最終順列が常に互換となる互換法順列生成†

玉木久夫†† 孟繁楨†††

互換による再帰的順列生成法において，すべての順列を生成しつくしたあとの最終的な順列が，常に最初の状態に対して互換になるようなアルゴリズムを導く．この性質を利用すると，部分順列生成のスキップを一回の互換で行うことができる．このアルゴリズムは通常の一引数の再帰手続きによる方法を二引数に一般化することによって得られる．アルゴリズム導出の過程において，一引数によっては要求をみたすアルゴリズムは得られないことが示される．この結果は Wells によって提出された問題にたいする解答となっている．

1. まえがき

与えられた n 個の要素の並びから $n!-1$ 回の互換によってすべての順列を生成する方法のあるものは，次のような再帰的アルゴリズムとして定式化すると分かりやすい[1]．

[アルゴリズム 1]

```
procedure perm(n);
  if n = 2 then swap(1,2)
  else begin for i := 1 to n-1 do
                 begin perm(n-1);
                       swap(n, k(n, i))
                 end;
               perm(n-1)
       end;
```

ここで，swap(i, j) は，i 番目の要素と j 番目の要素を交換する基本手続きである．k は正整数 n および $1 \leqq i \leqq n$ なる整数 i に $1 \leqq k(n, i) \leqq n-1$ なる整数 $k(n, i)$ を対応させる関数（インデックス関数と呼ぶ）である．

手続き呼び出し perm(n) は，そのなかの再帰呼び出し perm($n-1$) が正しく働き，$n-1$ 回の swap が最初に一番目から $n-1$ 番目にあった要素をちょうど一回ずつ n 番目の位置に取り出すとき，正しく働く．このような条件を満たすインデックス関数として，Wells[2] および Heap[3] はそれぞれ次のような単純なものを与えている．

$$\text{Wells}: k(n, i) = \begin{cases} n-i & n \text{ が偶数で } i>2 \text{ のとき} \\ n-1 & \text{それ以外} \end{cases}$$

$$\text{Heap}: k(n, i) = \begin{cases} 1 & n \text{ が奇数のとき} \\ i & \text{それ以外} \end{cases}$$

この順列生成手続き perm (n) を実行した結果の（最初の状態に対する）順列を最終順列と呼ぶことにしよう．Wells, Heap のインデックス関数による方法は共に次のような顕著な性質を持っている．すなわち，perm(n) の最終順列は

（1） n が奇数のとき互換であり，

（2） n が偶数のとき周期 n の単一循環である．

(Wells[2] 参照．また，本稿 3 章の内容を応用して容易に証明することができる．)

性質（1）は，順列生成の途中で部分順列の生成をスキップしたいときに役立つ．すなわち，再帰呼び出し perm(n) は，n が奇数ならば一回の互換で置き換えることによって全体の生成系列を乱すことなくスキップすることができる．ところが n が偶数のときには，perm(n) のスキップには少なくとも n 個のデータ移動を行わなければならない．このため Wells[2]は，「同程度に簡単な方法で最終順列が常に互換になるようなものはないか」という問題を提出している．

次章以下で，再帰手続き perm を二引数に一般化することによってこの問題が解かれることを示す．その過程で一引数のままでは解がない，すなわち上の定式化のもとで求める性質を満たすインデックス関数は存在しないことが示される．

2. 定義と表記法

n を正整数とするとき，正整数全体から正整数全体への全単射 π で，n より大きいすべての整数 m について $\pi(m)=m$ であるようなものを長さ n の順列と呼び，その全体を Π_n とおく．あきらかに $\Pi_1 \subset \Pi_2 \subset \cdots$ である．$\cup_m \Pi_n$ を Π とおき，その要素を単に順列と

† Permutation Generation by Transposition with the Final Permutation Always Being a Transposition by HISAO TAMAKI (Department of Information Science, Faculty of Engineering, Ibaraki University) and FAN-ZHEN MENG (The Department of Mathematics of Tianjin University of China).

