論　文

自然数の組成の $O(1)$ 時間生成について

三河　賢治†

$O(1)$ Time Algorithms for Generating Integer Compositions

Kenji MIKAWA†

あらまし　順序を考慮して自然数 n を適当な自然数の和に分解する問題は自然数の組成と呼ばれる．和を構成する数字を和因子と呼び，次式 $a_1+\cdots+a_r=n$，$a_i \geq 1$，を満たす文字列 $a_1a_2\ldots a_r$ が自然数の組成である．Grimaldi と Meadows は，最大の和因子がたかだか k であるような自然数の組成の個数は k 段フィボナッチ数列に一致することを証明した．本論文では，ある自然数の組成から次の組成を生成する時間が $O(1)$ 時間であるような列挙アルゴリズムを二つ提案する．

キーワード　自然数の組成，k 段フィボナッチ数列，逐次生成，組合せアルゴリズム

1. まえがき

フィボナッチ数列を拡張した k 段フィボナッチ数列 $f(n,k)$ は，境界条件 $f(0,k)=1$ として，

$$f(n,k)=\begin{cases}0 & k=0 \text{ のとき}\\ 1 & k=1 \text{ のとき}\\ \sum_{i=1}^{k} f(n-i,k) & k\leq n \text{ のとき}\\ f(n,n) & k>n \text{ のとき}\end{cases}$$

のように再帰的に定義される．フィボナッチ数列は 2 段フィボナッチ数列と考えることができる．拡張されたフィボナッチ数列の中で，3 段及び 4 段フィボナッチ数列はフィボナッチ数列に因んで，それぞれトリボナッチ数列，テトラナッチ数列と呼ばれることがある．これらの数列は自然界に数多く出現している．例えば，ひまわりの種の配置やオウム貝のらせん模様などにフィボナッチ数列が出現する．

自然数の組成とは，順序を考慮して自然数 n を適当な自然数の和に分解する問題である [7]．例えば，4 の組成は 1+1+1+1，2+1+1，1+2+1，1+1+2，2+2，3+1，1+3，4 の 8 通り存在するが，最大の和因子をたかだか 2 とすると，4 の組成は 1+1+1+1，2+1+1，1+2+1，1+1+2，2+2 の 5 通りとなる．Grimaldi と Meadows [1] は，最大の和因子がたかだか k であるような自然数の組成の総数は k 段フィボナッチ数列に一致することを証明した．これと異なる自然数の組成の定義として，和因子の個数を固定し，和因子 '0' を許す自然数の組成は重複組合せとなり，和因子の個数が 3 であるような自然数 3 の組成は 3+0+0，2+1+0，2+0+1，1+2+0，1+1+1，1+0+2，0+3+0，0+2+1，0+1+2，0+0+3 の 10 通りである．

ある条件を満たす組合せ的な集合の要素を列挙する問題は古くから親しまれており，要素間の相異が最小となるように集合のすべての要素を列挙したものをグレーコードと呼ぶ．組合せ的な集合に対して，(1) グレーコードとなるような列挙は存在するか，グレーコードとなるような列挙が存在するならば (2) 計算機上で一つの要素当り平均 $O(1)$ 時間で列挙することは可能であるか，更に (3) 一つの要素当り $O(1)$ 時間で列挙することは可能であるか，が問題の対象である．和因子の個数を固定した自然数の組成を列挙する問題では，Klingsberg [2] によってグレーコードの存在が示され，個々の和因子の最大値を可変とする一般的な重複組合せ，すなわち和因子の個数を固定した自然数の組成に対するグレーコードは Ruskey と Savage [4] によって示された．しかしながら，和因子の個数を固定しない自然数の組成を列挙する問題では，グレーコード及び個々の組成を $O(1)$ 時間で列挙する方法は提案されていない．本論文では，最大の和因子がたか

† 新潟大学総合情報処理センター，新潟市
Integrated Information Processing Center, Niigata University, Niigata-shi, 950-2181 Japan

だか k であり，かつ和因子の個数を固定しない自然数の組成について，異なるグレーコードを出力する二つのアルゴリズムを提案する．

はじめに提案するアルゴリズムは，Savage [5] による和因子の個数を固定しない自然数の分割に対するグレーコードを自然数の組成に対して拡張したグレーコードを出力する．自然数の組成に対して拡張されたグレーコードの定義を次に与える．文字列を一定の順序で並べたものをリストと呼ぶことがあるが，以下，本論文でも組成を並べたものをリストと呼ぶことにする．このアルゴリズムによって生成されるリストを $L(n,k)$ として，**3.** と **4.** で説明する．

