得不到最优解的处理方法

- 讨论对于哪些输入贪心选择能够得到最优解
  - 输入应该满足的条件
- 讨论贪心法的解最坏情况下与最优解的误差
  - 绝对误差与相对误差估计

找零钱问题

问题描述：

设有 n 种零钱，

重量分别为：$w_1, w_2, \ldots, w_n$,

价值分别为：$v_1=1, v_2, \ldots, v_n$.

付的总钱数是：y

问：如何付钱使得所付钱的总重最轻?

$$\min\{w_1x_1 + w_2x_2 + \ldots + w_nx_n\}$$

$$v_1x_1 + v_2x_2 + \ldots + v_nx_n = y$$

$$x_i \in \mathrm{N}, \quad i = 1,2,\ldots,n$$

动态规划算法

属于整数规划问题，动态规划算法可以得到最优解

设 $F_k(y)$ 表示用前 k 种零钱，总钱数为 y 的最小重量

递推方程

$$F_{k+1}(y) = \min_{0 \le x_{k+1} \le \left\lfloor \frac{y}{v_{k+1}} \right\rfloor} \{F_k(y - v_{k+1}x_{k+1}) + w_{k+1}x_{k+1}\}$$

$$F_1(y) = w_1 \left\lfloor \frac{y}{v_1} \right\rfloor = w_1 y$$

Greedy算法

假设

$$\frac{w_1}{v_1} \geq \frac{w_2}{v_2} \geq \ldots \geq \frac{w_n}{v_n}$$

使用前 k 种零钱，总钱数为 y

贪心法的总重为 $G_k(y)$，则有如下递推方程

$$G_{k+1}(y) = w_{k+1}\left\lfloor \frac{y}{v_{k+1}} \right\rfloor + G_k(y \bmod v_{k+1})$$

$$G_1(y) = w_1\left\lfloor \frac{y}{v_1} \right\rfloor = w_1 y$$

n=1, 2 时得到最优解

$n=1$ 只有一种零钱，$F_1(y)=G_1(y)$， $F_2(y)=G_2(y)$

$n=2$，使用价值大的钱越多，得到的解越好

$$
\begin{aligned}
&[F_1(y-v_2(x_2+\delta))+w_2(x_2+\delta)] \\
&-[F_1(y-v_2x_2)+w_2x_2] \\
=&[w_1(y-v_2x_2-v_2\delta)+w_2x_2+w_2\delta] \\
&-[w_1(y-v_2x_2)+w_2x_2] \\
=&-w_1v_2\delta+w_2\delta=\delta(-w_1v_2+w_2)\le 0
\end{aligned}
$$

$n>2$时得到最优解的判定条件

定理2 假定 $G_k(y)=F_k(y)$,

$v_{k+1}>v_k$ 且 $v_{k+1}=pv_k-\delta$, $0\le\delta<v_k$, $p\in Z^+$,

则以下命题等价.

$$(1)\ G_{k+1}(y)\le G_k(y)$$

$$(2)\ G_{k+1}(y)=F_{k+1}(y)$$

$$(3)\ G_{k+1}(pv_k)=F_{k+1}(pv_k)$$

$$(4)\ w_{k+1}+G_k(\delta)\le pw_k$$

用条件(4)需 $O(k)$ 时间验证 $G_{k+1}(y)=F_{k+1}(y)$?

对 n 种零钱作出验证, 可在 $O(n^2)$ 时间内完成

实例

$$v_{k+1} = pv_k - \delta, \quad 0 \le \delta < v_k, \quad p \in Z^+$$
$$w_{k+1} + G_k(\delta) \le pw_k$$

例　$v_1=1, v_2=5, v_3=14, v_4=18, w_i=1, i=1, 2, 3, 4.$

对一切 y 有 $G_1(y)=F_1(y), G_2(y)=F_2(y)$.

验证　$G_3(y) = F_3(y)$

$v_3=pv_2-\delta \Rightarrow 14=5p-\delta \Rightarrow p=3, \delta=1.$

$w_3+G_2(\delta)=1+ G_2(1) =1+1 = 2$

$pw_2=3\times 1=3$

$w_3+G_2(\delta)\le p\, w_2$

实例

$$v_{k+1} = pv_k - \delta, \quad 0 \le \delta < v_k, \quad p \in Z^+$$
$$w_{k+1} + G_k(\delta) \le pw_k$$

$v_1=1$, $v_2=5$, $v_3=14$, $v_4=18$, $w_i=1$, $i=1, 2, 3, 4$.

$v_4= pv_3-\delta \Rightarrow 18=14p-\delta \Rightarrow p=2, \delta=10$

$w_4+G_3(\delta)=1+G_3(10)=1+2=3$

$pw_3=2\times 1=2$

$w_4+G_3(\delta) > pw_3$, $G_4(y)$不是最优解.

