《离散数学》课程实验报告 5 最优2元树在通信编码中的应用

实验内容

输入一组通信符号的使用频率，求各通信符号对应的前缀码。

实验原理和方法

> 二元前缀码：任何字符的代码不能作为其它字符代码的前缀

为求二元前缀码，我们需要寻找最优二元树，即构造霍夫曼树。

> 当用 n 个结点（都做叶子结点且都有各自的权值）试图构建一棵树时，如果构建的这棵树的带>权路径长度最小，称这棵树为“最优二叉树”，有时也叫“赫夫曼树”或者“哈夫曼树”。

在构建哈弗曼树时，要使树的带权路径长度最小，只需要遵循一个原则，那就是：权重越大的结点离树根越近。

构建霍夫曼树的过程：

对于给定的有各自权值的 n 个结点，构建哈夫曼树有一个行之有效的办法：

1. 在 n 个权值中选出两个最小的权值，对应的两个结点组成一个新的二叉树，且新二叉树的根结点的权值为左右孩子权值的和；
2. 在原有的 n 个权值中删除那两个最小的权值，同时将新的权值加入到 n–2 个权值的行列中，以此类推；
3. 重复 1 和 2 ，直到所有的结点构建成了一棵二叉树为止，这棵树就是哈夫曼树。

算法实现

（1）用一维数组f[N]存贮通信符号的使用频率，用求最优2元树的方法求出每个通信符号的前缀码。
（2）用链表保存最优2元树，输出前缀码时可以用树的遍历方法。

C++语言源代码

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<vector>

const int N =13;

struct tree
{
```

```
    int num;
    struct tree *Lnode;
    struct tree *Rnode;
}* fp[N];     //保存结点
char s[2*N];  //放前缀码

void init_node(std::vector<int>f,int n)  //生成叶子结点
{
    int i;
    struct tree *pt;
    for(i = 0;i < n;i++)
    {
        pt=new struct tree; //生成叶子结点
        pt->num = f[i];
        pt->Lnode = NULL;
        pt->Rnode = NULL;
        fp[i] = pt;
    }
}

void sort (struct tree * array[],int n)  //将第N-n个点插入到已排好序的序列中
{
    int i;
    struct tree *temp;
    for(i = N-n;i < N-1;i++)
        if(array[i]->num > array[i+1]->num)
        {
            temp = array [i+1];
            array[i+1] = array[i];
            array[i] = temp;
        }
}

struct tree * construct_tree(std::vector<int>f,int n)  //建立树
{
    int i;
    struct tree *pt;
    for(i = 1;i < N;i++)
    {
        pt=new struct tree; //生成非叶子结点
            pt->num = fp[i-1]->num + fp[i]->num;
            pt->Lnode = fp[i-1];
            pt->Rnode = fp[i];
            fp[i] = pt;  //w1+w2
        sort(fp,N-i);
    }
    return fp[N-1];
}

void preorder (struct tree *p, int k, char c)
{
    int j;
    if(p != NULL)
    {
        if(c == '1')
            s[k] = '0';
        else s[k] = '1';
        if (p->Lnode == NULL)
```

```
        {   //P 指向叶子
            std::cout<<p->num<<": ";
            for(j = 0;j <= k;j++)
                std::cout<<s[j];
            putchar('\n');
        }
        preorder (p->Lnode, k+1,'1');
        preorder (p->Rnode, k+1,'r');
    }
}

int main()
{
    int n;
    std::cout<<"请输入节点个数(必须是正整数):";
    std::cin>>n;
    std::vector<int> f;
    std::cout<<"请输入节点(以空格分隔):";
    for(int i = 0;i < n;i++)
    {
        int temp;
        std::cin>>temp;
        f.push_back(temp);
    }
    struct tree *head;
    init_node(f,N);  //初始化结点
    head = construct_tree(f,N);  //生成最优树
    s[0] = 0;
    preorder(head,0,'1');  //遍历树
    return 0;
}
```

实验结果

```
输入节点个数:13
输入节点:2 3 5 7 11 13 17 19 23 29 31 37 41
19:  0000
23:  0001
11:  00100
13:  00101
29:  0011
31:  0100
7:  010100
2:  01010100
3:  01010101
5:  0101011
17:  01011
37:  0110
41:  0111
请按任意键继续. . .
```