算法设计与分析

第 4 章 贪心算法

金英　编写

计算机科学技术学院
软件学院

2021 年 3 月

目　录

第 4 章 贪心算法（Greedy Algorithms）

4.1 贪心法基本思想

求解最优化问题的贪心算法包含一系列步骤。每一步都在一组选择中做出在当前看来最好的选择，希望通过做出局部优化选择达到全局优化选择。但贪心算法不一定总产生优化解，所以一个贪心算法是否产生优化解，需要严格证明。

例 1：背包问题。

给定 n 种物品和一个背包。物品 i 的重量是 w_i，其价值为 v_i，背包的容量为 c。问应如何选择物品装入背包，使得装入背包中的物品的总价值最大？在选择物品 i 装入背包时，可以选择物品 i 的一部分，而不一定要全部。

用贪心算法，每次选单位重量价值 v_i / w_i 最大的物品 i 放进包中。

背包问题具有最优子结构性质，它可以用动态规划算法来解。但用贪心算法更简单，解题效率更高。

贪心算法总是做出在当前看来是最好的选择。也就是说贪心算法并不从整体最优上加以考虑，所以贪心算法不是对所有问题都能得到整体最优解。

例 2：0-1 背包问题。

给定 n 种物品和一个背包。物品 i 的重量是 w_i，其价值为 v_i，背包的容量为 c。问应如何选择物品装入背包，使得装入背包中的物品的总价值最大？在选择物品装入背包时，对每种物品 i 只有两种选择，要么装入，要么不装入，不能将物品 i 装入背包多次，也不能只装入物品 i 的一部分。因此，该问题称为 0-1 背包问题。设有 3 物品，这 3 个物品的重量为 W={3,5,7}，价值为 V={9,10,12}，背包容量 c=10。用贪心法会将物品 1 和 2 装包，获得的价值为 19，但最优解是将物品 1 和 3 装包，获得的价值为 21。

0-1 背包问题也具有最优子结构性质。但是 0-1 背包问题却不能用贪心算法求解。

对于 0-1 背包问题，贪心选择之所以不能得到最优解是因为在这种情况下，它无法保证最终将背包装满，背包部分空间的闲置使每公斤背包空间所具有的价值降低了 (该问题不具有贪心选择性) 。事实上，在考虑 0-1 背包问题的物品选择时，应比较选择该物品和不选择该物品所导致的最终结果，然后再做出最好的选择。

贪心法通常用于求一个问题在某种意义下的最优解。适合采用贪心法的优化问题必

须具备最优子结构性质和贪心(Greedy)选择性。

当一个问题的优化解包含了子问题的优化解时，则称该问题具有优化子结构性质。若一个优化问题的全局优化解可以通过局部优化选择得到，则该问题称为具有 Greedy 选择性质。

如前所述，贪心算法不是对所有问题都能得到整体最优解。所以贪心算法是否产生优化解，需要严格证明。贪心算法正确性证明方法如下：

①证明算法所求解的问题具有优化子结构；

②证明算法所求解的问题具有 Greedy 选择性；

③算法是按②中的 Greedy 选择性进行局部优化选择的。

虽然贪心算法不是对所有问题都能得到整体最优解，但对范围相当广的许多问题能产生整体最优解，如单源最短路经问题，最小生成树问题，哈夫曼编码问题等。

在一些情况下，即使贪心算法不能得到整体最优解，但其最终结果却是最优解的很好的近似解。

4.2 活动安排问题（An activity-selection problem）

活动安排（或选择、调度）问题是可以用贪心算法有效求解的一个很好的例子。

设有 n 个活动的集合 E＝｛1，2，…，n｝，其中每个活动都要求以独占的方式使用同一资源，如演讲会场等，而在同一时间内只允许一个活动使用这一资源。每个活动 i 都有一个要求使用该资源的起始时间 s_i 和一个结束时间 f_i，且 $s_i < f_i$。如果选择了活动 i，则它在半开的时间区间$[s_i，f_i)$内占用资源。

若区间$[s_i，f_i)$与区间$[s_j，f_j)$不相交，则称活动 i 与活动 j 是相容的。也就是说，当 $s_i \geq f_j$ 或 $s_j \geq f_i$ 时，活动 i 与活动 j 是相容。