$G_4(pv_3) > F_4(pv_3)$. 即

$G_4(28)=\lfloor 28/18 \rfloor + \lfloor 10/5 \rfloor = 1+2 = 3$

$F_4(28)=28/14 = 2$.

应用：最优前缀码

前缀码：用0-1字符串作为代码表示字符，要求任何字符的代码都不能作为其它字符代码的前缀

非前缀码　*a*---001, *b*---00, *c*---010, *d*---01

实例 0100001:　解码1. 01, 00, 001　*d*, *b*, *a*

解码2. 010, 00, 01　*c*, *b*, *d*

前缀码的存储采用二叉树的结构，每个字符作为树叶，每个前缀码看作根到树叶的路径

输入：n个字符 $x_1, x_2, \ldots, x_n$,

每个字符传输概率 $f(x_i)$, $i=1, 2, \ldots, n$.

求：前缀码，使得平均传输一个字符的位数达到最小

算法：Huffman树得到最优解

Huffman算法

算法 Huffman(C)

1. $n \leftarrow |C|$;

2. $Q \leftarrow C$; // 按频率递增构成队列Q

3. for $i \leftarrow 1$ to $n-1$ do

4. $z \leftarrow$Allocate-Node() // 生成结点 z

5. $z.left \leftarrow Q$中最小元 // 取出Q中最小元作为z的左儿子

6. $z.right \leftarrow Q$中最小元 // 取出Q中最小元作为z的右儿子

7. $f[z] \leftarrow f[x]+f[y]$

8. Insert(Q,z); // 将 z 插入Q , $O(\log n)$

9. Return Q

时间：$O(n\log n)$

实例

例如 ***a*:45, *b*:13; *c*:12;**

***d*:16; *e*:9; *f*:5**

编码:

***f*--0000, *e*--0001,**

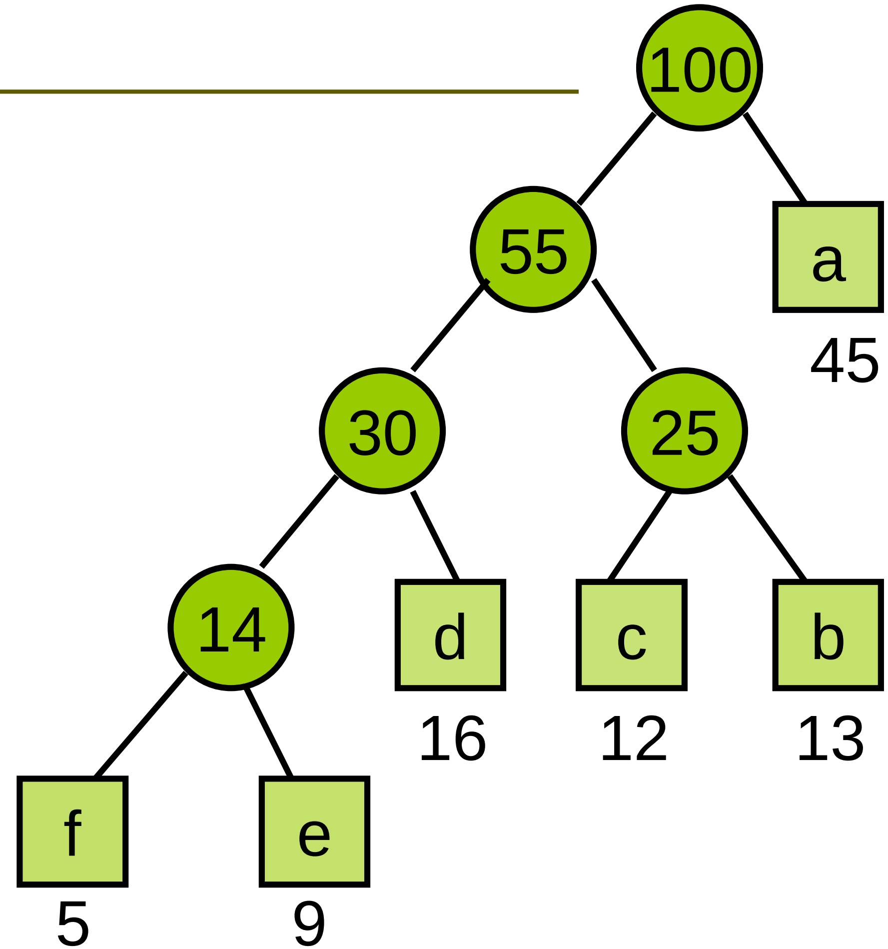
***d*--001, *c*--010,**

***b*—011, *a*--1**

平均位数:

4*(0.05+0.09)

+3*(0.16+0.12+0.13)+1*0.45= 2.24



证明：引理1

引理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，且仅在最后一位不同.

