数理解析研究所講究録
第1849巻 2013年 12-17

撹乱順列の線形時間ランキングとアンランキングについて

Linear time ranking and unranking of derangements

新潟大学・情報基盤センター　三河 賢治
Kenji Mikawa
Center for Academic Information Center,
Niigata University

神奈川大学・理学部　田中 賢
Ken Tanaka
Faculty of Science,
Kanagawa University

1 まえがき

n 個の自然数からなる集合 $[n]=\{1,2,\ldots,n\}$ 上の撹乱順列は，$[n]$ 上の順列 $\delta=\delta(1)\delta(2)\ldots\delta(n)$ のうち，不動点を持たないもの，すなわち，$\forall i\in[n]$ に対して $\delta(i)\neq i$ であるような順列として定義される．撹乱順列の個数を $!n$ とおくと，$!1=0$ と $!2=1$ を境界条件として，$n\geq 3$ について，

$$!n=(n-1)\,(!(n-1)+!(n-2)) \tag{1}$$

のように定義される．様々な $!n$ の定義が知られており，そのうちの一つとして，

$$!n=n\times!(n-1)+(-1)^n \tag{2}$$

のように表すこともできる．ランキング関数 r は，撹乱順列を要素とする集合から $\{0,1,\ldots,!n-1\}$ への全単射であり，アンランキング関数 r^{-1} はその逆関数である．

$[n]$ 上の順列に対するランクとアンランクを求める問題では，いくつかの方法がよく知られている．次章で述べるが，辞書順に並べられた順列に対して，$O(n)$ 領域を用いて $O(n\log n)$ 時間でランクとアンランクを求めるアルゴリズムが知られていいる．一方，辞書順ではないが，ある一定の順序で並べられた順列に対して，Myrvold らは，Fisher-Yates シャッフル法を応用して，線形時間で順列のランクとアンランクを求めるアルゴリズムを提案した [2]．彼らは，Fisher-Yates シャッフル法でランダムに交換要素を決定していた部分を剰余で置き換えるという巧妙な技法を導入して，ランダム生成アルゴリズムからアンランキングアルゴリズムに変換する方法を示したと言える．Myrvold らの方法は本報告でも重要な役割を果たすので，第3章で詳しく述べる．

撹乱順列のランクとアンランクを求める問題では，不動点を持たないという条件がこの問題を難しくしており，これまでに提案されたアルゴリズムは，撹乱順列を列挙するアルゴリズムを基に，$O(n)$ 領域を用いて $O(n^2)$ 時間でランクとアンランクを求めるものであった．最近，著者らは，Myrvold らのアルゴリズムを撹乱順列に応用して，辞書順に並べられた撹乱順列の巡回表記のランクとアンランクを $O(n\log n)$ 時間で求めるアルゴリズムと，線形時間で撹乱順列をランダムに生成するアルゴリズムをそれぞれ発表した [3, 4]．本報告では，Fisher-Yates シャッフル法と Myrvold らのアルゴリズムを応用して，撹乱順列のランクとアンランクを線形時間で求めるアルゴリズムを提案する。

2 巡回表記

置換 π はいくつかのサイクルに分割することができる．例えば，次の置換

$$\pi=\begin{pmatrix}1&2&3&4&5&6&7\\5&7&4&3&1&2&6\end{pmatrix} \tag{3}$$

は，異なる三つのサイクル (15)，(276)，(34) からなる．(15)(276)(34) のようにサイクルを並べて表記したものは置換 π の巡回表記と呼ばれる．各サイクル (15)，(276)，(34) は互いに素であるから，サイクルを並べる順序は自由である．また，サイクルを構成する要素の順序についても，置換の順序が変わらなければ，(276)，(762)，(627) は同じとみなされる．本報告では，巡回表記で表される置換を一意に定めるために，次の二つの条件による標準表現 [5] を定める．(1) 各サイクルの最小要素を各サイクルの最後に並べ，(2) 各サイクルをそれらの最小要素の昇順に並べる．以下，この標準表現を巡回表記と呼ぶことにする．標準表現に従うと，サイクルの括弧を省略しても置換を一意に定めることができるので，以下，巡回表記は括弧を省略して表すことにする．上記の置換 π は 5176243 と書ける．