活动安排问题是要在所给的活动集合中选出最大的（两两）相容活动子集合。

⑴ **算法设计**

贪心思想：为了选择最多的相容活动，每次选 f_i 最小的活动，使剩余的可安排时间段极大化，以便接待尽可能多的相容活动。

例： 设活动集 S={1，2，…，11}中的活动已按结束时间非减序排列，如表 4-1 所示。执行过程如下：

第 1 步，选活动 1；

第 2 步，在剩余相容活动子集{4,6,7,8,9,11}中选结束时间最早的活动 4；

表 4-1　待安排活动集 S

活动 i	1	2	3	4	5	6	7	8	9	10	11
开始时间 s_i	1	3	0	5	3	5	6	8	8	2	12
结束时间 f_i	4	5	6	7	8	9	10	11	12	13	14

第 3 步，在剩余相容活动子集{8,9,11}中选结束时间最早的活动 8；

第 4 步，在剩余相容活动子集{11}中选结束时间最早的活动 11。

这时，剩余相容活动子集{}变为空集，从而得 S 的一个最优解 A={1,4,8,11}。

活动安排问题的贪心算法如下：

输入：各活动的起始时间和结束时间存储于数组 s[n]和 f[n]中，且按结束时间的非减序：$f_1 \le f_2 \le \ldots \le f_n$ 排列。

输出：最大相容活动子集 A。

```
void GreedySelector (int n, Type s[ ], Type f[ ], bool A[ ])
{     A[1] = true;    //选择活动 1
      int    j=1;       // j 用以记录最近一次加入到 A 中的活动
      for (int i=2; i<=n; i++)
          if (s[i] >= f[j])
              { A[i]=true;
                j=i;
              }
          else    A[i]= false;
}
```

⑵ **算法复杂性分析**

当活动按结束时间已排序，$T(n)=\Theta(n)$。

当活动按结束时间未排序，$T(n)=\Theta(n)+\Theta(n\log n)=\Theta(n\log n)$。

⑶ **算法正确性证明**

① 证明问题具有优化子结构

引理 1 设 S={1, 2, …, n}是 n 个活动的集合，s_j, f_j 是活动 j($1 \le j \le n$)的起（始截）止时间，且 $f_1 \le f_2 \le \ldots \le f_n$，则 S 的活动选择问题的某个优化解包括活动 1。

证：设 A 是一个优化解，按结束时间排序 A 中活动，设其第一个活动为 k，第二个

活动为 j。

如果 k=1，引理成立。

如果 k≠1(即 k>1)，令 B=A-{k}∪{1}，

由于 A 中的活动相容， $f_1 \le f_k \le s_j$，B 中的活动相容。

因为|B|=|A|，所以 B 是一个优化解，且包含活动 1，即 B 是一个以贪心选择活动 1 开始的最优活动安排。

引理 1 说明了总存在一个以贪心选择开始的最优活动安排方案。

定理 1 设 S={1, 2, …, n}是 n 个活动的集合，s_j, f_j 是活动 j($1 \le j \le n$)的起止时间，且 $f_1 \le f_2 \le \ldots \le f_n$。设 A 是 S 的调度问题的一个优化解，且包括活动 1，则 A2=A-{1}是 S2={ j∈S | $s_j \ge f_1$ }的调度问题的优化解。

证：显然，A2 中的活动是相容的。

我们仅需要证明 A2 是最大的。

若不然，存在一个 S2 的活动选择问题的优化解 B2，|B2|>|A2|。

令 B={1}∪B2，对于 j∈S2，$s_j \ge f_1$，B 中活动相容。B 是 S 的一个解。

由于|A|=|A2|+1，|B|=|B2|+1 > |A2|+1=|A|，与|A|最大矛盾。

定理 1 说明活动选择问题具有最优子结构性质（在选择了活动 1 后，原问题就简化为对 S 中所有与活动 1 相容的活动 S2={ j∈S | $s_j \ge f_1$ }进行活动安排的子问题。A 是原问题的一个最优解，A2=A-{1}是活动 S2 安排子问题的一个最优解）。

② 证明问题具有贪心选择性

定理 2 设 S={1, 2, …, n}是 n 个活动的集合，s_j, f_j 是活动 j($1 \le j \le n$)的起止时间，且 $f_1 \le f_2 \le \ldots \le f_n$。$l_i$ 是 S_i ={ j∈S | $s_j \ge f_{l_{i-1}}$ }中具有最小结束时间 f_{l_i} 的活动(设 l_0=0,f_0=0)。设 A 是 S 的包含活动 1 的优化解，则 $A = \bigcup_{i=1}^{k} \{l_i\}$。

