西南大學

最优前缀码

张里博

lbzhang@swu.edu.cn

目录

PART 01

最优前缀码问题

Q1. 给定一个文本，其包含 32 个字符 (26 英文字符，空格，其它标点符号)，如何将文本编码（寻找编码序列）？

固定长度编码（定长编码）
用长度为5位二进制的字符对原文件进行编码，2^5 =32个不同字符。

可以用不同长度的编码来表示字符，以压缩编码后的文件大小

Q2. 变长编码后的序列，解码时，怎样判断下一个字符的开始？

使用分隔符号；

有效字符编码方案：
确保没有任何一个编码是另一个编码的前缀来确保解码无二义性的编码方案，即前缀码

前缀码

什么是前缀码

前缀码特性：
任何一个字符的编码都不能是其他字符编码的前缀；

具有前缀码特性的编码即为前缀码

例

假设编码方案a=001,b=00,c=010,d=01

b是a的前缀,d是c的前缀

解码的歧义

例如：字符串 0100001
解码1:01,00,001；dba
解码2:010,00,01；cbd

二元前缀码

用0-1字符串作为代码来表示字符信息，并要求任何字符串的代码不能作为其他字符代码的前码（前缀码特性），这样的码称为二元前缀码。

含弘光大 继往开来

二元前缀码

采用二叉树存储二元前缀码

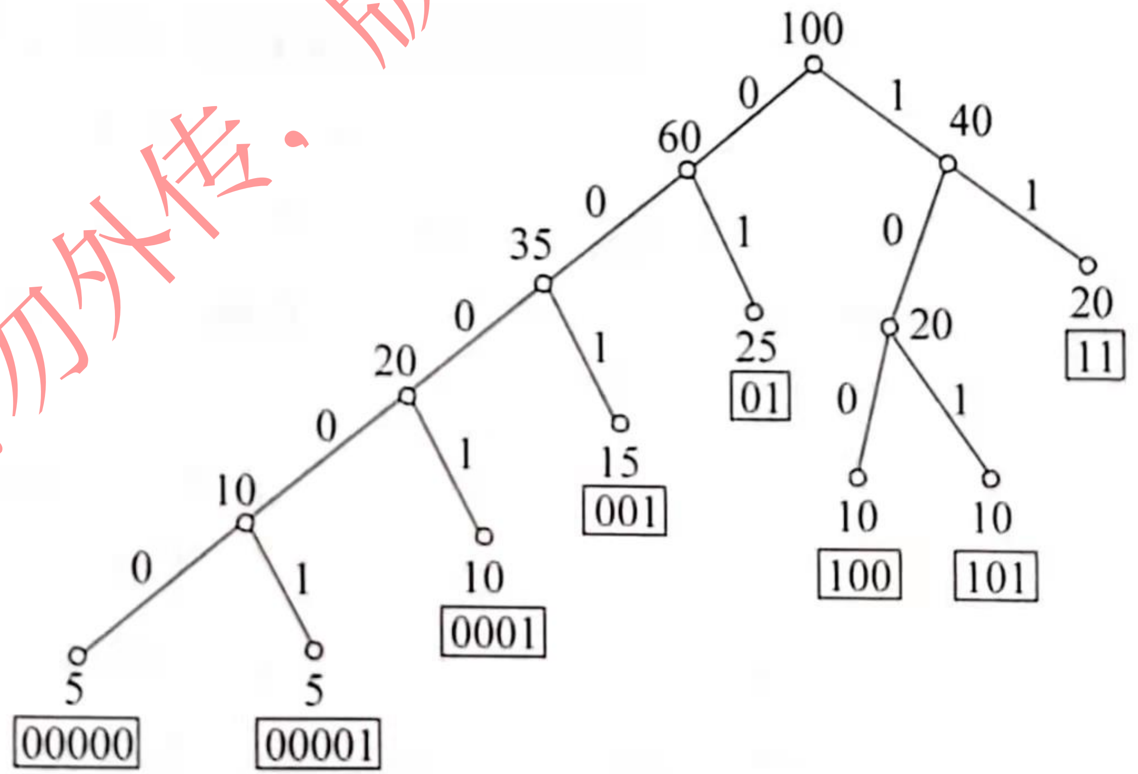
令每个字符作为树叶，对应这个字符的前缀码看作根到这片树叶的一条路径；

规定每个结点通向左儿子的边记作0，通向右儿子的边记作1；

图中这棵二叉树对应的前缀码是：

{00000，00001，0001，001，01，100，101,11}

前缀码的二进制位数即为字符的码长，也是树叶的深度。



含弘光大　继往开来

最优前缀码

不同字符在信息中出现的频率不同

设C={x_1, x_2, …, x_n}是n个字符的集合，x_i的频率是f(x_i), i=1,2,…,n,那么存储一个字符所使用的二进制位数的平均值（平均码长）是：

