建非叶结点*/
{
select(ht,i-1,&s1,&s2);  /*在ht[1]~ht[i-1]中选择权值最小的根结点,将位置返回*/
ht[i].weight=ht[s1].weight+ht[s2].weight;  /*修改新建根结点的权值*/
ht[s1].parent=i; ht[s2].parent=i;  /*修改两个最小结点的双亲*/
ht[i].Lchild=s1; ht[i].Rchild=s2;  /*修改新建结点的左右子树值*/
}
for(i=1;i<=m;i++) /*打印哈夫曼树的终态*/
printf("%d\t%d\t%d\t%d\n", ht[i].weight, ht[i].parent, ht[i].Lchild, ht[i].Rchild);
}
```

2、算法分析

此算法的时间主要消耗在从结点集合(森林)中选择权值最小的两棵根结点上,对每一个结点首先要判断是根结点(即parent=0)然后比较权值的大小,这一过程是用循环嵌套完成,每构建一个非叶结点(i从n+1开始)需要比较i-1次,构建n-1个非叶结点总的比较次数为(3n-2)(n-1)/2次,因此算法的时间复杂度为$O(n^2)$,空间复杂度为O(1)。

【参考文献】

[1]耿国华.数据结构——C语言描述[M].高等教育出版社,2005.
[2]陈德裕.数据结构学习指导与习题集[M].清华大学出版社,2010.
[3]李春葆.数据结构习题与解析(第3版)[M].清华大学出版社,2006.

作者简介:侯永莉,内蒙古科技大学包头师范学院信息科学与技术学院。