一、【实验目的】  
1、理解哈夫曼树的基本概念

2、掌握哈夫曼树的构造及数据结构设计

3、掌握哈夫曼编码问题设计和实现

二、【实验内容】  
1、假设用于通信的电文仅由8个字母 {a, b, c, d, e, f, g, h} 构成，它们在电文中出现的概率分别为{ 0.07, 0.19, 0.02, 0.06, 0.32, 0.03, 0.21, 0.10 }，试为这8个字母设计哈夫曼编码。

提示:包含两个过程:（1）构建哈夫曼树，（2）输出编码。

三、【实验源代码】

 #include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct node {

    char letter;

    float freq;

    struct node \*left, \*right;

};

void swap\_node(struct node \*\*a, struct node \*\*b) {

    struct node \*temp = \*a;

    \*a = \*b;

    \*b = temp;

}

void sort\_nodes(struct node \*\*nodes, int n) {

    int i, j;

    for (i = 0; i < n; i++) {

        for (j = i + 1; j < n; j++) {

            if (nodes[j]->freq < nodes[i]->freq) {

                swap\_node(&nodes[i], &nodes[j]);

            }

        }

    }

}

void print\_codes(struct node \*root, char \*code) {

    if (root == NULL) {

        return;

    }

    if (root->letter != '\0') {

        printf("%c: %s\n", root->letter, code);

    }

    char \*left\_code = malloc(strlen(code) + 2);

    strcpy(left\_code, code);

    strcat(left\_code, "0");

    print\_codes(root->left, left\_code);

    char \*right\_code = malloc(strlen(code) + 2);

    strcpy(right\_code, code);

    strcat(right\_code, "1");

    print\_codes(root->right, right\_code);

    free(left\_code);

    free(right\_code);

}

void huffman(struct node \*\*nodes, int n) {

    sort\_nodes(nodes, n);

    while (n > 1) {

        struct node \*new\_node = malloc(sizeof(struct node));

        new\_node->left = nodes[0];

        new\_node->right = nodes[1];

        new\_node->freq = nodes[0]->freq + nodes[1]->freq;

        new\_node->letter = '\0';

        nodes[0] = new\_node;

        n--;

        int i;

        for (i = 1; i < n; i++) {

            nodes[i] = nodes[i+1];

        }

        sort\_nodes(nodes, n);

    }

}

int main() {

    const char letters[8] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'};

    const float freqs[8] = {0.07, 0.19, 0.02, 0.06, 0.32, 0.03, 0.21, 0.10};

    int i, n = 8;

    struct node \*\*nodes = malloc(n \* sizeof(struct node \*));

    for (i = 0; i < n; i++) {

        struct node \*new\_node = malloc(sizeof(struct node));

        new\_node->letter = letters[i];

        new\_node->freq = freqs[i];

        new\_node->left = NULL;

        new\_node->right = NULL;

        nodes[i] = new\_node;

    }

    huffman(nodes, n);

    print\_codes(nodes[0], "");

    for (i = 0; i < n; i++) {

        free(nodes[i]);

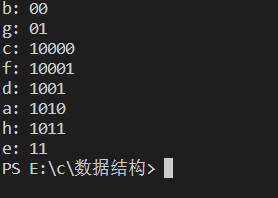
    }

    free(nodes);

    return 0;

}

四、【实验结果】



五、【实验心得】

定义了一个结构体 来表示哈夫曼树的节点，字母、频率以及左右子树。然后，实现了 swap\_node 和 sort\_nodes 函数来排序节点。接着，定义了一个递归函数 print\_codes 来遍历哈夫曼树并输出每个字母的编码。最后，搞定，有思路好做些