[定義 1.1] 自然数 n の組成のリスト A について，第 i 番目の組成が第 $i-1$ 番目の組成から次式

(1) 和因子の値を 1 増やし，1 より大きい和因子の値を 1 減らす，
(2) 和因子の値を 1 増やし，和因子 '1' を取り除く，
(3) 1 より大きい和因子の値を 1 減らし，和因子 '1' を新しく加える，

のいずれかから得られるとき，リスト A はグレーコードである．

次に提案するアルゴリズムは，組成の表記を以下のように工夫し，各組成の文字列長は入力の自然数の値と等しくなるようにしている．擬似的な和因子 0 を導入して，1 より大きい和因子 a に対し，0 を $a-1$ 個連結して組成を表す．このように組成を表すと，自然数 n の各組成は擬似的な和因子を含めて n 個の和因子からなる．擬似的な和因子は単に組成の表記に工夫を凝らすためだけのもので，生成される組成は，重複組合せ的な組成ではなく，和因子の個数を固定しない自然数の組成に分類されることに注意したい．例えば，4 の組成は 1+1+1+1, 2+0+1+1, 1+2+0+1, 1+1+2+0, 2+0+2+0, 3+0+0+1, 1+3+0+0, 4+0+0+0 のように，各組成は擬似的な和因子を含めて 4 個の和因子からなる．このアルゴリズムは，定義 1.1 の (1) のみを満たすグレーコードを出力する．このアルゴリズムによって生成されるリストを $L'(n,k)$ として，**5.** と **6.** で説明する．

どちらの方法も k を定数とみなしたとき，各組成を $O(1)$ 時間で生成できることを示す．

2. 準　　備

はじめに，組成の表記法を定義する．和因子 a を $\langle a\rangle$ と表し，$\langle a\rangle$ が b 個連続する組成を $\langle a\rangle^b$ と表す．例えば，8 の組成 2+2+2+1+1 は $\langle 2\rangle^3\langle 1\rangle^2$ である．

n 個の組成 $l_1, l_2, \ldots, l_n$ からなるリスト L を $L = (l_1, l_2, \ldots, l_n)$ と表す．リスト L，和因子 $\langle x\rangle$ に対して，$\langle x\rangle \cdot L$ は L に含まれる各組成の先頭に $\langle x\rangle$ を付加する．例えば $L = (\langle 2\rangle\langle 1\rangle, \langle 3\rangle\langle 1\rangle)$ のとき，$\langle 4\rangle \cdot L = (\langle 4\rangle\langle 2\rangle\langle 1\rangle, \langle 4\rangle\langle 3\rangle\langle 1\rangle)$ である．リスト L, L' に対して，$L \circ L'$ は二つのリストを連結する．例えば $L = (\langle 2\rangle\langle 1\rangle, \langle 3\rangle\langle 1\rangle)$, $L' = (\langle 4\rangle\langle 3\rangle, \langle 5\rangle\langle 3\rangle)$ のとき，$L \circ L' = (\langle 2\rangle\langle 1\rangle, \langle 3\rangle\langle 1\rangle, \langle 4\rangle\langle 3\rangle, \langle 5\rangle\langle 3\rangle)$ である．

n 個の組成 $l_1, l_2, \ldots, l_n$ からなるリスト $L = (l_1, l_2, \ldots, l_n)$ に対して，先頭の組成 l_1 を示す $\mathrm{first}(L)$ と末尾の組成 l_n を示す $\mathrm{last}(L)$ を定義する．また，リスト L に含まれる組成の各列を逆順に並べたリスト $\overline{L} = (l_n, \ldots, l_2, l_1)$ を定義し，$\overline{L}$ を L の反転と呼ぶ．明らかに $\mathrm{first}(\overline{L}) = \mathrm{last}(L)$, $\mathrm{last}(\overline{L}) = \mathrm{first}(L)$ が成り立つ．