†† 茨城大学工学部情報工学科
††† 中国天津大学数学系

よぶ．Πは写像の合成$\circ$のもとに$(\pi_1\circ\pi_2(i)=\pi_2(\pi_1(i)))$群を成し，各$\Pi_n$はその部分群である．その単位元（恒等写像）を$I$，順列$\pi$の逆元を$\pi^{-1}$で表す．

順列πと順列の集合Pとの積を

$$\pi\circ P=\{\pi\circ\rho\mid\rho\in P\}$$

によって定義する．

また$a_1,\cdots,a_k$をk個の相異なる正整数とするとき$(k\geqq 1)$，表記法$<a_1,\cdots,a_k>$によって次のように定義される順列を表し，これを周期kの単一循環と呼ぶ．

$$\begin{aligned}<a_1,\cdots,a_k>(m)&=a_i \quad m=a_{i-1},\ 1<i\leqq k\ \text{のとき}\\ &=a_1 \quad m=a_k\ \text{のとき}\\ &=m \quad \text{それ以外}\end{aligned}$$

明らかに$<a_1,\cdots,a_k>$，$<a_2,\cdots,a_k,a_1>$，$\cdots$等k通りの表現はすべて同一の単一循環を表す．周期1の単一循環$<a>$は恒等置換Iに等しく，周期2の単一循環$<a,b>$$(a\neq b)$は互換と呼ばれる．

以下の節で使われる順列の性質のうち特に重要なものを次にあげる．

（1） $\pi=\in\Pi_{n-1}$, $i<n$ ならば
$<n,i>\circ\pi=\pi\circ<n,\pi(i)>$

（2） $k\geqq 3$のとき
$<a_1,\cdots,a_k>$
$=<a_1,a_2>\circ<a_1,a_3>\circ\cdots\circ<a_1,a_k>$

（3） 任意の$\pi\in\Pi_n$に対して$\Pi_n=\pi\circ\Pi_n$

（4） $n\geqq 2$に対して

$$\begin{aligned}\Pi_n=\ &\Pi_{n-1}\\ &\cup\ <n,1>\circ\Pi_{n-1}\\ &\cdots\\ &\cup\ <n,n-1>\circ\Pi_{n-1}\end{aligned}$$

3. アルゴリズムの導出

我々のアルゴリズムは，1章の終わりで述べたように，2引数の再帰手続きとしてあらわされる．

[アルゴリズム2]

```
procedure perm(n, j);
  if n = 2 then swap(1,2)
  else begin for i := 1 to n−1 do
            begin perm(n−1, h(n, j, i));
                  swap(n, k(n, j, i))
            end;
          perm(n−1, h (n, j, n))
  end;
```

ここで，引数jは$1\leqq j\leqq n-1$なる整数であり，hおよびkは正整数n，$1\leqq j\leqq n-1$なる整数j，および$1\leqq i\leqq n$なる整数iに対して$1\leqq h(n,j,i)\leqq n-2$，$1\leqq k(n,j,i)\leqq n-1$なる整数値を定める関数（インデックス関数）である．

手続き perm(n,j)の意図は，

（1） 長さnのすべての順列を生成し，

（2） その最終順列は互換$<n,j>$になることであり，そうなるようにインデックス関数を定めることが我々の目的である．

上の条件を厳密に述べるために，与えられたインデックス関数h,kに対して，手続き perm(n,j)の行う互換の列を$\sigma(h,k,n,j)$で表そう．すなわち，

$\sigma(h,k,2,1)$は互換$<1,2>$のみの単元列であり，

$\sigma(h,k,n,j)$ $(n>2)$は，

互換列$\sigma(h,k,n-1,h(n,j,1))$の次に

互換$<n,k(n,j,1)>$を置き，次に

互換列$\sigma(h,k,n-1,h(n,j,2))$を続け，次に

互換$<n,k(n,j,2)>$を置き，

これを$n-1$回繰り返して最後に

互換列$\sigma(h,k,n-1,h(n,j,n))$を続けたものである．

$\sigma(h,k,n,j)$の長さを$L(n)$とおく．

順列の有限列$\sigma=\pi_1,\pi_2,\cdots,\pi_M$に対して，先頭$m$要素$(0\leqq m\leqq M)$の積$\pi_1\circ\cdots\circ\pi_m$を$\sigma\backslash m$で表す．（ただし，$\sigma\backslash 0=I$とする．）また，列全体の積$\sigma\backslash M$を$\sigma\backslash$で表す．