证：对|A|作归纳法（即对贪心选择的次数作归纳法）。

当|A|=1 时，由引理 1，命题成立；

设|A|<k 时，命题成立；

当|A|=k 时，A={1}∪A2，由定理 1，A2 是 S 2 ={ j∈S | $s_j \ge f_1$ }的优化解。由归纳假设，$A2 = \bigcup_{i=2}^{k} \{l_i\}$。于是，$A = \bigcup_{i=1}^{k} \{l_i\}$。

定理 2 说明活动选择问题具有贪心选择性。

③ 活动安排问题的贪心算法 GreedySelector 是按选择性进行局部优化选择的。

因此，GreedySelector 算法对于活动安排问题来说，总能求得整体最优解 (即相容活动集合 A 的规模最大)。

4.3 最优装载问题

有一批集装箱要装上一艘载重量为 c 的轮船。其中，集装箱 i 的重量为 w_i。最优装载问题要求确定，在装载体积不受限制的情况下，应如何装载才能将尽可能多的集装箱装上轮船。

设 n 是集装箱总数。该问题可形式化地描述为: 求一个 n 维的 0-1 向量 $(x_1, x_2, ..., x_n)$，$(x_i \in \{0,1\}, i = 1,2,...,n)$，在满足 $\sum_{i=1}^{n} w_i x_i \le c$ 时，使 $\sum_{i=1}^{n} x_i$ 最大。其中，

$$x_i = \begin{cases} 0 & \text{表示不装入集装箱i} \\ 1 & \text{表示装入集装箱i} \end{cases}$$

⑴ **算法设计**

贪心思想：用重量最轻者先装的贪心选择策略。由此可产生装载问题的最优解。

算法输入：n 个集装箱的重量，轮船的载重量 c。

算法输出： 装进轮船的集装箱编号（最多的集装箱）。

假设集装箱已按重量非减的次序排序。最优装载问题的贪心算法如下。

```
void Loading(int x[ ], Type w[ ], Type c, int n)
{     for ( int i=1;i<=n; i++)
            x[i]=0;   // 数组元素 x[i]=0 表示不装入集装箱 i。
       for ( int i=1; i<=n &&w[i]<= c; i++)
            { x[i] =1;
              c- = w[i];
            }
}
```

⑵ **算法复杂性分析**

当集装箱已依其重量从小到大排序，T(n)= Θ (n)。

当集装箱未排序，T(n)= Θ (n)+ Θ (nlogn)= Θ (nlogn)。

⑶ **算法正确性证明**

首先证明最优装载问题具有贪心选择性质：

设集装箱已依其重量从小到大排序，$(x_1, x_2, \ldots, x_n)$是最优装载问题的一个最优解。又设 $k = \min_{1 \le i \le n}\{i \mid x_i = 1\}$。易知，$1 \le k \le n$。

①当 k=1 时，$(x_1, x_2, \ldots, x_n)$是一个以贪心选择开始的最优解。

②当 k>1 时，取 $y_1 = 1; y_k = 0; y_i = x_i, 1 < i \le n, i \ne k$，则

$$\sum_{i=1}^{n} w_i y_i = w_1 - w_k + \sum_{i=1}^{n} w_i x_i \le \sum_{i=1}^{n} w_i x_i \le C$$

因此，$(y_1, y_2, \ldots, y_n)$是最优装载问题的一个可行解。

另一方面，又 $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} x_i$ 知，$(y_1, y_2, \ldots, y_n)$是一个以贪心选择开始的最优解。

所以，最优装载问题具有贪心选则性质。

然后证明最优装载问题具有最优子结构性质：

设 $(x_1, x_2, \ldots, x_n)$是最优装载问题的一个满足贪心选择性质的最优解，则 x_1=1，$(x_2, x_3, \ldots, x_n)$是轮船载重量为 C-$w1$ 且待装船集装箱为$\{2, 3, \ldots, n\}$时相应最优装载问题的一个最优解（如若不然可以导出一个与$(x_1, x_2, \ldots, x_n)$是最优解相矛盾的事实）。所以，最优装载问题具有最优子结构性质。

