南京邮电大学

实 验 报 告

(2017 / 2018 学年 第 一 学期)

课程名称	数据结构A
实验名称	二叉树的基本操作及哈夫曼编码译码实现
实验时间	2017年10月10日
指导单位	计算机软件与工程学院
指导老师	邹志强

学生姓名	柏超宇	班级学号	Q15010125
学院	贝尔英才学院	专业	信息安全

实验名称	二叉树的基本操作及哈夫曼编码译码实现			指导教师	邹志强
实验类型	验证	实验学时	2+2	实验时间	2017.10.10

一、实验目的和要求:

目的:

1.掌握二叉树的二叉链表储存表示，及遍历操作实现方法
2.实现二叉树遍历运算的应用：求二叉树中叶子节点个数，节点总数，二叉树的高度，交换二叉树的左右子树等。
3.掌握二叉树的应用———哈夫曼编码的实现

二、实验环境(实验设备)

macOS High Sierra10.13
sublime text 3 + Sublime Clang

三、实验原理及内容

原理:

在计算机科学中，二叉树（英语：Binary tree）是每个节点最多只有两个分支(不存在分支度大于2的节点)的树结构。通常分支被称作“左子树”和“右子树”。二叉树的分支具有左右次序，不能颠倒。

与普通树不同，普通树的节点个数至少为1，而二叉树的节点个数可以为0；普通树节点的最大分支度没有限制，而二叉树节点的最大分支度为2；普通树的节点无左、右次序之分，而二叉树的节点有左、右次序之分。

实验内容：

完成二叉树的先序创建，先序遍历，中序遍历，后序遍历，求二叉树的节点个数，叶子节点个数，求二叉树高度以及交换二叉树所有左右子树的操作。

实验代码：

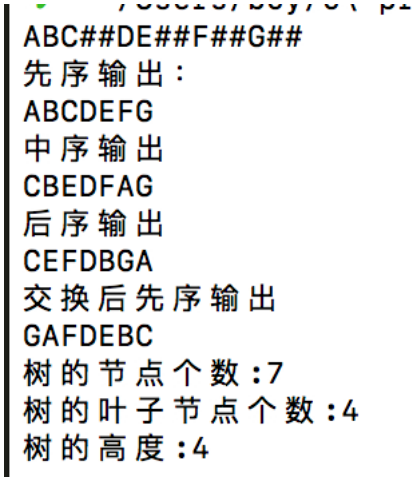
```
#include <stdio.h>
#include <stdlib.h>
typedef struct BinaryTreeNode {
    char data;
    struct BinaryTreeNode *LChild, *RChild;
} BinaryTreeNode, *BinTree;
//先序创建
BinaryTreeNode *PreCreateBt(BinaryTreeNode *t) {
    char ch;
    ch = getchar();
    if (ch == '#')
        t = NULL;
    else {
        t = (BinaryTreeNode *)malloc(sizeof(BinaryTreeNode));
        t->data = ch;
        t->LChild = PreCreateBt(t->LChild);
        t->RChild = PreCreateBt(t->RChild);
    }
    return t;
}
//先序输出
void PreOrderTransverse(BinTree t) {
    if (t == NULL) return;
    printf("%c", t->data);
    PreOrderTransverse(t->LChild);
    PreOrderTransverse(t->RChild);
}
//中序输出
void InOrderTransverse(BinTree t) {
    if (t == NULL) return;
    InOrderTransverse(t->LChild);
    printf("%c", t->data);
    InOrderTransverse(t->RChild);
}
//后序输出
void PostOrderTransverse(BinTree t) {
    if (t == NULL) return;
    PostOrderTransverse(t->LChild);
    PostOrderTransverse(t->RChild);
    printf("%c", t->data);
}
//求二叉树节点个数
int SSize(BinTree t) {
    int nodes = 0;
    if(t == NULL) return 0;
    nodes = SSize(t->LChild) + SSize(t->RChild) + 1;
    return nodes;
}
//求二叉树叶子节点个数
int numberOfLeafs(BinTree t) {
    if(t == NULL) {
```

```
        return 0;
    } else if((t->LChild == NULL) && (t->RChild == NULL)) {
        return 1;
    } else {
        return numberOfLeafs(t->LChild) + numberOfLeafs(t->RChild);
    }
}
//求二叉树高度
int HeightOfLeafs(BinTree t) {
    if (t == NULL) {
        return 0;
    } else if((t->LChild == NULL) && (t->RChild == NULL)) {
        return 1;
    } else {
        return (HeightOfLeafs(t->LChild) > HeightOfLeafs(t->RChild) ?
                HeightOfLeafs(t->LChild) : HeightOfLeafs(t->RChild)) + 1;
    }
}
//求镜像二叉树
int exchange(BinTree t) {
    BinaryTreeNode *temp ;
    if(t == NULL) {
        return 0;
    }
    if((t->LChild == NULL) && (t->RChild == NULL)) {
        return 0;
    }
    temp = t->LChild;
    t->LChild = t->RChild;
    t->RChild = temp;
    if(t->RChild) exchange(t->RChild);
    if(t->LChild) exchange(t->LChild);
    return 0;

}
//主函数
int main() {
   BinaryTreeNode *tree;
     tree = PreCreateBt(tree);
    printf("先序输出: \n");
    PreOrderTransverse(tree);
    printf("\n");
    printf("中序输出\n");
    InOrderTransverse(tree);
    printf("\n");
    printf("后序输出\n");
    PostOrderTransverse(tree);
    printf("\n");
    exchange(tree);
    printf("交换后先序输出\n");
    InOrderTransverse(tree);
    printf("\n");
    printf("树的节点个数:%d\n", SSize(tree));
    printf("树的叶子节点个数:%d\n",numberOfLeafs(tree));

}
```