そうすると，上で述べた手続き perm(n,j)に対する条件は次のように述べ直すことができる．

（条件1） $\{\sigma(h,k,n,j)\backslash m\mid 0\leqq m\leqq L(n)\}=\Pi_n$

（条件2） $\sigma(h,k,n,j)\backslash=<n,j>$

$n=2$に対しては条件1，2はh,kに無関係に成立している．そこで，nに関する帰納法によってh,kを定めるために，nとjを固定し$(n>2, 1\leqq j\leqq n-1)$，$n'=n-1$なるn'，および$1\leqq j'\leqq n'-1$なる任意のj'に対して条件1，2が成立していると仮定しよう．

i番目の再帰呼び出しの最終順列$\sigma(h,k,n-1,h(n,j,i))\backslash$を$\phi_i$とおき，さらに順列$\alpha_i$ $(i=1,\cdots,n)$およびβ_i $(i=0,\cdots,n)$を次のように定義する．

$$\begin{aligned}&\alpha_1=\beta_0=I\\ &\alpha_{i+1}=\alpha_i\circ\phi_i\circ<n,k(n,j,i)> \quad i=1,\cdots,n-1\\ &\beta_i=\beta_{i-1}\circ\phi_i \quad i=1,\cdots,n\end{aligned}$$

つまりα_iはi番目の再帰呼び出し直前の順列である．

そうすると,

$$\{\sigma(h,k,n,j)\backslash m \mid 0\leqq m\leqq L(n)\}$$
$$=\{\sigma(h,k,n-1,h(k,n,j))\backslash m \mid 0\leqq m\leqq L(n-1)\}$$
$$\cup\alpha_2\circ\{\sigma(h,k,n-1,h(k,n,j))\backslash m \mid 0\leqq m\leqq L(n-1)\}$$
$$\cdots$$
$$\cup\alpha_n\circ\{\sigma(h,k,n-1,j)\backslash m \mid 0\leqq m\leqq L(n-1)\}$$

と書けるが，帰納法の仮定のもとにこれは

$$\Pi_{n-1}\cup\alpha_2\circ\Pi_{n-1}\cup\cdots\cup\alpha_n\circ\Pi_{n-1}$$

と等しい.

ところで,

$$k'(i)=\beta_i^{-1}(k(n,j,i)) \tag{3.1}$$

とおくと

$$\alpha_i=<n,k'(1)>\circ\cdots\circ<n,k'(i-1)>\circ\beta_{i-1} \tag{3.2}$$

が成り立つ. なぜならば,

$$\alpha_1=I=\beta_0$$

であり，(3.2)式を仮定すると

$$\alpha_{i+1}=\alpha_i\circ\phi_i\circ<n,k(n,j,i)>$$
$$=<n,k'(1)>\circ\cdots\circ<n,k'(i-1)>\circ\beta_{i-1}\circ\phi_i\circ<n,k(n,j,i)>$$
$$=<n,k'(1)>\circ\cdots\circ<n,k'(i-1)>\circ\beta_i\circ<n,k(n,j,i)>$$
$$=<n,k'(1)>\circ\cdots\circ<n,k'(i-1)>\circ<n,k'(i)>\circ\beta_i$$

が導かれるからである. ($\beta_i\in\Pi_{n-1}$ に注意)