$$B=\sum_{i=1}^{n} f(X_i)d(X_i)$$

其中d(x_i)是表示字符x_i的二进制位数，也就是x_i的码长。

平均码长：存储一个字符的平均二进制位数称为该字符的平均码长。

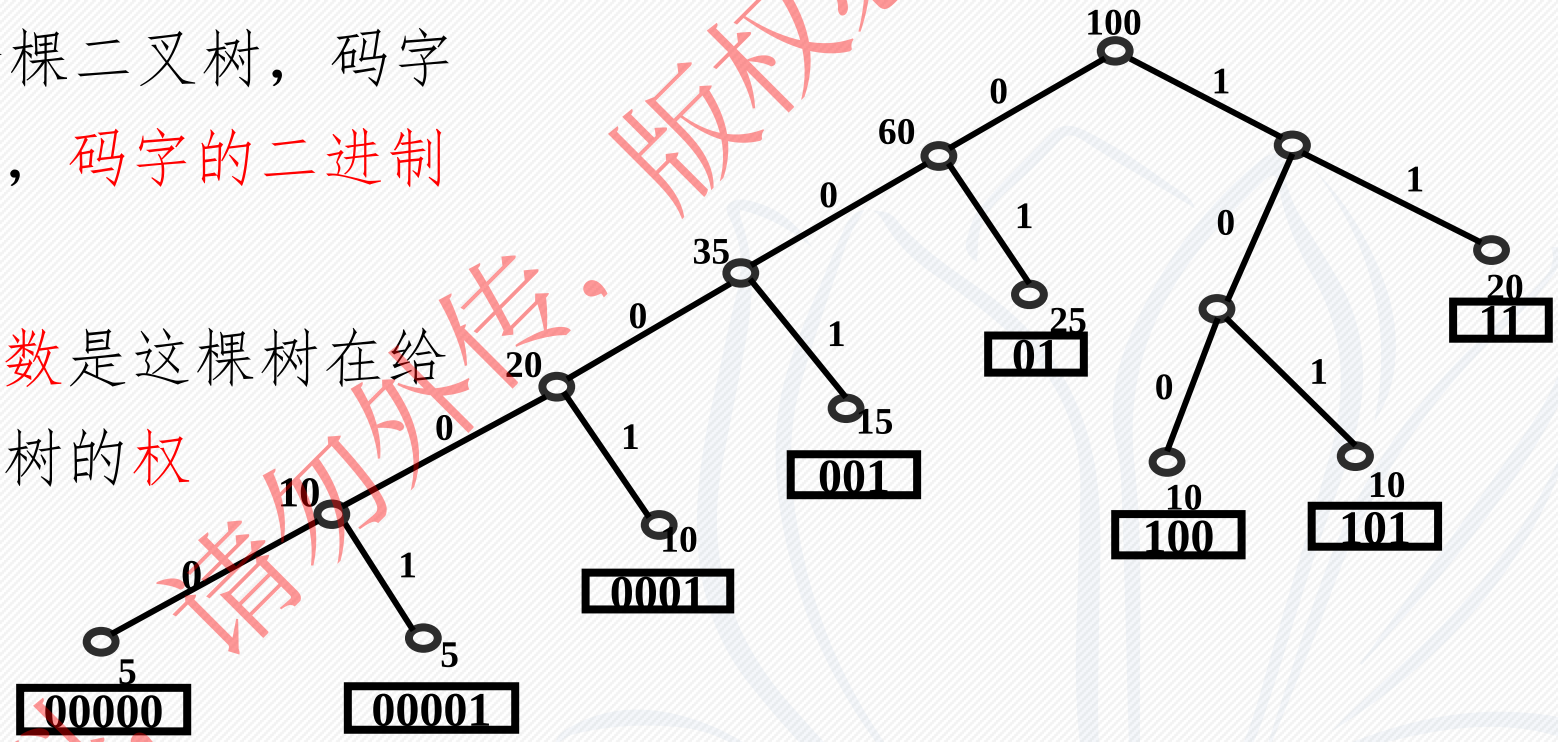
最优前缀码：平均码长达到最小的前缀码编码方案。

权值

$$B=[(5+5)\times5+10\times4+(15+10+10)\times3+(25+20)\times2]/100=2.85$$

由于一个二元前缀码对应了一棵二叉树，码字是树的树叶，频率即为树叶的权值，码字的二进制位数就是树叶的深度。

存储一个字符的平均二进制位数是这棵树在给定频率下的平均深度，也称为这棵树的权



Idea：用更少的bit来编码常用字符，而其他出现频率相对较少的字符使用相对较多的bit进行编码可以减少最终编码长度。

一个著名的构造最优前缀码的贪心算法就是哈夫曼（Huffman）算法。

PART 02

Huffman算法

哈夫曼算法的基本思想是以频率作为权构建一棵哈夫曼树，然后利用哈夫曼树对字符集进行编码，从而完成最有前缀码的构造。

核心思想是频率（权值）越大的叶子（字符）离根越近。

核心策略是**每次从树的集合中取出没有双亲且权值最小的两棵树作为左右子树**。

具体过程：构造一棵哈夫曼树，是将所要编码的字符作为叶子结点，将频率作为叶子结点的权值，以自底向上的方式，通过n-1次的“合并”运算后构造出的一棵树。

