不等长记录的公式索引分组字典排序

徐绪松　　周建钦

(武汉大学)　(山东曲阜师范大学)

LEXICOGRAPHIC SORT OF RECORD OF VARYING LENGTH USING A NEW FORMULA TO DIVIDE STRING INTO GROUPS FORMED INDEX

Xu Xusong　　Zhou Jianqin

(*Wuhan University*)　(*Shandong Qufu Eduction University*)

ABSTRACT

This paper provides a lexicographic sort algorithm. Using a new build formula to divide strings into groups and to form an index. Algorathm sort a sequence of n elements in O(n) expected time. In an essential manner, without the comparison between the elements in traditional algorithm. The new method uses the new formula to calculate, getting the result of the sort directly.

摘　要

本文提出了一种公式索引分组字典排序法，其期望时间复杂性为O(n)。该算法基本上不象传统的排序方法那样进行元素间的比较，主要是用数学公式计算，直接得到排序结果。

排序在大型软件系统中有着重要作用，是一个值得研究的课题。本文提出了一种公式索引分组字典排序法，这种排序方法是利用公式$j=\left[\frac{B_i-\min}{\max-\min}\cdot n\right]$将字符串序列所对应的27 进制整数文件分成若干个组，组的顺序正好是串的字典顺序。

§1. 公式索引分组字典排序

1.1 输入

不等长字符串文件$A_1, A_2, \cdots, A_n$，其中$A_i=(a_{i1}, a_{i2}, \cdots, a_{ik})$。

1989 年10 月12 日收到，1990 年3 月25 日定稿。

1.2 输出

$A_1, A_2, \cdots, A_n$ 的一个排列$C_1, C_2, \cdots, C_n$, 满足$C_1 \le C_2 \le \cdots \le C_n$。

1.3 方法

(1)对输入字符串$A_i (i = 1, 2, \cdots, n)$, 截取前$x$ 位, $x = [\log_{27} n]$, 若字符串最长为p位, 且$p < x$, 则取$x = p$。截取字符串的前x 位, 不满x 位者用空格补齐, 得到相应的字符串序列$S_1, S_2, \cdots, S_N$, 其中$S_i = (s_{i1}, s_{i2}, \cdots, s_{ix})$, 用相应的十进制数替代英文字母(即: $0, 1, 2, \cdots, 26 \longleftrightarrow$ ⊔, $a, b, \cdots, z$), 再转换成27 进制数, 得到相应的整数文件: $B_1, B_2, \cdots, B_n$。

(2) 把字符串文件$A_1, A_2, \cdots, A_n$ 分成$n + 1$ 组。

设min = minimun $\{B_1, B_2, \cdots, B_n\}$, max = maximum $\{B_1, B_2, \cdots, B_n\}$, 用公式 $j = \left[\dfrac{B_i - \min}{\max - \min} \cdot n\right]$ (公式1.1), 则将A_i 分到第j 组($j = 0, 1, 2, \cdots, n$)。组号顺序为$A_1, A_2, \cdots, A_n$的字典顺序。

(3) 若某个组的字符串个数多于1 个, 即发生冲突, 则用链地址法处理, 并用插入排序将串按字典顺序链接。

(4) 增加记录时, 将它加在串文件的后边, 用公式1.1 得到它所在的组, 仅对一个组做插入。删去记录亦是如此。

§2. 公式索引分组字典排序算法

2.1 存贮结构

用数组$a[\ ]$顺序存放字符串, 每个串尾放ϕ。

建立索引表$b[\ ]$, 结点结构为 | number | link |, 其中number: 存放截取字符串前x 位后转换的27 进制数; link: 存放字符串在数组$a[\ ]$中的首地址。

构造一个索引分组表$c[\ ]$ ($c[\ ]$ 的下标为组号), 该表为链式结构, 用来存放分组后的串。在不同组的串用表头向量$c[\ ]$存放, 在同一组的用单链表链在$c[\ ]$的后边。它们的结点结构同为 | point | next |, 其中point: 存放字符串在数组b[ ]中的序号(下标地址); next: 该组中的下一个。