撹乱順列は制限された置換のひとつなので，順列の巡回表記と同様に，撹乱順列も巡回表記で表す．$[n]$ 上のすべての撹乱順列の巡回表記を要素とする集合を $\mathfrak{C}([n])$ と表し，$\sigma, \sigma' \in \mathfrak{C}([n])$ について，σ が σ' よりも先順であることを $\sigma \prec \sigma'$ と表す．σ が辞書順で σ' よりも先順であるとは，$\sigma(1) < \sigma'(1)$ または $(\sigma(1) = \sigma'(1)) \wedge (\sigma(2)\ldots\sigma(n) \prec \sigma'(2)\ldots\sigma'(n))$ のどちらか一方を満たすときである．撹乱順列は不動点を持たない，すなわち $\sigma(1) \neq 1$，かつ，すべてのサイクルは少なくとも 2 個以上の要素からなるので，先頭の要素 $\sigma(1)$ はいつでも差集合 $[n] \setminus \{1\}$ の中から選ばれる．第 2 番目の要素 $\sigma(2)$ は $\sigma(2) = 1$ または $\sigma(2) \neq 1$ のどちらか一方の状態にあり，先頭の 2 つの要素 $\sigma(1)$，$\sigma(2)$ に注目すると，$\sigma(1)\sigma(2)$ で先頭のサイクルを形成する場合は $\sigma(2) = 1$ が成り立ち，先頭のサイクルが開いたままである場合は $\sigma(2) \in [n] \setminus \{1, \sigma(1)\}$ が成り立つ．先頭のサイクル $\sigma(1)\sigma(2)$ が固定されると，残りの要素から長さ $n-2$ の撹乱順列 $\sigma(3)\sigma(4)\ldots\sigma(n)$ が生成される．このような撹乱順列の総数は $(n-1)\times!(n-2)$ 個である．一方，先頭のサイクルが開いたまま，すなわち $\sigma(2) \neq 1$ のとき，残りの要素から長さ $n-1$ の撹乱順列 $\sigma(2)\sigma(3)\ldots\sigma(n)$ を新規に生成することに等しい．このような撹乱順列の総数は $(n-1)\times!(n-1)$ 個である．したがって，巡回表記で表された $[n]$ 上の撹乱順列の総数も (1) 式と同じ漸化式を得る．

本節の最後に，撹乱順列 δ とその巡回表記 σ との関係について述べる．以下の性質を利用して，線形時間で σ から δ に変換するアルゴリズムを構成する．k 個のサイクルで構成された巡回表記 $\sigma_1\sigma_2\ldots\sigma_k$ について，サイクル σ_i は m_i 個の要素 $\sigma_i(1)\sigma_i(2)\ldots\sigma_i(m_i)$ からなると仮定する．標準表現の定義から，

$$\sigma_i(m_i) = \min \sigma_i(1), \sigma_i(2), \ldots, \sigma_i(m_i) \tag{4}$$

と

$$\sigma_1(m_1) < \sigma_2(m_2) < \cdots < \sigma_k(m_k) \tag{5}$$

が成り立つ．また，巡回表記を通し番号で表した $\sigma(1)\sigma(2)\ldots\sigma(n)$ について，

$$\sigma(j) = \begin{cases} \delta(\sigma(j-1)) & \\ \qquad \sum_{t=1}^{i-1} |\sigma_t| + 1 < j < \sum_{t=1}^{i} |\sigma_t| & \\ \delta(m_i) \quad \sigma(j-1) = m_{i-1} \text{ の場合} & \end{cases} \tag{6}$$

が成り立つ．これらの性質を利用して，$\sigma(n)$ から先頭に向けて進み，各 $\sigma(j)$ で (6) 式から δ を復元する．各 $\sigma(j)$ において，δ を構成する要素の値が重複なく 1 個だけ決まるので，線形時間で σ から δ に復元できる．

3 既存の研究成果

本節では，Derstenfeld が紹介した Fisher-Yates シャッフル法 [1] と Myrvold らが提案した順列のアンランキングアルゴリズムを解説する．また，著者らが文献 [4] で提案した撹乱順列のランダム生成アルゴリズムを解説する．