假设有n个字符，则构造出的哈夫曼树有n个叶子结点。n个权值分别设为 w_1、w_2、…、w_n，则哈夫曼树的构造**规则**为：

(1) 将w_1、w_2、…，w_n看成是有n棵树的森林(每棵树仅有一个结点);

(2) 在森林中选出两个根结点的权值最小的树合并，作为一棵新树的左、右子树，且新树的根结点权值为其左、右子树根结点权值之和;

(3)从森林中删除选取的两棵树，并将新树加入森林;

(4)重复(2)、(3)步，直到森林中只剩一棵树为止，该树即为所求得的哈夫曼树。

例如字符和频率如下：

a: 45, *b*: 13; *c*: 12; *d*: 16; *e*: 9; *f*: 5

算法4.4 Huffman(*C*)

输入：$C=\{x_1, x_2, \dots, x_n\}$， $f(x_i)$, $i=1, 2, \dots, n$.

输出：Q //队列

1. $n \leftarrow |C|$;
2. $Q \leftarrow C$; //频率递增队列Q
3. for $i \leftarrow 1$ to $n-1$ do
4. z←Allocate-Node() ; // 生成结点 z
5. z.left←Q中最小元; // 取出Q最小元作z的左儿子
6. z.right←Q中最小元 ; // 再取出当前Q最小元作z的右儿子
7. $f(z) \leftarrow f(x)+f(y)$;
8. Insert(*Q*,*z*); // 将 z 插入Q
9. end
10. return *Q*

