**最优二叉树（Huffman树）**

对有一组带有确定权值的叶子结点，构造具有最小带权路径长度的二叉树。

**基本概念**

路径：从一个结点到另一个结点之间的分支序列。

路径长度：从一个结点到另一个结点所经过的分支数目。

结点的权：树中每个结点所赋予的具有某种实际意义的实数。

带权路径长度：从树根到某一结点的路径长度与该结点的权的乘积。

树的带权路径长度：树中从根到所有叶子结点的各个带权路径长度之和。

*记为：WPL= Wi × Li*

*∑*

*i=1*

*n*

在所有含 n 个叶子结点、并带相同权值的二叉树，必存在一棵其带权路径长度取最小值的树，称为“最优树”，即哈夫曼树。

**算法步骤：（以二叉树为例）**

①根据给定的 n 个权值 {w1, w2, …, wn}，构造 n 棵二叉树

的集合F = {T1, T2, … , Tn}，其中每棵二叉树中均只含一

个带权值 为 wi 的根结点， 其左、右子树为空树；

②在 F 中选取其根结点的权值为最小的两棵二叉树，分别作为左、右子树构造一棵新的二叉树，并置这棵新的二叉树根结点的权值为其左、右子树根结点的权值之 和；

③从F中删去这两棵树，同时加入刚生成的新树；

④重复 (2) 和 (3) 两步，直至 F 中只含一棵树为止。

**哈夫曼编码**

编码：指用不同的0/1序列代表不同的字符，这个0/1序列就是其对应字符的编码。

译码：指将已编了码的信息还原成原来的形式。

在电文传输中，需要将电文中出现的每个字符进行二进制编码。在设**计编码时需要遵守两个原则：**

1. 要能唯一地译码。
2. 编码长度要尽量短。

等长编码 特点：每个字符的编码长度相同。

不等长编码 特点：使用频度较高的字符分配一个相对比较短的

编码，使用频度较低的字符分配一个比较长的编码。

**具体构造方法：**

设需要编码的字符集合为{d1,d2,…,dn}，各个字符在电文中出现的次数集合为{w1,w2,…,wn}，以d1,d2,…,dn作为叶结点，以w1,w2,…,wn作为各叶结点的权值构造一棵二叉树，规定哈夫曼树中的**左分支为0，右分支为1**，则从根结点到每个叶结点所经过的分支对应的0和1组成的序列便为该结点对应字符的编码。这样的编码我们称之为哈夫曼编码。

求叶结点的哈夫曼编码需从叶结点出发走一条从叶子到根的路径，每经过一个分支就得到一位哈夫曼编码值。

**存储结构：**

void CrtHuffmanTree(HuffmanTree ht,int w[ ],int n)

{ for(i=1;i<=n;i++)

ht[i]={w[i],0,0,0};

m=2\*n-1;

for(i=n+1;i<=m;i++)

ht[i]={0,0,0,0}；

for(i=n+1;i<m;i++)

{select(ht,i-1,&s1,&s2);

ht[i].weight=ht[s1].weight+ht[s2].weight;

ht[s1].parent=i;

ht[s2].parent=i;

ht[i].LChild=s1;

ht[i].RChild=s2;

}

}

void CrtHuffmanCode(huffmantree ht,huffmancode hc,int n);

{

char \*cd;

cd=(char \*)malloc((n+1)\*sizeof(char));

cd[n]= ’\0’;

for (i=1; i <=n; i ++)

{

start=n;

c= i;

p=ht[i].parent;

while (p!=0)

{

--start;

if (ht[p].lchild==c) cd[start]=‘0’;

else cd[start]=‘1’;

c=p;

p=ht[p].parent;

}

hc[i]= (char \*)malloc((n-start)\*sizeof(char));

strcpy(hc[i],&cd[start]);

}

free(cd);

}