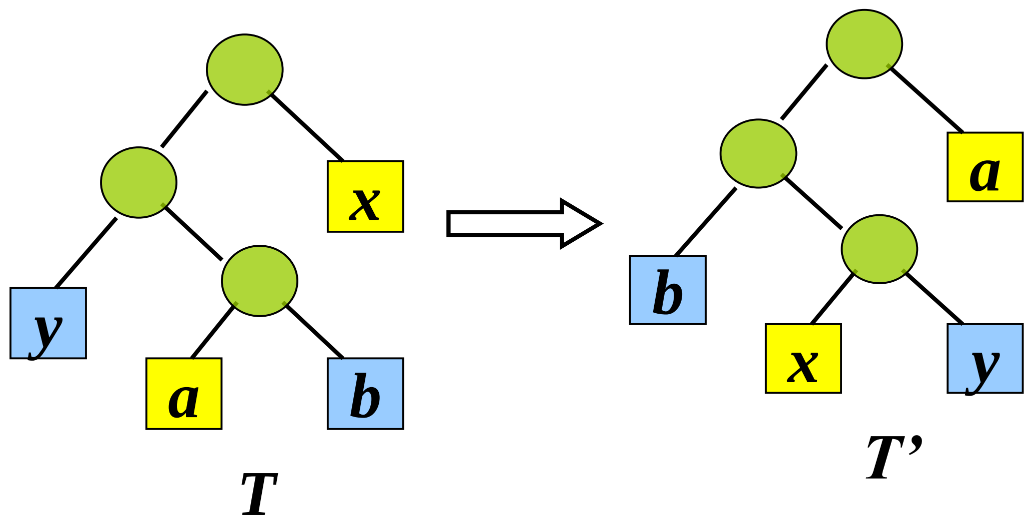
$T \rightarrow T'$

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

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

a与 x 交换

b与 y 交换



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）

引理2

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

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

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

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

证明：归纳法

定理 Haffman 算法对任意规模为n（$n \geq 2$）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 的字符集都得到最优前缀码. 考虑规模为$k+1$的字符集$C=\{x_1, x_2, \ldots, x_{k+1}\}$, 其中$x_1, x_2 \in C$是频率最小的两个字符. 令

$$C' = (C - \{x_1,x_2\}) \cup \{z\}, \quad f(z)=f(x)+f(y)$$

根据归纳假设，Huffman算法得到一棵关于字符集C'、频率$f(z)$和$f(x_i)$（$i=3,4,\ldots,k+1$）的最优前缀码的二叉树T'.

证明：归纳法(续)

把x_1和 x_2作为 z 的儿子附加到T'上，得到树T，那么T是关于字符集$C=(C'-\{z\})\cup\{x_1,x_2\}$ 的最优前缀码的二叉树.

如若不然，假如T 不是关于C 的最优二元前缀码对应的二叉树，那么存在更优的树. 根据引理1，存在最优树T^*，其最深层树叶是 x_1,x_2，且 $B(T^*)< B(T)$.

去掉T^*中的x_1和x_2，根据引理2，所得二叉树$T^{*\prime}$满足

$$B(T^{*\prime})=B(T^*)-(f(x)+f(y))<B(T)-(f(x)+f(y)) =B(T')$$

与T'是一棵关于C'的最优前缀码的二叉树矛盾.

文件归并

问题：给定一组不同长度的排好序文件构成的集合

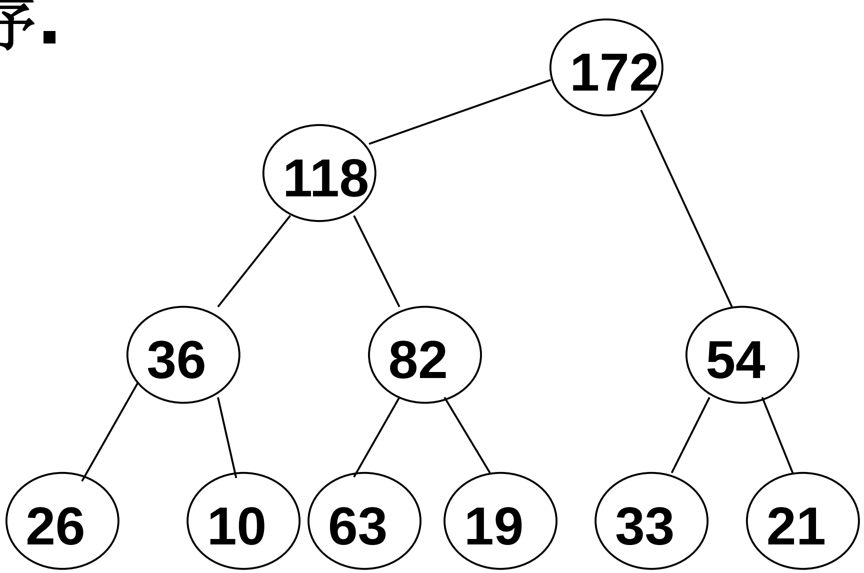
$$S = \{ f_1, \ldots , f_n\}.$$

其中 f_i 表示第 i 个文件含有的项数.