2.2 公式索引分组排序算法

该算法含四个子算法。

子算法1— 字符串转换 PROCEDURE stringtrans (p, a[ ], u, w), {p 表示字符串在a[ ]中的地址, u、w 为工作单元, 其中, u 存放二进制数, w 存放截取的字符。}

```
    BEGIN
1     x: =round (log (n) /log (27))
2     FOR  i: =1   TO   n  DO
3         y:=x; u: =0
4         p: =b[i]. link
5         WHILE p-b[i]. link+1 ≤ x  DO
7           IF p ≥ b [i+1]. link
8           THEN w: =0
9           ELSE w: =a[p]
10          u: =u+w * (exp ((y-1)* ln (27)));
11          y: =y+1;
```

```
12          p:=p+1;
13        b [i]. number: =u;
    END;
```

子算法2 — 挑出最大、最小者 PROCEDURE maxmin (max, min, b[ ]), { 在b[ ]. number 中挑出最大者存放在max 中，挑出最小者存放在min 中。}

```
    BEGIN
1   min: =b[1]. number; max: =min;
2   FOR   i:=2   TO   n   DO
3         IF   b[i]. number <min
4         THEN  min: =b[i]. number
5         IF   b[i]. number >max
6         THEN   max: =b[i]. number
    END;
```

子算法3— 公式索引分组字典排序 PROCEDURE sort (i, j, p, s, q), {j 为组号(c[ ] 的下标), 若c[j] 空，则将i(相应的整数在b[ ] 中的下标) 放入c[j].point 中；否则，按字典顺序链在c[j] 上，链接的结点地址p 从可利用栈取得，链接的位置通过调用过程search (s, q) 得到(p 插在s, q 之间)。}

```
    BEGIN
1   FOR i:=1 TO n DO
2     j:=round ((b[i]. number-min)*n/(max-min));
3     IF  c[j].point =∧
4     THEN [c[j].point: =i; c[j]. next: =∧]
5     ELSE [newl (p)
6             CASE
7               b[i]. number <b [c[j]. point ]. number:
                  [p↑. next:=c[j]. next; p↑. point: =c[j]. point;
                  c[j]. next: =p; c[j]. point: =i]
8               b[i]. number =b[c[j]. point]. number:
                  [WHILE a[b[i].link +x]=a[b[c[j].point].link+x]
                        DO [x: =x+1]
                  IF a[b[i]. link+x ]<a[b[c[j].point].link+x]
                  THEN [p↑. next:= c[j]. next; p↑. point: =c[j]. point;
                       c[j].next:=p; c[j].point:=i]
                       ELSE [p↑. point: =i;
                       IF c[j]. next=NIL
                       THEN [c[j]. next:=p; p↑next:=NIL]
                  ELSE [q: =c[j]. next;
                          CALL search (q, s);
                          p↑. next:=q; s↑. next:=p]]]
9               b[i]. number > b[c[j]. point]. number:
                     [p↑. point:=i;
                     IF c[j]. next=NIL
                     THEN [c[j]. next: =p; p↑. next:=NIL]
                     ELSE [q:=c[j]. next;
                             CALL search (q, s);
                             p↑. next:=q; s↑.next: =p]]
```

```
    END;
                        PROCEDURE      search (q, s);
{ 在以c[j] 为头的单链表中查找结点i 的插入位置。}
    BEGIN
1   WHILE (q<>0) AND (b[i]. number >b [q↑.point]. number)
2             DO [s:=q; q:=q↑. next]
3   IF        b[i]. number =b[q↑.point].number
4   THEN  [WHILE a[b[i]. link+x=a[b[q↑.point]. link+x]
5                DO   [x:=x+1]
6           IF    a [b[i]. link +x > a [b[q↑. point]. link +x]
            THEN   [REPEAT
                        s:=q; q:=q↑. next;
                        WHILE   a[b[i]. link+x]=a[b[q↑. point]. link+x]
                           DO [x:=x+1]
                    UNTIL (a[b[i]. link+x]< a[b[q↑. point]. link+x)
                           or (q=NIL);]
    END;
```