$[n]$ 上の順列 $\pi = \pi(1)\ldots\pi(n)$ に対する Fisher-Yates シャッフル法を以下に示す．関数 rand(i) は 0 から $i-1$ までの一様な乱数を生成する．

for $i := n$ **downto** 2 **do begin**
  swap($\pi(i)$,π(rand(i) + 1));
end;

Fisher-Yates シャッフル法は，$\pi(n)$ から $\pi(2)$ まで要素の交換を繰り返して順列を確定する．$\pi(i)$ と交換するためにランダムに選ばれた要素の位置ベクトルを $p = p(1)p(2)\dots p(n)$ とすると，異なる要素列からは異なる順列が生成される．$1 \le p(i) \le i$ から，$1 \times 2 \times \cdots \times n = n!$ 通りの位置ベクトルが存在することも明らかであろう．

Myrvold らは，互いに異なる $n!$ 個の位置ベクトル p に着目し，p から類推されるランク値 r に対応する順列を計算するアルゴリズムを構成した．便宜的に，ランク値は $0 \le r < n!$ とする．以下に Myrvold らのアルゴリズムを示す．

for $i := n$ **downto** 2 **do begin**
  swap($\pi(i)$,$\pi((r \bmod i) + 1)$);
  $r := \lfloor r/i \rfloor$;
end;

ランク値 r は，各ステップで商と剰余に分解される．商は次のステップのランク値に用いられ，剰余は $\pi(i)$ と交換すべき要素の特定に用いられる．ステップ i で要素の交換が完了すると，以降のステップにおいて，接尾辞 $\pi(i)\pi(i+1)\dots\pi(n)$ が変更されることはない．また，

$$\begin{aligned}&\{\lfloor r/i \rfloor : r \in \{0,1,\dots,i!-1\}\}\\ &\quad = \{0,1,\dots,(i-1)!-1\}\end{aligned} \tag{7}$$

が成り立ち，ステップ i で $\pi(i)\pi(i+1)\dots\pi(n)$ が確定後，以降のステップで，帰納的に，$(i-1)!$ 個の異なる $\pi(1)\pi(2)\dots\pi(i-1)$ が得られる．

次に，著者らが提案した，撹乱順列を線形時間でランダムに生成するアルゴリズム [4] を示す．

for $i := 1$ **to** $n-1$ **do begin**
  if some cycle should close at $\sigma(i)$ **then begin**
    $j := \sigma^{-1}(\min([n] \setminus \{\sigma(1),\dots,\sigma(i-1)\}))$;
    swap($\sigma(i)$, $\sigma(j)$);
  end else begin
    $j :=$ rand($n-i$) + i;
    if $\sigma(j) = \min([n] \setminus \{\sigma(1),\dots,\sigma(i-1)\})$
    then swap($\sigma(i)$, $\sigma(n)$) **else** swap($\sigma(i)$, $\sigma(j)$);
  end
end;

巡回表記 σ に対して，先頭の要素 $\sigma(1)$ から末尾の要素 $\sigma(n)$ に向かって要素の交換を繰り返す．Fisher-Yates シャッフル法と同様に，$\sigma(i)$ と交換すべき要素を $\sigma(i)$ から $\sigma(n)$ までの要素の中からランダムに選ぶ．ただし，巡回表記の性質から，$\sigma(i)$ でサイクルを閉じる場合と閉じない場合に分けて，$\sigma(i)$ の値を決定している．$\sigma(i)$ でサイクルを閉じるための条件については後述するとして，以下，$\sigma(i)$ でサイクルを閉じる場合（サイクルを閉じない場合）の要素のランダム交換について説明する．

$\sigma(i)$ でサイクルを閉じる場合，要素の選択にランダム性はなく，残りの要素 $[n] \setminus \{\sigma(1),\dots,\sigma(i-1)\}$ から最小の要素を選ぶ．$\sigma(i)$ でサイクルを閉じない場合，残りの要素から長さ $n-i+1$ の巡回表記 $\sigma(i)\dots\sigma(n)$ を新規に生成することに等しい．残りの要素から最小ではない要素を選ぶので，$n-i$ 個の要素からランダムに要素 $\sigma(j)$ を選び，$\sigma(j)$ が残りの要素の中で最小であれば $\sigma(n)$ を選ぶ．したがって，提案アルゴリズムが線形時間でランダム生成を完了するためには，$\sigma(i)$ でサイクルを閉じるための判定を $O(1)$ 時間で完了すること，残りの要素の中から最小の要素を $O(1)$ 時間で見つけること，の 2 点を実現しなければいけない．文献 [4] では，$\sigma(i)$ でサイクルを閉じるための判定について，