最后，最优装载问题的贪心算法 Loading 是按选择性进行局部优化选择的。

因此，Loading 算法对于最优装载问题来说，总能求得整体最优解 (即装上轮船的集装箱数目最大)。

注：关于贪心法的正确性证明的补充说明

贪心法的“短视的”贪心策略有时候只能导致局部最优，而不是全局最优。因此，怎样选择合适的贪心策略并证明该策略的正确性就成了算法设计的关键。

贪心法的正确性证明常见有两种证明方法，一是用数学归纳法，二是使用交换论证的方法。所谓交换论证的思想就是：从任意一个最优解出发，经过不断用新的成分替换解中的原有成分来改变这个解。用交换论证的方法证明贪心法正确性的具体例子请参看相应的参考书。下面再详细介绍用数学归纳证明贪心法正确性的方法。

用数学归纳证明贪心法正确性，可以通过对算法步数的归纳来证明贪心法的正确性，也可以通过对问题规模的归纳来证明贪心法的正确性。

但是，在使用归纳法之前需要叙述一个相关的命题。如果对算法步数归纳，命题的主要内容是：对于任何正整数 k，贪心法的前 k 步都导致最优解。如果对问题规模归纳，命题的主要内容是：对于任何正整数 k，贪心法对于规模为 k 的实例都得到最优解。

为了使用对实例规模的归纳证明求解最优装载问题贪心算法 Loading 的正确性，需要先叙述的可用归纳证明的命题为：

定理 3 对于任何正整数 k，算法 Loading 都对 k 个集装箱的实例得到最优解。

证明 k=1，只有 1 个集装箱，其重量 $w_1 \le c$，任何算法都只有一种装法，就是将这个集装箱装上船。算法 Loading 得到最优解。

假设算法 Loading 对于规模为 k 的输入都能得到最优解，考虑规模为 k+1 的输入 N={1, 2, …, k, k+1}，W={w_1, w_2, …, w_k, w_{k+1}}是集装箱重量，其中 $w_1 \le w_2 \le ... \le w_k \le w_{k+1}$。从 N 中拿掉最轻的集装箱 1，得到：

$$N'=N-\{1\}=\{2, 3, \ldots, k, k+1\}$$

$$W'=W-\{w_1\}=\{w_2, w_3, \ldots, w_k, w_{k+1}\}$$

$$c'=c-w_1$$

根据归纳假设，对于 N'，W'和 c'，算法 Loading 得到最优解 I'。令

$$I = I' \bigcup \{1\}$$

那么 I 是 N 的最优解。这也恰好是算法 Loading 对于 N，W 和 c 的解。

如若不然，存在包含 1 的关于 N 的最优解 I^*（如果 I^*中没有 1，用 1 替换 I^*中的第一个集装箱得到的解也是最优解），且$|I^*|>|I|$，那么 $I^*-\{1\}$是关于 N'，W'和 c'的解，且

$$|I^*-\{1\}|>|I-\{1\}|=|I'|$$

与 I'的最优性矛盾。

4.4 哈夫曼编码问题

哈夫曼编码是广泛应用于数据文件压缩的十分有效的编码方法，其压缩率通常在 20% ~ 90%之间。哈夫曼编码使用字符在文件中出现的频率为字符建立一个用 0, 1 串表示的编码。哈夫曼编码是不等长编码，在文件中出现频率越高的字符其编码长度越短，

它是一种最优编码方案。

1. 不等长前缀（编）码

⑴ 前缀码

文件中字符用二进制进行编码，即每个字符用一个二进制 0, 1 串来表示。字符编码可分为：等长编码和不等长编码。

等长编码：所有字符的编码都用等长的 0, 1 串来表示。

可变长编码：经常出现的字符用短编码，不经常出现的用长编码。

前缀编码：无任何字符的编码是另一个字符编码的前缀。

不等长编码应该是前缀编码，不然接收方译码时会产生歧义。例如：假设在电文中出现的字符集为{a, b, c, d}，采用不等长编码方案：

a 的编码为：0

b 的编码为：01

c 的编码为：00

d 的编码为：1

发送方：

想发送的电文原文是：abcd。

编码后发送给接收方的 0, 1 序列是：001001。

接收方：

接收方接收“001001”后，译码时可以有多种理解：aadaad、abcd、aadab 等。

接收方译码时之所以会产生歧义，是因为本例给出的是不等长编码，且不是前缀编码，如，字符 a 的编码是字符 b 编码的前缀，字符 a 的编码也是字符 c 编码的前缀。

