**《离散数学》求最优二元树项目文档**

1. **题目简介**

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。

1. **实验原理**
2. Huffman树基本概念

路径长度(Path Length)：两个结点之间的路径长度是连接两结点的路径上的分支数。树的路径长度是各叶结点到根结点的路径长度之和。带权路径长度达到最小的扩充二叉树即为霍夫曼树。在霍夫曼树中，权值大的结点离根最近。

2.霍夫曼编码

主要用途是实现数据压缩。设给出一段报文:CAST CAST SATATA TASA

字符集合是{C,A, S,T}，各个字符出现的频度(次数)是W={ 2,7,4,5}。

若给每个字符以等长编码A: 00 T: 10 C:01 S : 11；则总编码长度为(2+7+4+5 )\*2=36；若按各个字符出现的概率不同而给予不等长编码，可望减少总编码长度。因各字符出现的概率为{2/18,7/18,4/18,5/18}。化整为{2，7，4，5}，以它们为各叶结点上的权值，建立霍夫曼树。左分支赋0，右分支赋1，的霍夫曼编码（变长编码）。计算可得总编码长度为35.比等长编码的情形要短。

总编码长度正好等于霍夫曼树的带权路径长度。霍夫曼编码是一种无前缀编码。解码时不会混淆。由此生成的树就是霍夫曼编码树。

1. **核心算法**

将所有节点作为叶子结点输入：

void init\_node(int f[], int n) //生成叶子结点

{

int i;

struct tree\* pt;

for (i = 0; i < n; i++)

{

pt = (struct tree\*)malloc(sizeof(struct tree)); //生成叶子结点

pt->num = f[i];

pt->Lnode = NULL;

pt->Rnode = NULL;

fp[i] = pt;

}

}

首先对输入的频率按照从小到大的顺序排序（此处采用插入排序法），接着生成最优二叉树:在所有入度为0的顶点(不一定是树叶)中选出两个权最小的顶点，添加一个新分支点，以这2个顶点为儿子，其权等于这2个儿子的权之和。接着利用循环结构重复此步骤,直到只有1个入度为О的顶点为止。

最后通过遍历树，生成并输出各节点的编码:本程序中使用了前序遍历(遍历的方式不影响编码结果)。对每个分支点，若关联2条边，则给左边标0，右边标1;若只关联1条边，则可以给它标0(看作左边)，也可以标1(看作右边)。将从树根到每一片树叶的通路上标的数字组成的字符串便构成一个前缀码。

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

cout << p->num << ":";

for (j = 0; j <= k; j++)

cout << s[j];

putchar('\n');

}

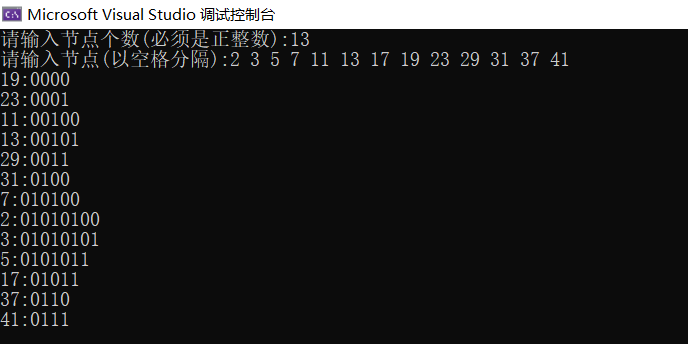
preorder(p->Lnode, k + 1, '1');

preorder(p->Rnode, k + 1, 'r');

}

}

1. **实验结果**



1. **心得体会**

将老师给的源代码转换成C++语言，和数据结构相比，数据结构要写出霍夫曼树类，而现在可以直接用一个结构体来代替，代码简洁，实现起来比较方便。其中，排序算法还可以进一步优化，可以用快速排序法等等。

通过本次项目，我对最优二元树在通信编码中应用的一些程序细节问题有了进一步思考。并理解了霍夫曼编码的具体过程和程序实现过程。