# 1、解题思路

1.1、需求分析：

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。

1.2、实验原理：

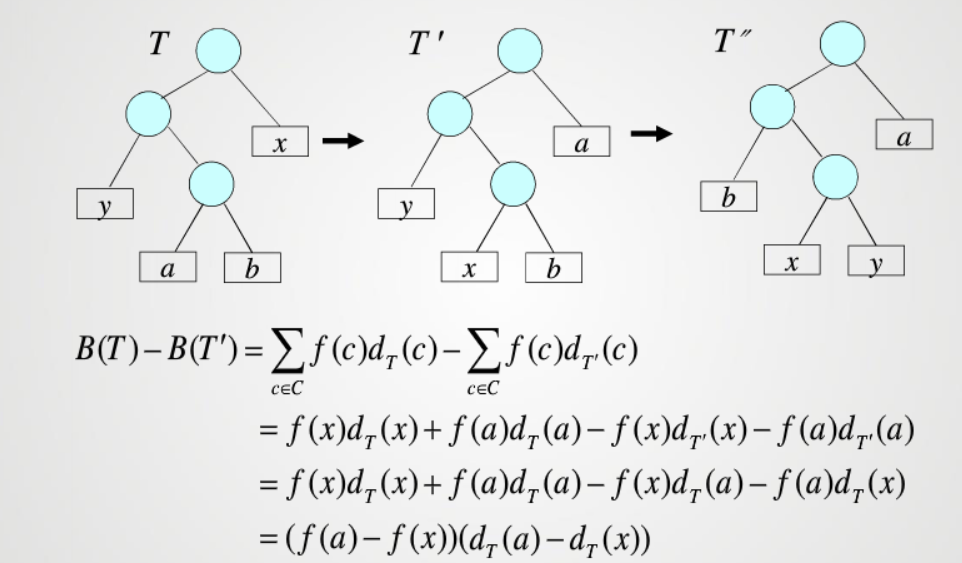
给定对应一种前缀编码的二叉树T，很容易计算出编码一个文件所需的位数。对字母表C中的每个字符c，设f (c)表示c在文件中出现的频度，d(c)表示c的叶子在树中的深度，也是字符c的编码的长度。这样，编码一个文件所需的位数就是

把它定义为树T的代价。

定理1：

令C为一个字符表。对任意的c∈C，字符c在文件中出现的频率为f(c)。令x和y为C中出现频率最小的两个字符。那么对C存在一个最优前缀编码。在这种编码中x和y的编码长度最长，且长度相等，只有最后一位不同。

证明：



定理2：

令C为一个字符表。对任意的c∈C，字符c在文件中出现的频率为f(c)。令x和y为C中出现频率最小的两个字符。从中删除x和y两个字符并加入新的字符z，得到新的字符表C'={C- {x,y}}U{z}。除了f(z)=f(x)+fv)外，C'中字符频率的定义同C。树T代表C'的最优前缀编码，对树的叶子节点z，添加两个孩子节点x和y，就可以得到树T，则T代表c的最优前缀编码。

证明：

设T为C的一种最优前缀编码，

故

若T'表示C'上的一种非最优前缀编码,则存在树T"使得B(T"')<B(T)。设z为x , y的父节点且有,则我们可以构造树T’。然后

然后因为

所以

因此

即

又因为

所以

因而

故

这与T的最优性矛盾。所以T‘表示C’一种最优前缀编码。

由定理1和定理2可知Huffman编码可以产生一种最优编码

## 1.3、设计思路：

使用静态数组存储字符出现的频率。在编码过程中，首先依照以下规则建立一棵Huffman树：从所有没有父母的节点中选出权值最小的两个节点，将它们作为一个新节点的孩子，该节点的权值等于其后代权值之和，直到只有一个节点没有父母，作为Huffman树的根节点。

然后根据建立的这一棵Huffman树计算Huffman编码并存储到Huffman编码树中。采用从叶子节点到根节点的策略。遍历所有叶子节点，从每一个叶子节点出发，走一条到根的路径，对于在路径上的每一个节点，如果是其父母节点的左孩子，则在其编码后加上“0”，否则加上“1”（初始编码为空），将这些编码储存到编码树中。

# 2、数据结构：

**Huffman树的存储结构：**

HTNode:

|  |
| --- |
| weight |
| parent |
| lc |
| rc |
| ch |
| HTNode() |

存储结构说明：采用静态三叉链表作为存储结构，weight中存放该节点的权重；parent存放该节点父亲的下标；lc存放该节点左孩子的下标；rc存放该节点右孩子的下标；ch存放该节点代表的字符。

# 3、核心算法与复杂度分析：

**1、Select函数（从没有父母的节点中选出权值最小的两个节点）：**

函数执行过程：对静态数组进行遍历，寻找权值最小且没有父母节点的两个节点，时间耗费主要来自遍历过程。

时间复杂度：，其中，N小于Huffman树的大小。

**2、Huffman编码过程：**

函数执行过程：主要分为两个部分，第一个部分：建立Huffman树；第二个部分：进行Huffman编码。在该程序中，n为固定的127。

建立Huffman树时总共要进行次合并，每一次合并时要运行一次Select函数，Select函数的时间复杂度与字符总数n呈线性关系所以时间复杂度是。而空间复杂度主要来自储存Huffman树，故为。

进行Huffman编码时，遍历所有频率值，遍历每一个频率值时，从根节点向下直到叶子节点，由于Huffman树高在最坏情况下可以达到的复杂度，故该算法的时间复杂度也是。空间复杂度主要来自储存Huffman Coding，为。

时间复杂度：

空间复杂度：

由于n为固定的常数，所以该算法的时间复杂度和空间复杂度均为常数，并且这个常数对于当今的计算机并不大，故Huffman算法本身可以在很短的时间内完成。

# 4、程序测试：

1、测试输入：

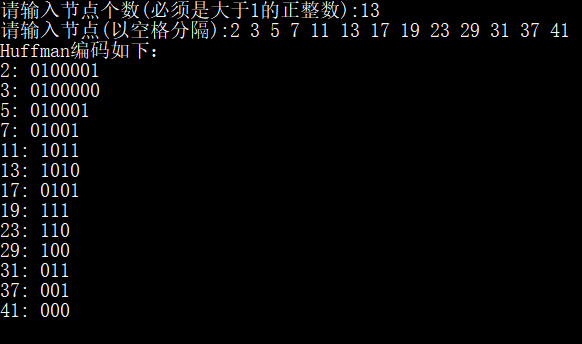
13

2 3 5 7 11 13 17 19 23 29 31 37 41

实验预期：

Huffman编码

实验结果：



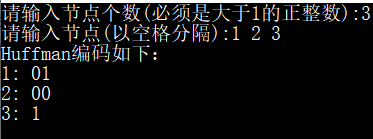
（结果与C语言代码所给不同，但是是正确的）

2、测试输入：

3

1 2 3

实验结果：

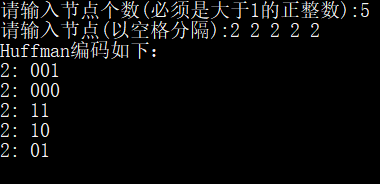


3、测试输入：

5

2 2 2 2 2

实验结果：



4、测试输入：

5

1 1000 10 100 5

实验结果：