⑵ 不等长前缀码的二叉树表示

译码过程需要方便地取出编码的前缀，为此可以选择二叉树作为表示前缀编码的数据结构，如图 4-1 所示。其中，叶结点用字符及其出现频率（或频数）标记；内结点用其子树叶子的频率（或频数）和标记。此外，一个结点与其左子女的边（左分支）标记为 0 ，结点与其右子女的边（右分支）标记为 1 。每个字符的编码是从根到对应的叶子路径上的 0, 1 串。

给定编码字符集 C 及其概率分布 f，即 C 中任意字符 c 以概率 f(c)在数据文件中出现。C 的一个前缀编码方案对应一棵二叉树 T。字符 c 在树 T 中的深度记为 $d_T(c)$。$d_T(c)$

也是字符 c 的前缀码长。该编码方案的平均编码长度定义为：$B(T)=\sum_{c\in C} f(c)d_T(c)$。

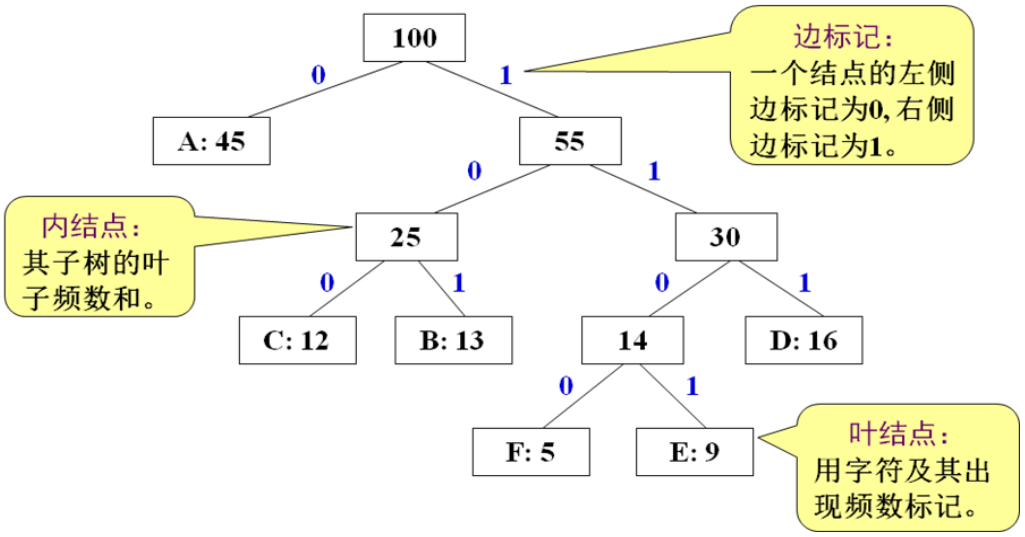

图 4-1 前缀码的二叉树表示

使平均编码长度达到最小的前缀码称为最优前缀码。

2. 构造哈夫曼编码

哈夫曼提出了构造最优前缀码的贪心算法，由此产生的编码方案称为**哈夫曼编码**。

⑴ 构造最优前缀码的哈夫曼算法

构造哈夫曼树是构造哈夫曼编码的基础。哈夫曼树是指叶结点的加权路径长度之和最小的二叉树。例如在图 4-1 中的叶子 C，从根到 C 的路径长度为 3（经历的边数），其权值为 12，所以其加权路径长度为 36（即 3 与 12 的乘积）。

基本思想： 循环地选择根结点频率最低的两棵二叉树，合并成一棵二叉树，直到形成一棵二叉树。

算法输入：字符表 C={$c_1, c_2, \ldots, c_n$}，及每个字符的频率 $f(c_i)(1\le i\le n)$。

算法输出：Huffman 树。

算法描述：

```
HUFFMAN(C)              /*  使用堆操作实现  */
{ n=|C|;
  Q=Initialize（C）;    /*  初始建堆，优先队列用堆来实现  */
   for(i=1; i<n; i++)
   {   z=malloc(sizeof(node));
       x=DeleteMin(Q);        /*  堆操作  */
```

```
        y= DeleteMin(Q);        /* 堆操作 */
        z->lchild=x;
        z->rchild=y;
        z->f=x->f + y->f;
        Insert(z, Q);           /* 堆操作 */
    }
    return(DeleteMin(Q));
}
```