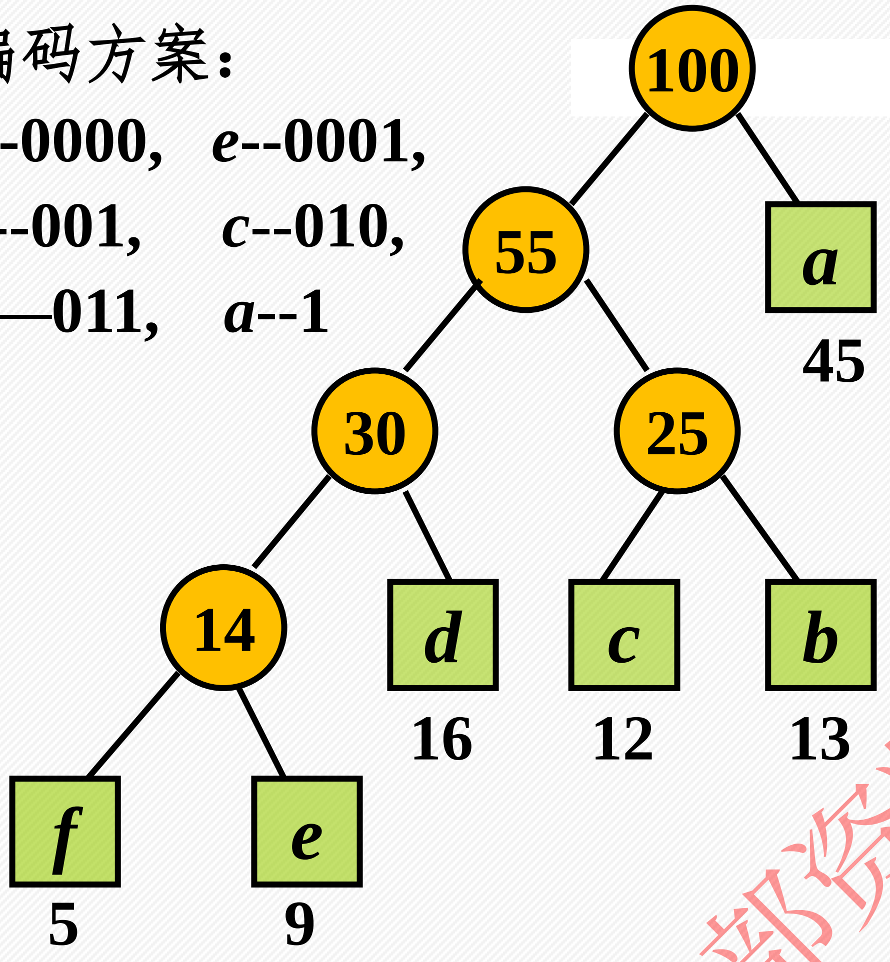
左儿子权值小于等于右儿子

例如字符和频率如下：

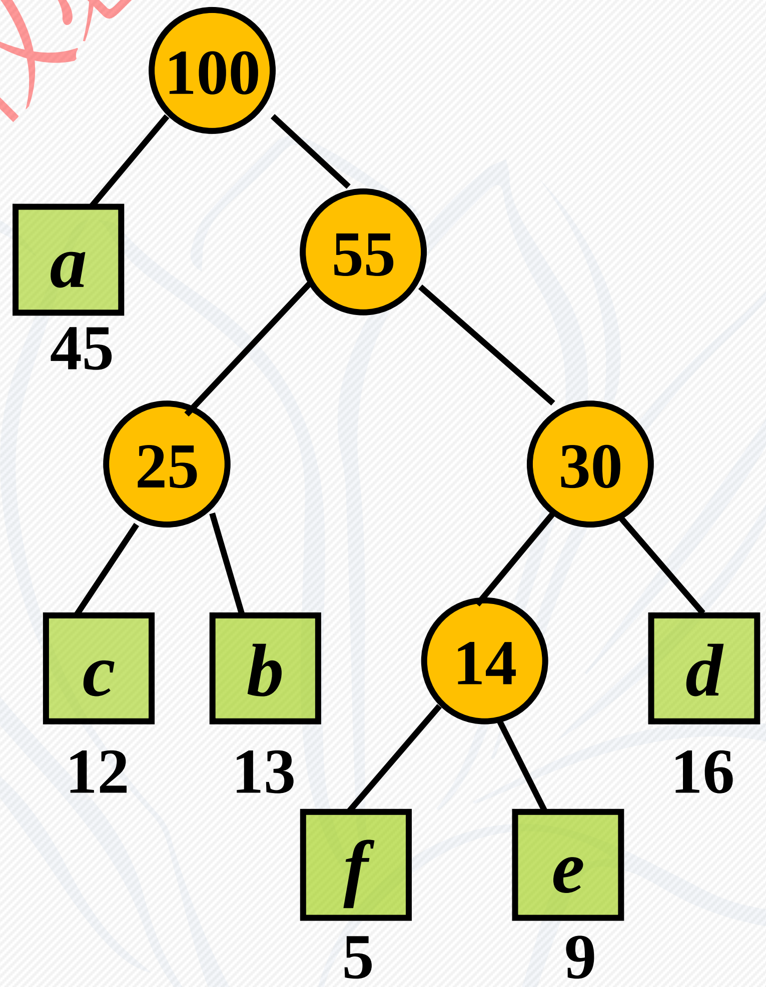
a: 45, *b*: 13; *c*: 12; *d*: 16; *e*: 9; *f*: 5