$$\text{rand}(!(n-i)+!(n-i+1)) < !(n-i) \tag{8}$$

を用いて $O(1)$ 時間で完了することを示した．具体的には，$!(n-i+1)$ の値は，すでに直前のステップで求めているので，

$$!(n-i) = \frac{!(n-i+1) - (-1)^{n-i}}{n-i+1} \tag{9}$$

から $O(1)$ 時間で計算することができる．

次に，残りの要素から最小の要素を見つけることについて，文献 [4] では，扱う集合が任意の集合で

はなく，線形順序集合であることに注目して，要素 $i \in [n]$ が使用済であるか否かを表す使用済フラグ used[i]，i が使用済区間の末尾要素であるときの使用済区間の先頭要素へのポインタ head[i]，i が使用済区間の先頭要素であるときの使用済区間の末尾要素へのポインタ tail[i] を用意して，最小元を $O(1)$ 時間で求める方法を与えた．提案アルゴリズムの初期状態では，全ての要素は未使用なので，used[i] = **false**，head[i] = i，tail[i] = i を初期値としている．ステップ i で $\sigma(i)$ と交換すべき要素 a が確定した後，各配列の値がどのように更新されるか図 1 に示す．要素 a が未使用から使用済に移行するとき，各配列の値の更新は，使用済区間内の要素 a の位置によって 4 通りの場合に分けられる．図 1(a) は，a に隣接する両方の要素が使用済の場合である．要素 a が使用済となり，左右の使用済区間が連結される．このとき，連結した使用済区間の先頭要素と末尾要素へのポインタを，次のように更新する．

$$\begin{aligned} &\mathrm{head}[\,\mathrm{tail}[a+1]\,] \leftarrow \mathrm{head}[a-1] \\ &\mathrm{tail}[\,\mathrm{head}[a-1]\,] \leftarrow \mathrm{tail}[a+1] \end{aligned} \tag{10}$$

図 1(b) は，a の右に隣接する要素だけが使用済の場合である．要素 a が使用済となり，a と右に隣接する使用済区間が連結される．このとき，先頭要素 a と右に隣接する使用済区間の末尾要素へのポインタを，次のように更新する．

$$\begin{aligned} &\mathrm{head}[\,\mathrm{tail}[a+1]\,] \leftarrow \mathrm{head}[a] \\ &\mathrm{tail}[a] \leftarrow \mathrm{tail}[a+1] \end{aligned} \tag{11}$$

図 1(c) は，a の左に隣接する要素だけが使用済の場合である．要素 a が使用済となり，a と左に隣接する使用済区間が連結される．このとき，末尾要素 a と左に隣接する使用済区間の先頭要素へのポインタを，次のように更新する．

$$\begin{aligned} &\mathrm{head}[a] \leftarrow \mathrm{head}[a-1] \\ &\mathrm{tail}[\,\mathrm{head}[a-1]\,] \leftarrow \mathrm{tail}[a] \end{aligned} \tag{12}$$

最後に，図 1(d) は，a に隣接するの両方に要素が未使用の場合である．要素 a が使用済となり，a だけの使用済区間が発生する．a だけの使用済区間は，当然，その先頭要素も末尾要素も a となり，ポインタを更新する必要がない．

補題 3.1 (文献 [4]) ステップ i において，次式が成り立つ．

$$min([n] \setminus \{\sigma(1), \ldots, \sigma(i-1)\}) = \begin{cases} 1 & 1 \in [n] \text{ が未使用の場合} \\ tail[1]+1 & 1 \in [n] \text{ が使用済の場合} \end{cases} \tag{13}$$