⑵ **算法复杂性分析**

设优先队列 Q 由一个堆实现。

第二步使用堆排序的初始建堆算法实现，需要的时间为 O(n)；

每个堆操作要求 O(log$n$),循环 n-1 次，需要 O(nlog$n$)；

T(n)= O(n)+ O(nlog$n$) = O(nlog$n$)。

⑶ **哈夫曼算法的正确性证明**

① 证明问题具有最优子结构性质

定理 4（最优子结构） 设 T 是字母表 C 的哈夫曼树，$\forall c \in C$，$f(c)$是 c 在文件中出现的概率。设 x，y 是 T 中任意两个相邻叶结点，z 是它们的父结点，则 z 作为概率为 $f(z)=f(x)+f(y)$的字符，$T_1=T-\{x,y\}$表示了字母表 $C_1=C-\{x,y\}\cup\{z\}$的哈夫曼树，如图 4-2。

证：

$$
\begin{aligned}
B(T) &= \sum_{c\in C} f(c)d_T(c) \\
&= \sum_{v\in C-\{x,y\}} f(v)d_T(v) + f(x)d_T(x) + f(y)d_T(y) \\
&= \sum_{v\in C-\{x,y\}} f(v)d_{T_1}(v) + (f(x)+f(y))(d_{T_1}(z)+1) \\
&= \sum_{v\in C-\{x,y\}} f(v)d_{T_1}(v) + f(z)d_{T_1}(z) + (f(x)+f(y)) \\
&= \sum_{w\in C_1} f(w)d_{T_1}(w) + (f(x)+f(y)) \\
&= B(T_1) + f(x) + f(y)
\end{aligned}
$$

若 T_1 是 C_1 的非优化前缀编码树，则必存在 T_2，其叶子在 C_1 中，使 $B(T_2)<B(T_1)$。

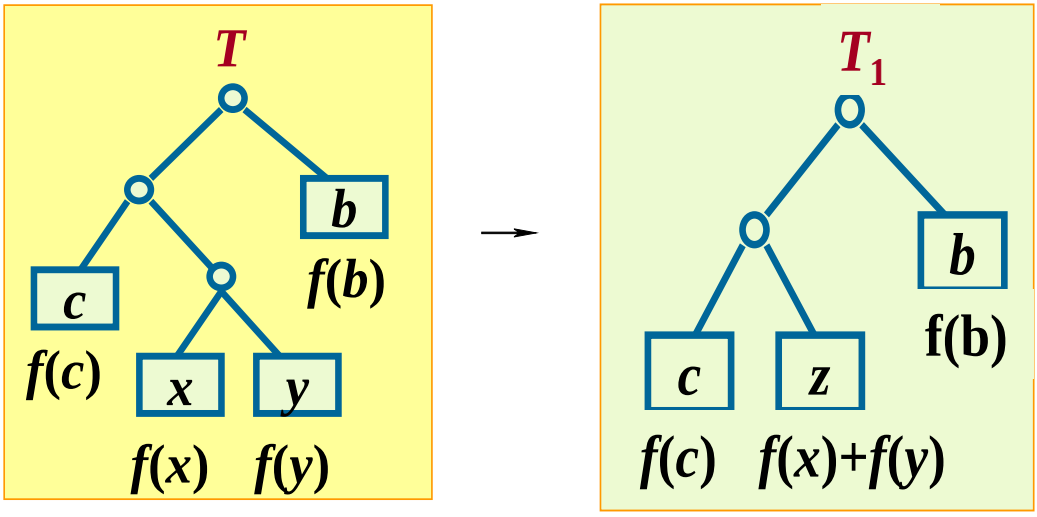

图 4-2 把兄弟叶结点删掉缩小问题规模

因为 z 在 C_1 中是一个字符，它必是 T_2 中的叶子。把结点 x，y 加入 T_2，作为 z 的子结点，则得到 C 的一个前缀树 T_3，如图 4-3。T_3 的代价为