编码方案：

f--0000, *e*--0001,
d--001, *c*--010,
b—011, *a*--1



左儿子权值小于等于右儿子



编码方案：

f--1100, *e*--1101,
d--111, *c*--100,
b—101, *a*--0

平均位数（树的权）：4 × (0.05 + 0.09)+ 3 × (0.16+0.12+0.13) + 1×0.45 = 2.24

PART 03

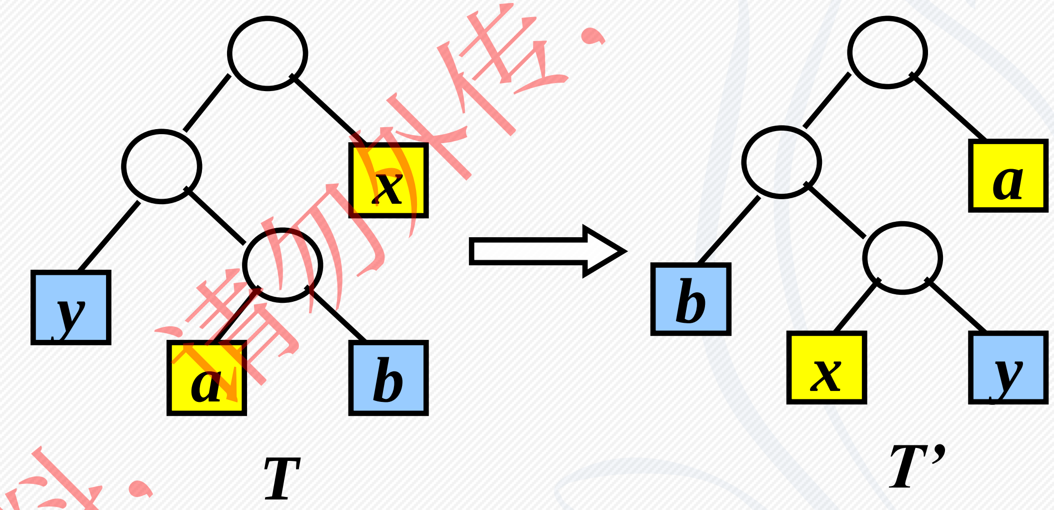
算法正确性证明

引理4.1：设C是字符集，$\forall c \in C$， f(c)为频率，x，$y \in C$，f(x)，f(y)频率最小，那么存在最优二元前缀码使得 x，y 的码字等长，仅在最后一位不同。