ここで

$$k''(i)=(<n,k'(1)>\circ\cdots\circ<n,k'(i)>)^{-1}(n)$$

とおくと，$<n,k'(1)>\circ\cdots\circ<n,k'(i)>$ は $\gamma_i\in\Pi_{n-1}$ なる γ_i によって $<n,k''(i)>\circ\gamma_i$ と表すことができる. ただし，便宜上 $<n,n>$ は恒等置換 I を表すものと解釈する. したがって

$$\alpha_i\circ\Pi_{n-1}$$
$$=(<n,k''(i-1)>\circ\gamma_{i-1}\circ\beta_{i-1})\circ\Pi_{n-1}$$
$$=<n,k''(i-1)>\circ(\gamma_{i-1}\circ\beta_{i-1}\circ\Pi_{n-1})$$
$$=<n,k''(i-1)>\circ\Pi_{n-1}$$

となるから,

$$\{\sigma(h,k,n,j)\mid 0\leqq m\leqq L(n)\}$$
$$=\Pi_{n-1}\cup<n,k''(1)>\circ\Pi_{n-1}$$
$$\cdots$$
$$\cup<n,k''(n-1)>\circ\Pi_{n-1}$$

が成り立つ. これが Π_n に等しいための必要十分条件は明らかに

$$\{k''(1),\cdots,k''(n-1)\}=\{1,\cdots,n-1\} \tag{3.3}$$

となることである. ところが，$k''(i)$ はその定義より $k'(1),\cdots,k'(i)$，あるいは n のいずれかに等しいから，(3.3)式が成り立つためには,

$$\{k'(1),\cdots,k'(n-1)\}=\{1,\cdots,n-1\} \tag{3.4}$$

であることが必要である. また，(3.4)が成り立つとき，$k''(i)$ は $k'(i)$ と等しくなるので，(3.3)も成り立つ.

ゆえに，(3.4)式は帰納法の仮定のもとで条件1が成り立つための必要十分条件である.

さてこの条件の成り立つとき,

$$\sigma(h,k,n,j)\backslash$$
$$=\alpha_n\circ\phi_n$$
$$=<n,k'(1)>\circ\cdots\circ<n,k'(n-1)>\circ\beta_n$$
$$=<n,k'(1),\cdots,k'(n-1)>\circ\beta_n$$

であるから，これが互換 $<n,j>$ となるためには

$$\beta_n=<n,k'(1),\cdots,k'(n-1)>^{-1}\circ<n,j>$$

でなければならない. $\beta_n\in\Pi_{n-1}$ すなわち $\beta_n(n)=n$ の条件より $k'(n-1)=j$ が定まり，このとき

$$\beta_n=<k'(1),\cdots,k'(n-1)>^{-1}$$
$$=<k'(n-1),\cdots,k'(1)> \tag{3.5}$$

となる.

ここにおいて，再帰手続きを2引数に一般化したことの必要性がわかる. すなわち，1章における1引数の定式化は，2引数の定式化においてインデックス関数 k の値がその第2引数 j に依存しない特別な場合に相当するが，その場合 $\phi_1,\cdots,\phi_n$ はすべて同一の順列となり，それが互換であるという帰納法の仮定のもとでは $n\geqq 4$ に対して(3.4)式は成立しえない. 言い換えれば，1引数の定式化のもとでは perm$(n-1)$ の最終順列が互換であるとき，perm(n) の最終順列もまた互換になるような正しいインデックス関数は存在しない.

さて $k'(i)$ $(i=1,\cdots,n-1)$ の値を $k'(n-1)=j$ および(3.4)の条件を守ったうえで任意に決めると，$\phi_i=<n-1,h(n,j,i)>$ という帰納法の仮定のもとで，(3.5)式を成立させるよう，$h(n,j,i)$ $(i=1,\cdots,n)$ の値を選ぶことができる. このとき，$k(n,j,i)$ $(i=1,\cdots,n-1)$ の値は，k' の定義式(3.1)より自動的に決まる.