4 撹乱順列のアンランキング

Myrvold らは，Fisher-Yates シャッフル法のランダムに要素を選ぶ部分を剰余算に置き換え，線形時間のアンランキングを実現した．本報告も，文献 [4] のランダム生成アルゴリズムのランダムに要素を選ぶ部分を剰余算に置き換え，アンランキングアルゴリズムを構成する．以下に提案アルゴリズムを示す．

for $i := 1$ **to** $n-1$ **do begin**
  if some cycle should close at $\sigma(i)$ **then begin**
    $j := \sigma^{-1}(\min([n] \setminus \{\sigma(1), \ldots, \sigma(i-1)\}))$;
    swap$(\sigma(i), \sigma(j))$;
    $r := r - !(n-i+1)$;
  end else begin
    $j := r \bmod (n-i) + i$;
    if $\sigma(j) = \min([n] \setminus \{\sigma(1), \ldots, \sigma(i-1)\})$
    then swap$(\sigma(i), \sigma(n))$ **else** swap$(\sigma(i), \sigma(j))$;
    $r := \lfloor r/(n-i) \rfloor$;
  end
end;

はじめに，$\sigma(i)$ でサイクルを閉じるための判定部分を考える．ステップ i でのランク値 r_i について，

$$r_i \geq !(n-i+1) \tag{14}$$

が成り立つとき，$\sigma(i)$ でサイクルを閉じる．ランダム生成の場合と全く正反対の判定式のように見えるが，$r_i < !(n-i)$ としないことによって，各ステップでのランク値を正しく制御できるようになる．アルゴリズムに対して，ランク値 r は $0 \leq r < !n$ の

範囲で入力される．ランク値 r_i は，$\sigma(i-1)$ の振る舞いによって異なる値を取る．$\sigma(i-1)$ でサイクルを閉じている場合，$\sigma(i)$ でサイクルを閉じることはなく，$\sigma(i)$ から長さ $n-i+1$ の撹乱順列の巡回表記を構成することに等しい．したがって，ランク値 r_i は，$0 \leq r_i < !(n-i+1)$ の値を取る．提案アルゴリズムは，次のステップのランク値 r_{i+1} として，$r_{i+1} = \lfloor r_i/(n-i) \rfloor$ を与えるので，

$$\begin{aligned} &\{\lfloor r_i/(n-i) \rfloor : 0 \leq r_i < !(n-i-1)\} \\ &\quad = \{0, 1, \ldots, !(n-i)+!(n-i-1)-1\} \end{aligned} \tag{15}$$

を得る．一方，$\sigma(i-1)$ でサイクルを閉じていない場合，(15) 式から帰納的に類推されるように，r_i は $0 \leq r_i < !(n-i+1)+!(n-i)$ の値を取る．(14) 式の条件によって，$\sigma(i)$ でサイクルを閉じるか否かを判定する．$r_i \geq !(n-i+1)$ が成り立つ場合，$\sigma(i)$ でサイクルを閉じ，提案アルゴリズムは，次のステップのランク値 r_{i+1} として，$r_{i+1} = r_i - !(n-i+1)$ を与える．したがって，r_i は

$$!(n-i+1) \leq r_i < !(n-i+1)+!(n-i) \tag{16}$$

の範囲の値を取り，r_{i+1} は，

$$\begin{aligned} &\{r_i - !(n-i+1)\} \\ &\quad = \{0, 1, \ldots, !(n-i)-1\} \end{aligned} \tag{17}$$

を得る．一方，$r_i \geq !(n-i+1)$ が成り立たない場合，$\sigma(i)$ でサイクルを閉じず，次のステップのランク値 r_{i+1} として，$r_{i+1} = r_i/(n-i)$ を与える．したがって，r_i は $0 \leq r_i < !(n-i+1)$ の範囲の値を取り，r_{i+1} は (15) 式と同じ結果を得る．

次に，$\sigma(i)$ と交換すべき要素を選ぶ部分について考える．これは，Myrvold らのアルゴリズムと同様に，剰余算にそのまま置き換えることができる．したがって，$\sigma(i)$ と交換すべき要素 $\sigma(j)$ は，

$$\sigma(j) = \sigma(r \bmod (n-i)+i) \tag{18}$$

となる．

文献 [4] のアルゴリズムから置き換えた計算式は，いずれも定数時間で完了できることは明らかであろう．本節の最後に，次の定理を与える．

定理 4.1 提案アルゴリズムは線形時間で撹乱順列のアンランキングを完了する．

5 まとめ

本報告は，撹乱順列に対するランクとアンランクを線形時間で計算できることを示した．ランダム生成との関係から，他の組合せ的な集合についても，効率的なランキングとアンランキングのアルゴリズムを構成しやすくなりそうである．

