1982年 第2卷 第4期
Vol. 2, No. 4, 1982

数学研究与评论
JOURNAL OF MATHEMATICAL RESEARCH & EXPOSITION

# An Algorithm for Inverting Permutation*

Huang Bing-chao

Department of Computer Science, Jilin University

A permutation, $\pi$, of a finite set $S$ is a one-one transformation of $S$ into $S$. For example, the set might be $\{1, 2, 3, 4, 5, 6\}$, and the permutation 621543, also noted as

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 2 & 1 & 5 & 4 & 3 \end{pmatrix},$$

is the transformation $\pi$ such that $1\pi = 6$, $2\pi = 2$, $3\pi = 1$, $4\pi = 5$, $5\pi = 4$, $6\pi = 3$.

Inverting is a fundamental operation of a permutation. The inverse $\pi^{-1}$ of a permutation $\pi$ is the rearrangement which undoes the effect of $\pi$; if $i\pi = j$, then $j\pi^{-1} = i$. Thus the product $\pi\pi^{-1}$ equals the identity permutation.

Every permutation has an inverse; for example, the inverse of

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 2 & 1 & 5 & 4 & 3 \end{pmatrix} \text{ is } \begin{pmatrix} 6 & 2 & 1 & 5 & 4 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 2 & 6 & 5 & 4 & 1 \end{pmatrix}.$$

There is a simple method to compute the inverse of a permutation. Let us assume we are dealing with permutations of the numbers $\{1, 2, 3, \ldots, n\}$. If X[1]X[2]X[3]...X[$n$] is such a permutation, set Y[X[$k$]]$\leftarrow k$ for $1 \leqslant k \leqslant n$. Then Y[1]Y[2]...Y[$n$] is the desired inverse. This method uses $2n$ memory cells. When $n$ is a very large number, this method will use too much additional memory space. We want to compute the inverse "in place" so that after our algorithm is finished X[1]X[2]...X[$n$] is the inverse of the original permutation.

Professor D. E. Knuth introduces two algorithm for computing the inverse "in place" in his famous book[1]. One of them is the algorithm I (Reference: CACM 8, 1965.). It is implemented by MIX program I. This program uses 19 MIX instructions. Its running time is $(17N - 8C - S + 1)$, where N is the order of the permutation, C is the total number of cycles when it is expressed as a product of

* *Received Dec.* 24, 1981.

disjoint cycles, and S is the number of fixed elements (singleton cycles). The average value of C is $H_N = \ln N + 0.577 + O(1/N)$, the average value of S is 1, hence the average running time of program I is $17N - 8H_N$. Another algorithm is the algorithm J which is due to J. Boothroyd. It is implemented by MIX program J. This program is a little shorter than the preceding one. It has 14 MIX instructions. But its average running time is essentially proportional to $n \ln n$, while that of algorithm I is essentially proportional to $n$. Hence the algorithm I is definitely superior.

This paper gives a very simple algorithm for computing the inverse "in place". Its MIX program uses only 13 MIX instructions.

**Algorithm A.** (Inverse in place). Replace $X[1]X[2]...X[n]$, a permutation on $1, 2, ..., n$, by its inverse.

A1. [Initialize.] Set $m \leftarrow n$, $j \leftarrow -m$.

A2. [Next element.] Set $i \leftarrow X[m]$. If $i < 0$, go to A5 (this element has already been processed).

A3. [Invert one] Set $X[m] \leftarrow j$, $j \leftarrow -m$, $m \leftarrow i$.

A4. [End cycle.] Set $i \leftarrow X[m]$. If $i > 0$, go to A3; otherwise set $i \leftarrow j$.

A5. [Loop on $m$.] Set $X[m] \leftarrow -i$, decrease $m$ by 1; If $m > 0$, go to A2. Otherwise the algorithm terminates. ∎

This method is similar to the algorithm I, it is also based on inversion of successive cycles of the permutation. The main difference between the two methods is that the variable $m$ in the algorithm A acts as two variables $m$ and $k$ in the algorithm I.

For an example of this algorithm, see Table 1.

Table 1

COMPUTING THE INVERSE OF 621543 = (631)(54)(2) BY ALGORITHM A. (Read columns from left to right.)

| Step: | A1 | A4 | A4 | A4 | A5 | A4 | A4 | A5 | A5 | A5 | A4 | A5 | A5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X[1] | 6 | 6 | 6 | −3 | −3 | −3 | −3 | −3 | −3 | −3 | −3 | −3 | 3 |
| X[2] | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | −4 | 2 | 2 |
| X[3] | 1 | 1 | −6 | −6 | −6 | −6 | −6 | −6 | −6 | 6 | 6 | 6 | 6 |
| X[4] | 5 | 5 | 5 | 5 | 5 | 5 | −5 | −5 | 5 | 5 | 5 | 5 | 5 |
| X[5] | 4 | 4 | 4 | 4 | 4 | −1 | −1 | 4 | 4 | 4 | 4 | 4 | 4 |
| X[6] | 3 | −6 | −6 | −6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $m$ | 6 | 3 | 1 | 6 | 5 | 4 | 5 | 4 | 3 | 2 | 2 | 1 | 0 |
| $i$ | | 1 | 6 | −1 | −1 | 5 | −4 | −4 | −5 | −6 | −2 | −2 | −3 |
| $j$ | −6 | −6 | −3 | −1 | −1 | −5 | −4 | −4 | −4 | −4 | −2 | −2 | −2 |

**Program A** (Inverse in place). $rI1 = m$; $rI2 = (-i)$; $rI3 = j$. $n = \mathrm{N}$, a symbol to be defined when this program is assembled as part of a larger program.

```
01  INVERT  ENT1  N      1    A1. Initialize. m←n.
02           ENN3  0,1    1    j←−m.
03  2H       LD2N  X,1    N    A2. Next element. i←X[m].
04           J2P   5F     N    If i<0, go to A5.
05  3H       ST3   X,1    N    A3. Invert one. X[m]←j.
06           ENN3  0,1    N    j←−m.
07           ENN1  0,2    N    m←i.
08           LD2N  X,1    N    A4. End cycle. i←X[m].
09           J2N   3B     N    If i>0, go to A3.
10           ENN2  0,3    C    i←j.
11  5H       ST2   X,1    N    A5. Loop on m. X[m]←−i.
12           DEC1  1      N    m←m−1.
13           J1P   2B     N    If m>0, go to A2. ■
```

The running time of this program is $14\mathrm{N}+\mathrm{C}+2$, where N is the order of the permutation, C is the total number of cycles. $\mathrm{C} = (\min\ 1,\ \mathrm{ave}\ \mathrm{H_N},\ \max\ \mathrm{N})$. Hence, the minimum running time of program A is $14\mathrm{N}+3$, the maximum running time is $15\mathrm{N}+2$, and the average running time is $14\mathrm{N}+\mathrm{H_N}+2$. Thus it is superior to preceding two algorithm.

## Bibliography

1. Knuth, D. E., The Art of Computer Programming, Vol. 1 (Fundamental Algorithms), Addison-Wesley, Reading, Mass., 1968.

# 对于置换求逆的一个算法

黄秉超

（吉林大学计算机科学系）

本文提出的一种就地求逆算法只占用 $n+3$ 个存贮单元，相应的 MIX 程序只用 13 条指令，运行时间 $14N+C+2$ 比 [1] 中介绍的两种算法都更快、更省、更紧凑。