$T \to T'$

$f[x] \le f[a]$

$f[y] \le f[b]$



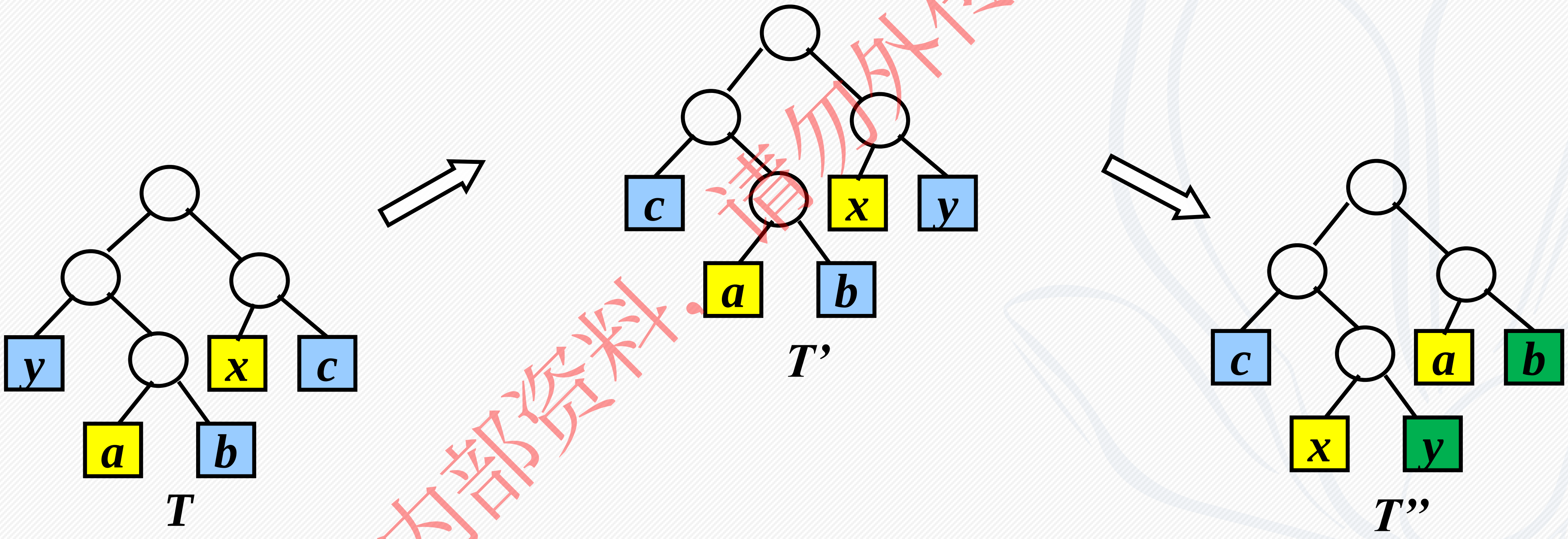
则 T 与 T' 的权之差为

$$B(T) - B(T') = \sum_{i \in C} f[i] d_T(i) - \sum_{i \in C} f[i] d_{T'}(i) \ge 0$$