运行截图：



实现了我们想要的功能

哈夫曼树的编码和解码

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define N 10000

typedef struct {
    int weight;
    int parent, lchild, rchild;
} hafuman;

typedef struct {
    char data[N]; //数据
    char copy[N][10 * N]; //编码
} bianma;

void display();
int input(int w[], bianma *bm);
void creat_hafuman(hafuman ht[], int w[], int n);
void select(hafuman ht[], int m, int *s1,  int *s2);
void encoding(hafuman ht[], bianma *bm, int n);
void coding(bianma *bm, int n);
void codingcoding(hafuman ht[], bianma *bm, int n);//译码
void output(bianma *bm, int n);
int i, j, k;

void display() {
    printf("\n\n\n");
    printf("1.输出编码\n");
    printf("2.译码\n");
    printf("3.exit\n");
    printf("请选择(1~~3):  ");
}
```

```
int input(int w[], bianma *bm) {
    int n = 0;

    printf("\n请输入要进行编码句子(#结束)\n");

    while(1) {
        bm->data[n] = getchar();

        if(bm->data[n] == '#')
            break;
        n++;
    }
    for(i = 0; i < n; i++) {
        w[i] = 1;

        for(j = i + 1; j < n; ) {
            if( bm->data[i] == bm->data[j] ) {
                w[i]++;

                for(k = j; k < n; k++) {
                    bm->data[k] = bm->data[k + 1];
                }
                n--;
            } else
                j++;
        }
    }
    printf("\n\n");
    printf("不同的字符\n");
    for(i = 0; i < n; i++) {
        printf("%c", bm->data[i]);
    }
    return n;
}
void creat_hafuman(hafuman ht[], int w[], int n) {
    int s1, s2;
    int t;
    for(t = 1; t <= n; t++) {
        ht[t].weight = w[t - 1];
        ht[t].parent = 0;
        ht[t].lchild = 0;
        ht[t].rchild = 0;
    }
    for(t = n + 1; t <= 2 * n - 1; t++) {
        ht[t].weight = 0;
        ht[t].parent = 0;
        ht[t].lchild = 0;
        ht[t].rchild = 0;
    }
    for(t = n + 1; t <= 2 * n - 1; t++) {
        select(ht, t - 1, &s1, &s2);
        ht[t].weight = ht[s1].weight + ht[s2].weight;
        ht[t].lchild = s1, ht[t].rchild = s2;
        ht[s1].parent = t, ht[s2].parent = t;
    }
}
void select(hafuman ht[], int m, int *s1,  int *s2) {
    int min1, min2, a, b;
    i = 1;
    while( ht[i].parent != 0) {
```

```
        i++;
    }
    min1 = ht[i].weight;
    a = i;
    for(j = i + 1; j <= m; j++) {
        if(min1 > ht[j].weight && ht[j].parent == 0) {
            min1 = ht[j].weight;
            a = j;
        }
    }
    i = 1;
    while( ht[i].parent != 0 || a == i ) {
        i++;
    }
    min2 = ht[i].weight;
    b = i;
    for(j = i + 1; j <= m; j++) {
        if(j == a)
            continue;
        if(min2 > ht[j].weight && ht[j].parent == 0) {
            min2 = ht[j].weight;
            b = j;
        }
    }
    *s1 = a;
    *s2 = b;