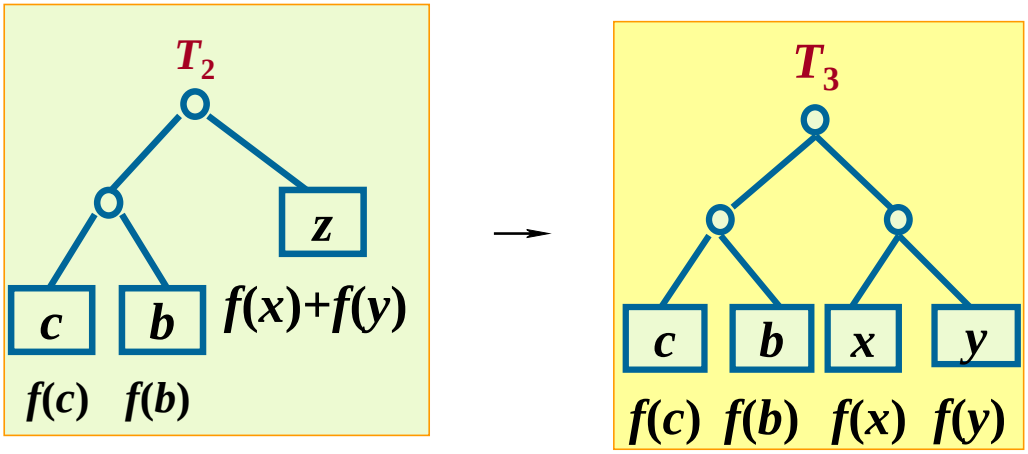

图 4-3 把结点 z 展开一对兄弟叶结点

$$
\begin{aligned}
B(T_3) &= \sum_{c\in C_1-\{z\}} f(c)d_{T_2}(c) + (f(x)+f(y))(d_{T_2}(z)+1) \\
&= \sum_{c\in C_1-\{z\}} f(c)d_{T_2}(c) + f(z)d_{T_2}(z) + f(x) + f(y) \\
&= B(T_2) + f(x) + f(y) \\
&< B(T_1) + f(x) + f(y)
\end{aligned}
$$

$= B(T)$

与 T 是优化的(即 HUFFMAN 树)矛盾，故 T_1 是优化的。

② 证明具有贪心选择性

定理 5（Greedy 选择性） 设 C 是字母表，$\forall c \in C$， $f(c)$是 c 在文件中出现的频率。设 x，y 是 C 中具有最小概率的两个字符，则存在一个 C 的优化前缀树，在该优化前缀树中 x，y 具有最大深度，其编码长度相同，仅在最末一位不同。

证：设 T 是 C 的优化前缀树，且 b，c 是具有最大深度的两个字符，如图 4-4。不失一般性，设 $f(b)<f(c)$，$f(x)<f(y)$。

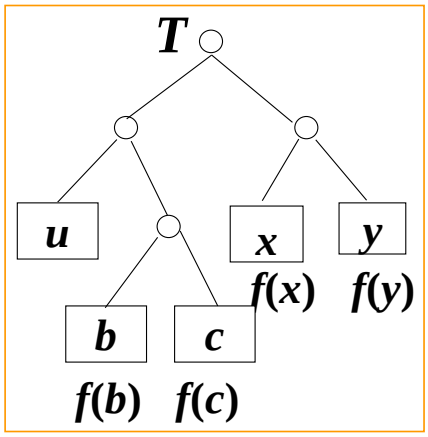

图 4-4 含有比 x，y 还深结点的优化前缀树

因 x 与 y 是具有最低概率的字符，所以 $f(b)\geq f(x)$，$f(c)\geq f(y)$。

从 T 构造 T_1，交换 T 的 b 和 x；从 T_1 构造 T_2，交换 T_1 的 c 和 y。往证，T_2 是更优前缀树。

$$B(T)-B(T_1)=\sum_{c\in C} f(c)d_T(c)-\sum_{c\in C} f(c)d_{T_1}(c)$$

$$= f(x)d_T(x)+f(b)d_T(b)-(f(x)d_{T_1}(x)+f(b)d_{T_1}(b))$$

$$= f(x)d_T(x)+f(b)d_T(b)-f(x)d_T(b)-f(b)d_T(x)$$

$$=(f(b)-f(x))(d_T(b)-d_T(x))$$

因为 $f(b)\geq f(x)$，$d_T(b)\geq d_T(x)$（因为 b 的深度最大）。所以 $B(T)$-$B(T_1)\geq 0$，$B(T)\geq B(T_1)$。