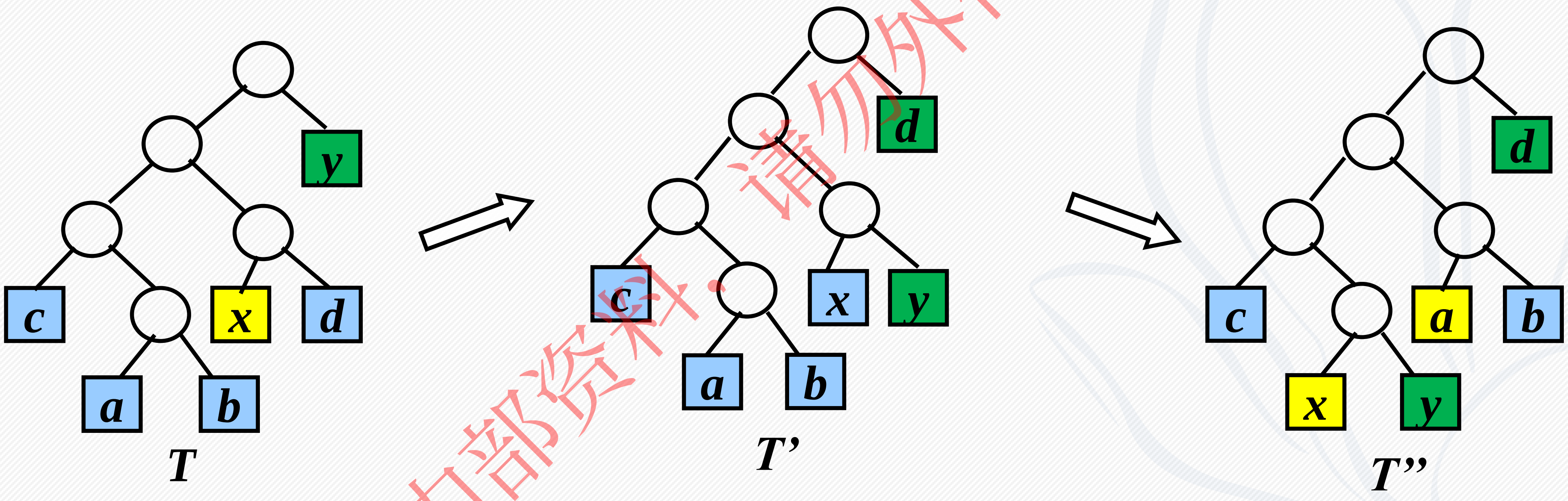
其中 $d_T(i)$为 i 在T 中的层数（i 到根的距离）

正确性证明

引理4.1：设C是字符集，∀c∈C，f(c)为频率，x，y∈C，f(x)，f(y)频率最小，那么存在最优二元前缀码使得 x，y 的码字等长，仅在最后一位不同。



引理4.1：设C是字符集，∀c∈C， f(c)为频率，x，y∈C，f(x)，f(y)频率最小，那么存在最优二元前缀码使得 x，y 的码字等长，仅在最后一位不同，且在最深层。



为什么一定在最深层？

由平均码长计算公式：$B=\sum_{i=1}^{n} f(x_i)d(x_i)$，其中 $d_T(i)$ 为 i 在T 中的层数（i 到根的距离。

上图中，三棵树的平均码长分别为：

$$B(T)=3f(x)+f(y)+4f(a)+4f(b)+3f(c)+3f(d)$$

$$B(T')=3f(x)+3f(y)+4f(a)+4f(b)+3f(c)+f(d)$$

$$B(T'')=4f(x)+4f(y)+3f(a)+3f(b)+3f(c)+f(d)$$

由f(x)，f(y) 频率最小可得：$B(T)-B(T')\geq 0, B(T')-B(T'')\geq 0$

即交换后，整棵树的权降低了。因此，最优解中的x和y一定在最深的叶子上，且互为兄弟。

引理4.3. 设T是二元前缀码所对应的二叉树，$\forall x,y \in T$，x,y是树叶兄弟，z是x和y的父亲，令 $T'=T-\{x, y\}$，且令z的频率$f(z)= f(x)+ f(y)$，T'是对应于二元前缀码$C'=(C-\{x,y\})\cup\{z\}$的二叉树，那么$B(T)= B(T')+f(x)+ f(y)$.

证: 记$d_T(i)$表示结点i在树T中的深度，显然

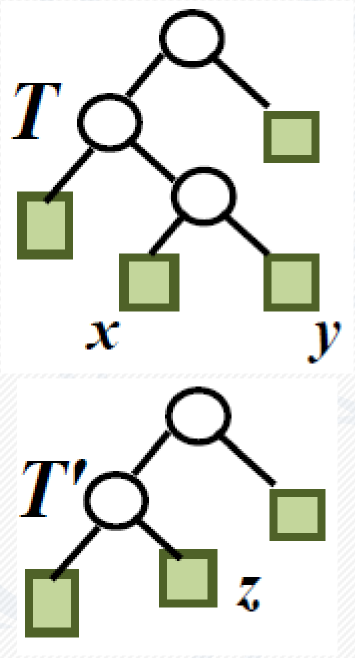
$$d_T(x)=d_T(y)=d_{T'}(z)+1$$

$$\forall c\in C-\{x, y\},\text{有 } d_T(c) = d_{T'}(c) \Rightarrow f(c)d_T(c) = f(c)d_{T'}(c);$$