3. リスト $L(n,k)$ の定義

和因子の総和が n であり，かつ最大の和因子がたかだか k であるような自然数の組成のリスト $L(n,k)$ とおき，λ を空文字列として，$L(n,0) = (\lambda)$, $L(n,1) = (\langle 1\rangle^n)$ のように定義する．$k > n$ について，$L(n,k) = L(n,n)$ のように定義する．一方，$1 < k \le n$ について，

$$\begin{aligned} L(n,k) = {} & \langle 1\rangle \cdot L(n-1,k) \circ \langle 2\rangle \cdot \overline{L(n-2,k)} \\ & \circ \langle 3\rangle \cdot L(n-3,k) \circ \langle 4\rangle \cdot \overline{L(n-4,k)} \\ & \circ \langle 5\rangle \cdot L(n-5,k) \circ \cdots \end{aligned}$$

のように k 個の部分リストの連結として定義する．このとき，$L(n,k)$ の先頭の部分リストが反転しないように k 個の部分リストを交互に反転する．すなわち偶数番目の部分リストは反転し，奇数番目の部分リストは反転しない．例として，リスト $L(6,3)$ の構造を表すグラフ $T(6,3)$ を図 1 に示す．このようなグラフは列挙木と呼ばれ，$T(n,k)$ の根から葉までの経路が $L(n,k)$ の組成に対応する．

組成 $a(1)a(2)\ldots a(r)$ について，先頭の和因子 $a(1)$ の選び方は $\langle 1\rangle, \langle 2\rangle, \ldots, \langle k\rangle$ の k 通りある．$a(1) = \langle i\rangle$ に固定すると，部分列 $a(2)\ldots a(r)$ は部分リスト $L(n-i,k)$ の要素と考えることができる．$a(1) = \langle i\rangle$ を満たす組成を要素とする $L(n,k)$ の部分リスト $\langle i\rangle \cdot L(n-i,k)$ について，$a(2)$ 以下の和因子についても順に値を特定していく．末尾の和因子 $a(r)$ の値を特定したとき，$L(n,k)$ の要素が一つ決定する．し

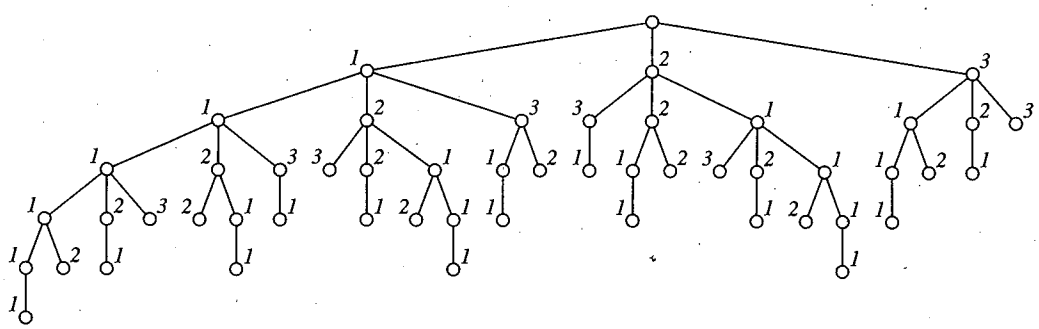

図 1　列挙木 $T(6,3)$
Fig. 1　Enumeration tree $T(6,3)$.

たがって，リスト $L(n,k)$ は，最大の和因子が k であるような自然数 n の組成を系統的に並べている．

[補題 3.1]　リスト $L(n,k)$ の先頭の組成について，

$$\mathrm{first}(L(n,k)) = \langle 1\rangle^n$$

が成立し，末尾の組成について，$n \geq k$ のとき，

$$\mathrm{last}(L(n,k)) = \begin{cases} \langle k\rangle^{\lfloor n/k\rfloor} & k \text{ が奇数かつ } s=0 \text{ のとき} \\ \langle k\rangle^{\lfloor n/k\rfloor}\langle s\rangle & k \text{ が奇数かつ } s>0 \text{ のとき} \\ \langle k\rangle\langle 1\rangle^{n-k} & k \text{ が偶数のとき} \end{cases}$$

が成立する．s は n を k で割ったときの余りである．$n < k$ のときは，$\mathrm{last}(L(n,k)) = \langle n\rangle$ が成り立つ．