同理可证，$B(T_1)\geq B(T_2)$。于是 $B(T)\geq B(T_2)$（即 T_2 是更优的）。

又由于 T 是优化的，所以 $B(T)\leq B(T_2)$。

于是 $B(T)=B(T_2)$。

所以 T_2 是 C 的最优前缀编码树，在 T_2 中 x，y 具有相同长度编码，而且仅最后一位不同（且在该优化前缀树中具有最大深度）。

③ 哈夫曼算法 HUFFMAN 是按定理 5 的 Greedy 选择性进行局部优化选择的。

因此哈夫曼算法是正确的，即 HUFFMAN(C)产生 C 的一个最优前缀码。

本章习题

1. 设有一个活动的集合 S={1,2,…,11}如下表所示。其中 s_j，f_j 是活动 j 的起止时间，且 $f_1 \le f_2 \le \ldots \le f_{11}$。$\boldsymbol{l_i}$ 是 S_i ={j∈S | $s_j \ge f_{l_{i-1}}$ }中具有最小结束时间 f_{l_i} 的活动(设 $l_0=0, f_0=0$)。设 A 是 S 的包含活动 1 的优化解。请写出每个 $\boldsymbol{S_i}$ 及 $\boldsymbol{l_i}$，并据此构造出 S 的包含活动 1 的优化解 A。最后针对此例子说明贪心选择性。

活动集合 S 中活动的起止时间表

S	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	5	0	3	5	6	2	8	8	12
f_i	4	5	6	7	8	9	10	11	12	13	14

2. 背包问题是给定 n 种物品和一个背包，物品 i 的重量是 w[i]，其价值是 p[i]，背包的容量为 M。设物品已按单位重量价值递减的次序排序。每种物品不可以装入背包多次，但可以装入部分的物品 i，要选择装入背包中的物品，使得背包的总价值最大。 要求：

(1) 设计背包问题的贪心策略。

(2) 用 C/C++程序设计语言或伪代码写求解背包问题的贪心算法。

(3) 证明背包问题具有最优子结构性质和贪心选择性。

(4) 分析算法的时间复杂性。

【背包问题的形式化描述：

输入：正数 P_1，P_2，…, P_n, W_1，W_2, …, W_m, M

输出：X_1，X_2，…, X_n，$0 \le X_i \le 1$，使得

$\Sigma_{1 \le i \le n} P_i X_i$ 最大

$\Sigma_{1 \le i \le n} W_i X_i \le M$

】

3. 有一批集装箱要装上一艘载重量为 M 的轮船，其中集装箱 i 的重量为 Wi。现要将尽可能多的集

装箱装到轮船上。设集装箱已依其重量从小到大排序，（$x_1,x_2,\ldots,x_n$）是最优装载问题的一个最优解，其中，$x_i \in \{0,1\}$，$1 \le i \le n$(n 是集装箱总数)，且 $x_1=1$。请证明（$x_2, x_3,\ldots, x_n$）是轮船装载重量为 $M-w_1$，且待装船集装箱为{2，3，…，n}时相应最优装载问题的一个最优解。

4. 设计贪心算法，对任意给定的 x 轴上的 n 个闭区间，去掉尽可能少的闭区间，使剩下的闭区间都不相交。

要求：(1) 说明算法的贪心思想。

(2) 证明问题具有贪心选择性和优化子结构性质；

(3) 给出算法的伪代码。

(4) 简要分析算法的时间复杂性。

5. 给定实数轴上的 n 个区间构成的集合 X，其中各个区间的左端点依次存储在数组 XL[1:n]中；相应地，各个区间的右端点依次存储于数组 XR[1:n]中。试设计一个贪心算法找出 X 的一个子集 Y 使得：(a)Y 覆盖 X，即任意实数 x，只要 x 属于 X 的一个区间，则 x 也属于 Y 的一个区间；(b)Y 中区间的数量最小。答案要求包含以下内容：

(1) 给出贪心策略。

(2) 证明问题具有贪心选择性；

(3) 证明问题具有优化子结构；

(4) 根据贪心选择性和优化子结构写出算法；

(5) 分析算法的时间复杂度。

参考书

1. 王晓东. 计算机算法设计与分析 (第 5 版)[M]. 北京:电子工业出版社，2018.
2. Thomas H.Cormen, Charles E. Leiserson, and Ronald L. Rivest. Introduction to Algorithms (Second Edition)[M]. The MIT Press, 2013.