$$B(T)=\sum_{i\in T} f(i)d_T(i)=\sum_{i\in T,i\neq x,y} f(i)d_T(i)+f(x)d_T(x)+f(y)d_T(y)$$

$$=\sum_{i\in T',i\neq z} f(i)d_{T'}(i)+f(z)d_{T'}(z)+(f(x)+f(y))$$

$$=B(T')+f(x)+f(y)$$



定理4.7(对规模进行归纳)：Huffman 算法对任意规模为 n（n ≥ 2）的字符集C 都得到关于C 的最优前缀码的二叉树.

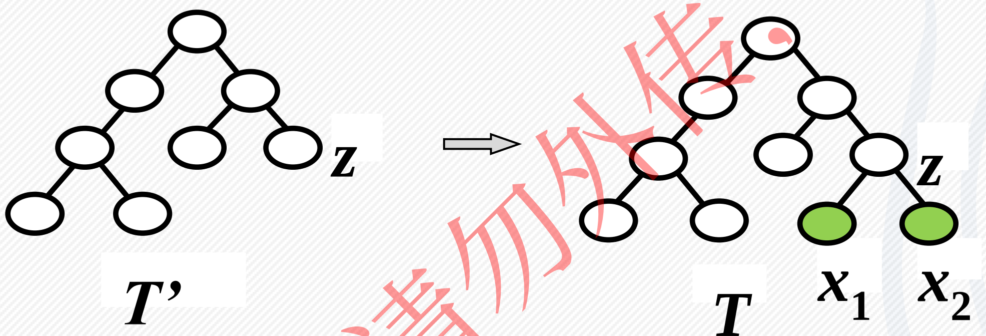
归纳基础：n=2,字符集C= {x_1,x_2}，Huffman 算法得到的代码是0和1，是最优前缀码。
归纳步骤：假设Huffman算法对于规模为k的字符集得到最优前缀码，推导，对于规模为k+1的问题也能得到最优前缀码。

现在考虑规模为k+1的字符集$C=\{x_1,x_2,..,x_{k+1}\}$，其中x_1,x_2 ∈C是频率最小的两个字符.令$C'=(C-\{x_1,x_2\})\cup\{z\}, f(z)=f(x)+f(y)$ 即将x和y捏合成z

根据归纳假设，Huffman算法得到一颗关于字符集C’、频率 f(z)和f(x_i)(i = 3,4,...,k＋1)的最优前缀码的二叉树T’

正确性证明

根据归纳假设：Huffman算法得到关于字符集C’的最优前缀码的二叉树T’。把x_1和x_2,作为z的儿子加入到树T'，得到树T，那么T是关于字符集C的最优前缀码的二叉树。



反证：

如果T不是关于C的最优前缀码二叉树，那么说明一定存在其他最优解，记为T*。由引理1知，x_1和x_2一定在树的最深层树叶。而且由于T*是最优解，那么B(T*)<B(T)。现从T*中去掉x_1,x_2，得到新树T*’，T*’满足：

$$B(T^{*\prime}) = B(T^*) - (f(x1)+f(x2)) < B(T) - (f(x1)+f(x2)) = B(T')。$$

这与T’是C’的最优解是矛盾的，因此T就是的最优解。

贪心算法回顾

1、贪心法适合于解决组合优化问题，求解过程是多步判断过程，最终的判断序列对应于问题的最优解

2、判断依据某种“短视的”贪心选择性质，这种短视的就是只看眼前。性质的好坏决定了算法的正确性，贪心性质的选择往往依赖于直觉或者经验。

3、要想说明一个贪心策略是否正确，往往是需要证明的，**所以贪心算法必须进行正确性证明**。那么对于贪心法的正确性证明方法有数学归纳法和交换论证等；**证明贪心策略不对：举反例**

4、贪心法的优势：算法简单，时间和空间复杂性低