謝辞

本研究の一部は，日本学術振興会学術研究助成基金（24500076）を受け実施したものである．

参考文献

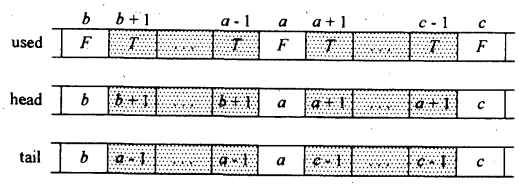
[1] R. Durstenfeld, Algorithm 235: Random permutation, Commun. ACM, vol.7, no.7, pp.420, 1964.

[2] W. Myrvold, F. Ruskey, Ranking and unranking permutations in linear time, Information Processing Letters, vol.79, no.6, pp.281–284, 2001.

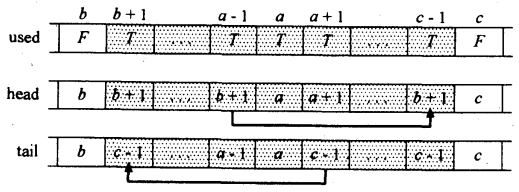
[3] 三河賢治，田中賢，巡回表記で表された撹乱順列に対する辞書順のランキングとアンランキングについて，信学技法，vol.112，no.113，pp.93-96，2012.

[4] 三河賢治，田中賢，撹乱順列の線形時間ランダム生成について，信学技法，vol.112，no.273，pp.53-58，2012.

[5] R.P. Stanley, Enumerative combinatorics, vol.I, Wadsworth & Brooks/Cole Advanced Books, California, 1986.

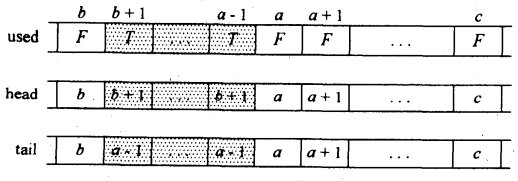


Array values when 'a' is still unused.

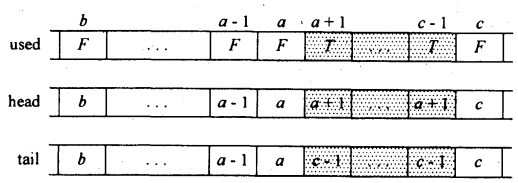


Array values after 'a' is used.

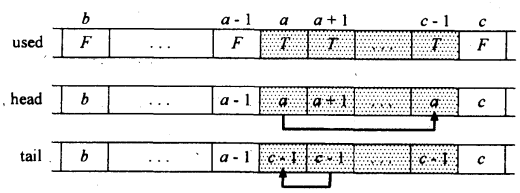
(a) Case of both sides of 'a' are used.



Array values when 'a' is still unused.



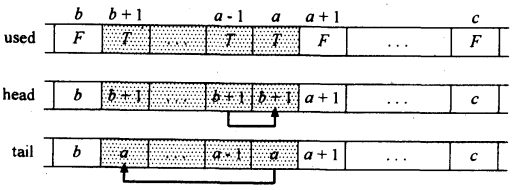
Array values when 'a' is still unused.



Array values after 'a' is used.

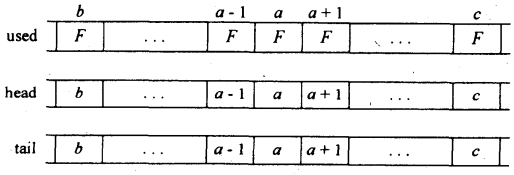
(b) Case of only right side of 'a' is used.

Array values when 'a' is still unused.

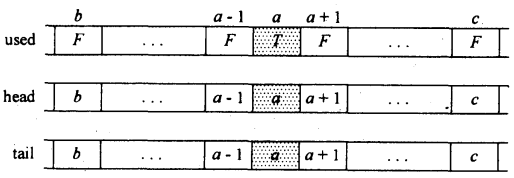


Array values after 'a' is used.

(c) Case of only left side of 'a' is used.



Array values when 'a' is still unused.



Array values after 'a' is used.

(d) Case of neither side of 'a' is used.

図 1: Data structure.