子算法4— 输出算法　PROCEDURE　output (y, t), { 按表头向量c[ ] 的顺序输出字符串序列，在同一链表中的则沿着链输出。y、t 为工作单元，Y 存放串在b[ ] 中的下标，t 存放串在a[ ] 中的首地址。}

```
    BEGIN
1   FOR j:=0  TO   n   DO
2     y:=c[j]. point;
3     IF    y ≠0
4     THEN  [t: =b[y]. link
5             WHILE a[t] ≠ φ   DO
6                write ('a[t]')
7                t:=t+1;
8             q:=c[j]. next;
9             WHILE q ≠ ∧   DO
10               y:=q↑. point;
11               t:=b[y]. link;
12               WHILE a[t] ≠ φ   DO
13                write ( 'a[t]')
                  t:=t+1;
15               q:=q↑. next;
    END;
```

§3. 预备定理

定理3.1 字符串序列$S_1, S_2, \cdots, S_n$ 的字典顺序与它相应的27 进制数的整数序列B_1, B_2, B_n 的大小顺序一致。

证明: (1) 若$S_i \le S_j$, 则$B_i \le B_j$。

若$S_i \le S_j$

则 ①存在整数q, 使$s_{iq} < s_{jq}$, 而对所有的$p < q$, 有$s_{ip} = s_{jp}$

于是$s_{i1} \cdot 27^{x-1} + \cdots + s_{iq} \cdot 27^{x-q} + \cdots + s_{ix} \le s_{j1} \cdot 27^{x-1} + \cdots + s_{jq} \cdot 27^{x-q} + \cdots + s_{jx}$

故$B_i \le B_j$

或②$p \le q$, 对所有的$1 \le k \le p$, 用$s_{ik} = s_{jk}$

于是$s_{i1} \cdot 27^{x-1} + \cdots + s_{ip} \cdot 27^{x-p} + 0 \cdot 27^{x-p-1} + \cdots + 0 \le s_{j1} \cdot 27^{x-1} + \cdots + s_{jp} \cdot 27^{x-p} + s_{jq} \cdot 27^{x-q} + \cdots + s_{jx}$

故$B_i \le B_j$

(2) 若$B_i \le B_j$, 则$S_i \le S_j$。

设字符串是1 位

因为$\sqcup, a, b, \cdots, j \longleftrightarrow 0, 1, 2, \cdots, 26$

所以当$B_i \le B_j$ 时, $S_i \le S_j$ 成立

假设字符串为x 位, 若$B_i \le B_j$, 则有$S_i \le S_j$ 成立。若设$B_i = s_{i1} \cdot 27^{x-1} + s_{i2} \cdot 27^{x-2} + \cdots + s_{ix}$, $B_j = s_{j1} \cdot 27^{x-1} + s_{j2} \cdot 27^{x-2} + \cdots + s_{jx}$, 由假设$(s_{i1}, \cdots, s_{ix}) < (s_{j1}, \cdots, s_{jx})$, 即存在一个$q$, $s_{iq} < s_{jq}$, 而对所有的$p < q$, $s_{ip} = s_{jp}$。

在字符串为$x+1$ 位时,

因为 $B_i = s_{i1} \cdot 27^{x} + s_{i2} \cdot 27^{x-1} + \cdots + s_{iq} \cdot 27^{x-q+1} + \cdots + s_{i,x+1}$, $B_j = s_{j1} \cdot 27^{x} + s_{j2} \cdot 27^{x-1} + \cdots + s_{jq} \cdot 27^{x-q+1} + \cdots + s_{j,x+1}$, 由归纳假设, $s_{iq} \le s_{jq}$

所以$(s_{i1}, \cdots, s_{iq}, \cdots, s_{i,x+1}) \le (s_{j1}, \cdots, s_{jq}, \cdots, s_{j,x+1})$