インデックス関数はなるべく規則的であることが望ましい. そこで,

$$k'(n-1)=j,$$
$$k'(n-2)=\mathrm{prev}(j),$$
$$\cdots\cdots$$
$$k'(n-i)=\mathrm{prev}^{i-1}(j)$$

……

$k'(1) = \mathrm{prev}^{n-2}(j)$ (3.6)

ただし

$\mathrm{prev} = <n-1, n-2, \cdots, 1>$

とおくと

$\beta_n = \mathrm{prev}$

となり，これは例えば

$\phi_i = <n-1, n-i-1>$ $i=1, \cdots, n-2$

$\phi_{n-1} = \phi_n = <n-1, n-2>$

として実現される．したがって帰納法の仮定のもとでは

$h(n, j, i) = n-i-1$ $i = 1, \cdots, n-2$

$h(n, j, n-1) = h(n, j, n) = n-2$ (3.7)

と定めれば良い．このとき β_i は

$\beta_i = <n-1, n-2> \circ \cdots \circ <n-1, n-i-1>$

$= <n-1, \cdots, n-i-1>$ $i = 1, \cdots, n-2$

$\beta_{n-1} = \beta_{n-2} \circ <n-1, n-2> = <n-2, \cdots, 1>$

$\beta_n = \beta_{n-2} = \mathrm{prev}$

となる．

インデックス関数 k の値は(3.1)および(3.6)より

$k(n, j, i) = \beta_i(k'(i))$

$= \beta_i(\mathrm{prev}^{n-i-1}(j))$

$= \beta_i(\mathrm{suc}^i(j))$ $i=1, \cdots, n-1$

ただし $\mathrm{suc} = \mathrm{prev}^{-1}$，となる．これらのインデックス関数の設定によってアルゴリズム2を具体化するとアルゴリズム3が得られる．

[アルゴリズム3]

```
procedure perm(n, j : integer);
  var h, s : integer;
  begin
    if n = 2 then swap(1, 2)
    else
      begin
        s := j;
        for h := n-2 downto 1 do
          begin {h = h(n, j, i)}
            perm(n-1, h);
            s := s mod (n-1)+1;
                 {s = suc^i(j)}
            if s > h then swap(n, s-1)
            else if s = h then swap(n, n-1)
              else swap(n, s)
          end;
        perm(n-1, n-2);
        if j = n-1 then swap(n, n-1)
          else if j = 1 then swap(n, n-2)
            else swap(n, j-1);
        perm(n-1, n-2)
      end
  end;
```

4. むすび

互換による再帰的順列生成法において最終順列が要素数の偶奇にかかわらず常に互換になる方法を導いた．また，その過程で通常の一引数再帰手続きによる方法ではこの性質を満たすことはできないことを示した．

実用的には，インデックス計算の余分な手間が必要なためこの方法が有効になるのは，次の2条件が満たされるときに限られるだろう．

(1) 比較的大きい n に対して，部分順列生成 perm(n) のスキップが頻繁に起こる．

(2) インデックス計算にくらべて要素交換のコストがかなり大きい．

謝辞 本研究の大部分は第二著者が茨城大学工学部情報工学科に留学中に行われた．御指導いただいた高岡忠雄教授ならびに情報工学科の皆様に感謝いたします．

参考文献

1) Sedgewick, R.: Permutation Generation Methods, *Comput. Surv.*, Vol. 9, No. 2, pp. 137-155 (1977).
2) Wells, M. B.: Generation of Permutations by Transposition, *Math. Comp.*, Vol. 15, pp. 192-195 (1961).
3) Heap, B. R.: Permutations by Interchanges, *Comput. J.*, Vol. 6, pp. 293-294 (1963).

(昭和61年9月18日受付)
(昭和62年3月25日採録)

玉木 久夫（正会員）
昭和27年生．昭和50年東京大学理学部物理学科卒業，52年同修士課程修了．茨城大学工学部情報工学科助手．現在の研究テーマは論理プログラミングとプログラム変換．ACM会員．

孟 繁楨
1946年生．1969年中国天津大学機械系卒業．1971年同大学数学系助手．1985年茨城大学大学院修士課程修了．現在天津大学数学系講師．データベースの研究・開発に従事している．中国科学技術協会会員．