使用二分归并将这些文件归并成一个有序的文件. 找到一个比较次数最少的归并次序.

归并过程对应于二叉树

实例： 26,10,63,19,33,21



归并的代价

归并树叶 f_i 和 f_j, 代价是 $|f_i| + |f_j|$.

$C(T)$ 是树的内结点的权之和

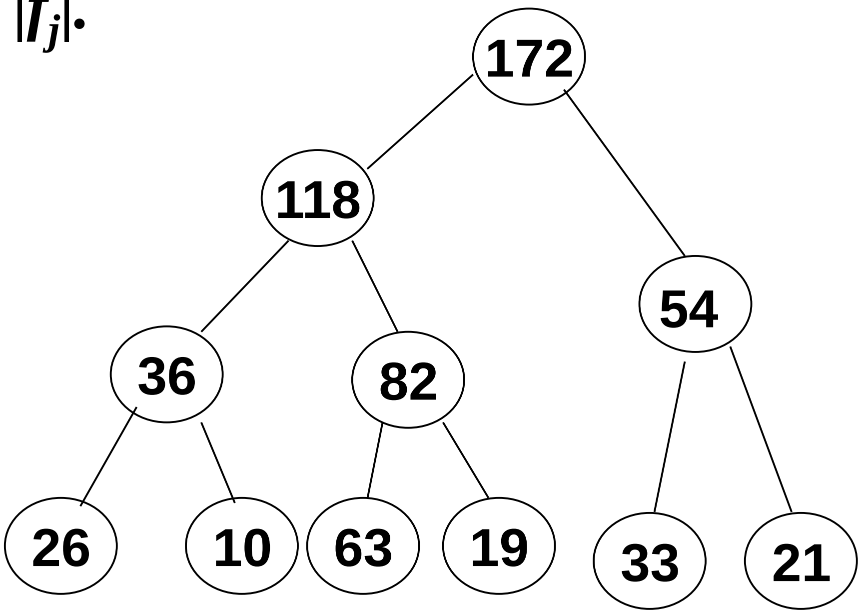
实例:

$C(T)$ = 36+82+54+118+172

= (26+10+63+19)×3

+ (33+21)×2

= 462

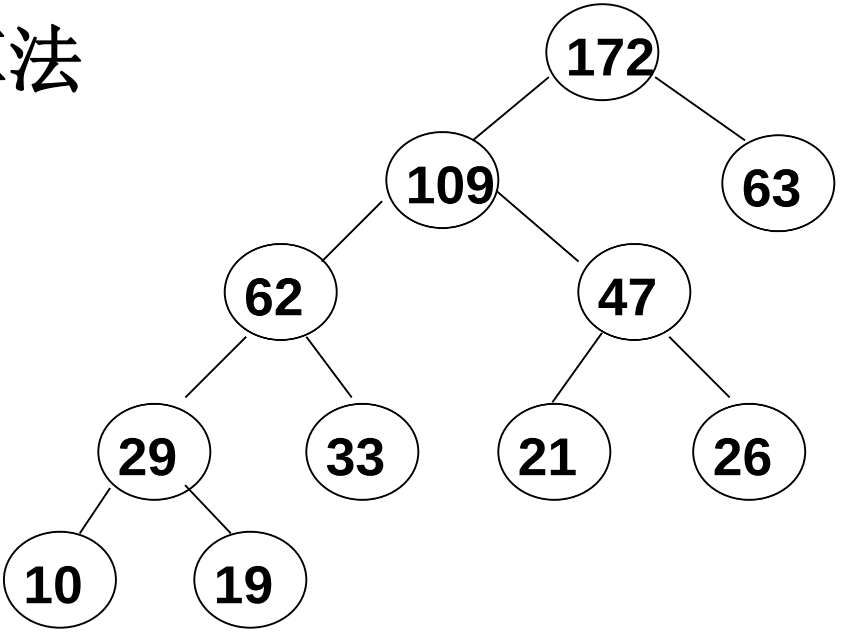


$$\sum_{k=1}^{n} | f_k | \times depth(f_k) - (n-1)$$

更好的归并方法

Huffman树的算法

时间$O(n\log n)$



$(10+19)\times 4+(33+21+26)\times 3+63$

$=116+240+63=419$