故$S_i \le S_j$

定理3.2　由公式1.1 将字符串分组的顺序与原整数序列的大小顺序一致。

证明: (1) 若$B_i < B_j$, 则$k_i \le k_j$ (k_i、k_j 为组的序号)

若$B_i < B_j$

则$\dfrac{B_i - min}{max - min} \cdot n < \dfrac{B_j - min}{max - min} \cdot n$, 于是$\left[\dfrac{B_i - min}{max - min} \cdot n\right] \le \left[\dfrac{B_j - min}{max - min} \cdot n\right]$

故$k_i \le k_j$

(2) 若$k_i < k_j$, 则$B_i < B_j$

因为$k_i < k_j$, 所以$\left[\dfrac{B_i - min}{max - min} \cdot n\right] < \left[\dfrac{B_j - min}{max - min} \cdot n\right]$, 所以$\dfrac{B_i - min}{max - min} \cdot n < \dfrac{B_j - min}{max - min} n$

故$B_i < B_j$

§4. 算法分析

定理4.1　公式索引分组字典排序的期望复杂性为$O(N)$。

算法的正确性。由定理3.2(1), 任意两个整数$B_i < B_j$, 公式1.1 分组后有$k_i \le k_j$。①若$k_i < k_j$, 由定理3.2(2) 必有$B_i < B_j$; ②若$k_i = k_j$, 算法用插入排序将B_i、B_j 按大小顺序链接在单链表$c[k_i]$ 中。所以数组c[ ] 及它的单链表已将整数序列排序。又由定理3.1, 字符串序列的字典顺序与用它转换成27 进制数的整数序列的大小顺序一致。所以由输出算法, 按组号0–N 的顺号(同一组的顺着链的顺序) 逐一输出的串符合字典顺序。

算法的复杂性。主要是排序所花的时间。

某一字符串分入第K 组, 相应的整数落入区间: $(min + (k-1)(max - min)/N, min + k(max - min)/N)$。这个事件发生的概率是:

$$P_k = \int_{min+(k-1)(max-min)/N}^{min+k(max-min)/N} f(x)dx$$

$f(x)$ 是字符串概率分布的密度函数，且$f(x) < D$(尽管某些字符串出现的频率很高，但都收敛于某一固定的概率值)。

所以$P_k < D(max - min)/N$

因为$max - min < B$

所以$P_k < D \cdot \frac{B}{N}$

因为插入r 个字符串的时间小于r^2，因而插入排序第k 组的，个字符串的平均时间小于:

$$\sum_{r=0}^{N} r^2 \cdot P(P是N个字符串进入第K\ 个组的概率)$$

$$= \sum_{r=0}^{N} r^2 \cdot \binom{N}{r} P_k^r (1 - P_k)^{N-r}$$

$$= NP_k(1 - P_k) + N^2 P_k^2$$

排序或插入全部N 个单词的平均时间小于:

$$\sum_{k=1}^{N+1} (NP_k(1 - P_k) + N^2 P_k^2)$$

$$\leq \sum_{k=1}^{N+1} (NP_k + N^2 P_k^2)$$

$$= N + N^2 \sum_{k=1}^{N+1} p_k^2$$

$$\leq N + N^2 \cdot \frac{BD}{N} \sum_{k=1}^{N+1} P_k = N(1 + BD)$$

所以算法的期望复杂性为$O(N)$。

公式索引分组字典排序N 个字符串序列的最坏复杂性为$O(N^2)$。

该算法已由武汉大学管理学院研究生张集元上机调试通过。用于实践，响应很快。

参考文献

[1] Aho A V, Hopcroft J E, Vuman J D. The Design and Analysis of Computer Algorithm. by Addison – Wesley Publishing Company, INC 1974.

[2] D. E. 克努特著，管纪文、苏运霖译，计算机程序设计技巧(第三卷 排序与查找)，国防工业出版社，1984。

[3] 徐绪松，数据结构与算法，武汉大学出版社，1987. 9。

[4] 徐绪松，对不等长度的字符串序列字典排序，武汉大学学报(自然科学版)，1989 年第2 期。