问题 A: DS二叉树--赫夫曼树的构建与编码

题目描述

给定n个字母及相应权值（频次），根据这些权值构造huffman树，并进行huffman编码
要求：赫夫曼的构建中，默认左孩子权值不大于右孩子权值

输入

第1行先输入n，表示有n个字母。
第2行输入n个字母。
第3行输入n个权值，其中第i个权值表示第2行中第i个字母的频次。
第4行输入一个字符串，表示需要编码的字符串

输出

前n行按输入顺序逐行输出每个字母对应的编码，格式如下：字母 编码
即每行先输出1个字母，再输出一个空格，最后输出对应编码，接着下一行输入下一个权值和编码。以此类推。
第n + 1行输出编码后的字符串。

样例输入

```
4
a b c d
9 3 2 6
abcd
```

样例输出

```
a 0
b 101
c 100
d 11
010110011
```

运行效果

```
4                   输入字符个数
a b c d             输入字符
9 3 2 6             输入字符权值
a 0                 输出第1个字符'a'的编码
b 101               输出第2个字符'b'的编码
c 100               输出第3个字符'c'的编码
d 11                输出第4个字符'd'的编码
abcd                输入待编码字符串
010110011           输出字符串的编码
```

程序模板

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define OK 0
#define ERROR -1

#define N 1005
#define LEN 15

struct HuffmanNode {
    char c;          // 字符（只有叶子节点才有）
    int freq;        // 频次（权值）
    int f, l, r;     // f父节点下标，l左孩子下标，r右孩子下标
    int vis;         // 当前节点是否被选择，1表示已选择，0表示未选择
    char code[LEN]; // 节点的编码值（只有叶子节点才有）
};

typedef struct HuffmanNode Node;

Node tree[2 * N];  ← tree 的数组，每格放Node.

// 选择函数
// 功能：返回未选过的节点中权值最小的节点的下标
int selectSmallestOne(Node tree[], int size) {
    // maxFreq表示未选过的节点中的最小频次，ret表示最小频次节点的下标
    // 现将minFreq初始化为一个很大的数
    int minFreq = 0x3f3f3f3f, ret = 0;

    for (int i = 0; i < size; i++)  // 遍历前size个节点，找出频次最小且未选过的节点
        ...                         // 如果第i个节点未选过，且频次小于minFreq，更新ret和minFreq

    ...                             // tree[ret]已被选择

    return ret;                     // 返回下标
}
```

```
// 创建Huffman树
// 根据字符数组c和权值数组freq创建一棵Huffman树，c和freq数组长度为n
void createTree(Node tree[], int n, char c[], int freq[]){
    int m, s1, s2;
    m = 2 * n - 1;      // 最终有2 * n - 1个节点

    for (int i = 0; i < m; i++){                    // 所有节点初始化
        tree[i].c = tree[i].freq = tree[i].f = tree[i].l = tree[i].r = -1;
        tree[i].vis = 0;                            // 0表示未访问；1表示已访问
    }

    for (int i = 0; i < n; i++) {                   // 前n个节点为叶子节点，赋值对应的频次和字符
        tree[i].c = c[i];
        tree[i].freq = freq[i];
    }

    for (int i = n; i < m; i++){                    // 每次取两个频次最低且未选过的节点，合并生成
        第i个节点
        s1 = selectSmallestOne(tree, i);
        s2 = selectSmallestOne(tree, i);
        ...             // 合并操作（s1、s2节点的父节点指向i，i节点左右孩子分别为s1、s2，i的权值为
            s1、s2权值总和）
    }
}

// 给叶子节点编码
void generateCodes(Node tree[], int n){
    int f, c;

    for (int i = 0; i < n; i++){
        char code[LEN] = "";        // code一开始为空

        ...     // 从叶子节点往根节点生成编码

        ...     // 将code字符串翻转，如"abcde"变成"edcba", "01101"变成"10110"

        strcpy(tree[i].code, code);         // 将code复制给字符编码
    }
}
```

```c
// 输出各叶子节点编码
void showCodes(Node tree[], int n){
    for (int i = 0; i < n; i++)
        printf("%c %s\n", tree[i].c, tree[i].code);
}

// 给字符串src进行编码
void encoding(Node tree[], int n, char src[], char dst[]){
    dst[0] = 0;

    for (int i = 0; i < strlen(src); i++)       // 遍历来源串中的字符
        ...             // 在huffman树中找到字符对应的编码，使用strcat函数将其接到dst末尾
}

int main(){
    int n;
    char c[N];
    int freq[N];

    scanf("%d", &n);
    for (int i=0;i<n;i++){
        getchar();
        scanf("%c", &c[i]);
    }
    for (int i=0;i<n;i++)
        scanf("%d", &freq[i]);

    createTree(tree, n, c, freq);
    generateCodes(tree, n);
    showCodes(tree, n);

    char src[N], dst[N];
    scanf("%s", src);
    encoding(tree, n, src, dst);
    printf("%s\n", dst);

    return 0;
}
```