}

void encoding(hafuman ht[], bianma *bm, int n) {
    int start, c, p;
    char *ch;
    ch = (char *)malloc( n * sizeof(char) );
    ch[n - 1] = '\0';

    for(i = 1; i <= n; i++) {
        start = n - 1;
        c = i, p = ht[i].parent;

        while(p != 0) {
            start--;
            if(ht[p].lchild == c)
                ch[start] = '0';
            else
                ch[start] = '1';

            c = p;
            p = ht[p].parent;
        }

        strcpy( bm->copy[i - 1], &ch[start] );

    }
    free(ch);

}

void codingcoding(hafuman ht[], bianma *bm, int n) {
    char s[10 * N];
    int p;
```

```
    printf("\n请输入要译码的字符\n");
    fflush(stdin);
    scanf("%s", s);
    printf("\n译码\n\n");
    p = 2 * n - 1;
    for(i = 0; s[i] != '\0'; i++) {

        if(s[i] == '0')
            p = ht[p].lchild;

        else if(s[i] == '1')
            p = ht[p].rchild;

        if(ht[p].lchild == 0 && ht[p].rchild == 0) {
            printf("%c", bm->data[p - 1]);
            p = 2 * n - 1;
            continue;
        }

    }
    puts("\n\n");
}

void output(bianma *bm, int n) {
    printf("\n");
    for(i = 0; i < n; i++) {
        printf("%c\t", bm->data[i] );

        printf("%s\n", bm->copy[i]);
    }
}

int main() {
    hafuman ht[N];
    bianma *bm;
    int w[N];
    int n, m;
    bm = (bianma *)malloc( sizeof(bianma) );
    n = input(w, bm);
    printf("\n不同字符总数: %d\n", n);
    creat_hafuman(ht, w, n);
    encoding(ht, bm, n);
    getchar();
    while(1) {
        display();
        scanf("%d",  &m);
        switch(m) {
        case 1:
            output(bm, n);
            printf("\n请按任意键继续");
            getchar();
            break;
        case 2:
            codingcoding(ht, bm, n);
            printf("\n请按任意键继续");
            getchar();
            break;
        case 3:
            return 0;
```

```
        default:
            printf("ERROR\n");
        }
    }
}
```

运行截图：

```
请输入句子或文章(以#结束)
Online social networks such as Facebook and Twitter have started allowing users t
o tag their posts with geographical coordinates collected through the GPS interfa
ce of users smartphones. While this information is quite useful and already indic
ative of user behavior, it also lacks some semantics about the type of place the
user is (e.g., restaurant, museum, school) which would allow a better understandi
ng of users' patterns. While some location based online social network services (
e.g., Foursquare) allow users to tag the places they visit, this is not an automa
ted process but one which requires the user help.#

不同字符总数： 38


1.输出编码
2.译码
3.exit
请选择(1~~3):  1

O       010101100
n       11110
l       11111
i       0110
e       001
        110
s       1110
o       1000
c       01111
a       1001
t       1010
w       011101
r       0100
k       0101000
u       10110
h       0000
F       01010010
b       1011110
d       101110
T       010101101
v       0111001
g       010110
p       010111
G       010101110
P       010101111
S       011100000
f       000101
m       000111
.       000110
W       01010011
q       0001000
y       0001001
,       1011111
(       01010100
)       01010101
```

```
1.输出编码
2.译码
3.exit
请选择(1~~3):  2

请输入密码
10111100111110001001

译码
bcy
```

根据实验结果我们可以看出代码成功实现了我们想要的功能，通过一个界面我们也可以轻松的知道每一步需要的输入，对于空格等特殊字符程序也能正确的完成编码译码操作。

实验小结

通过本次数据结构实验，让我进一步深入了对二叉树的理解。通过查阅资料我也学会使用了哈夫曼树这种这样一种非常高效的数据结构，增加了理论和实际之间的联系。哈夫曼树在调试的过程中还是遇到了一些困难的，今后应该更加注重自己代码能力的培养，能够想出来的算法争取可以实现出来。

五、指导教师评语

成绩		批阅人		日期	