(証明)　n に関する帰納法による．　□

[定理 3.1]　リスト $L(n,k)$ はグレーコードである．

(証明)　n に関する帰納法で証明する．明らかに $n=1$ について定理は成り立つ．$n \leq m$ について定理は成り立つと仮定し，$n = m+1$ について考える．奇数 i について，$L(m+1,k)$ を

$$\begin{aligned} L(m+1,k) = \langle 1\rangle \cdot L(m,k) \circ \cdots \\ \circ \langle i-1\rangle \cdot \overline{L(m-i+2,k)} \\ \circ \langle i\rangle \cdot L(m-i+1,k) \\ \circ \langle i+1\rangle \cdot \overline{L(m-i,k)} \circ \cdots \end{aligned}$$

のように $\min(m+1,k)$ 個の部分リストの連結で表す．$\min(m+1,k)$ は $m+1$ と k のうちの小さい方を表す．ただし，i は奇数である．帰納法の仮定により部分リストの内部で定理は成立しているので，各部分リストの境界について定理を示せばよい．補題 3.1 で示しているように，k の奇偶によって部分リストの末尾の文字列の振舞いが異なる．そこで，k が奇数のときと偶数のときの二つの場合に分けて考えていく．

(1) k が奇数である場合．はじめに，部分リスト $\langle i-1\rangle \cdot \overline{L(m-i+2,k)}$ の末尾の文字列と部分リスト $\langle i\rangle \cdot L(m-i+1,k)$ の先頭の文字列との境界で定理は成り立つことを示す．補題 3.1 より，部分リスト $\langle i-1\rangle \cdot \overline{L(m-i+2,k)}$ の末尾の文字列は，

$$\begin{aligned} &\mathrm{last}(\langle i-1\rangle \cdot \overline{L(m-i+2,k)}) \\ &\quad = \mathrm{first}(\langle i-1\rangle \cdot L(m-i+2,k)) \\ &\quad = \langle i-1\rangle\langle 1\rangle^{m-i+2} \end{aligned}$$

である．一方，部分リスト $\langle i\rangle \cdot L(m-i+1,k)$ の先頭の文字列は，

$$\mathrm{first}(\langle i\rangle \cdot L(m-i+1,k)) = \langle i\rangle\langle 1\rangle^{m-i+1}$$

である．部分リスト $\langle i-1\rangle \cdot \overline{L(m-i+2,k)}$ の末尾の文字列の先頭の和因子の値を一つ増やし，末尾の和因子 $\langle 1\rangle$ を取り除くことによって，部分リスト $\langle i\rangle \cdot L(m-i+1,k)$ の先頭の文字列を得る．したがって，定義 1.1 の (2) を満たしている．

次に，部分リスト $\langle i\rangle \cdot L(m-i+1,k)$ の末尾の文字列と部分リスト $\langle i+1\rangle \cdot \overline{L(m-i,k)}$ の先頭の文字列との境界で定理は成り立つことを示す．補題 3.1 より，部分リスト $\langle i\rangle \cdot L(m-i+1,k)$ の末尾の文字列は，

$$\mathrm{last}(\langle i\rangle \cdot L(m-i+1,k)) = \begin{cases} \langle i\rangle\langle k\rangle^{\lfloor m-i+1/k\rfloor} & s=0 \text{ のとき} \\ \langle i\rangle\langle k\rangle^{\lfloor m-i+1/k\rfloor}\langle s\rangle & s>0 \text{ のとき} \end{cases}$$

である．s は $m-i+1$ を k で割った余りである．一方，部分リスト $\langle i+1\rangle \cdot \overline{L(m-i,k)}$ の先頭の文字列は，

$$\begin{aligned}&\mathrm{first}(\langle i+1\rangle\cdot\overline{L(m-i,k)})\\&\quad=\mathrm{last}(\langle i+1\rangle\cdot L(m-i,k))\\&\quad=\begin{cases}\langle i+1\rangle\langle k\rangle^{\lfloor m-i/k\rfloor} & s'=0\text{ のとき}\\ \langle i+1\rangle\langle k\rangle^{\lfloor m-i/k\rfloor}\langle s'\rangle & s'>0\text{ のとき}\end{cases}\end{aligned}$$

である．s' は $m-i$ を k で割った余りである．和因子 $\langle k\rangle$ の個数を $t=\lfloor m-i+1/k\rfloor$，$t'=\lfloor m-i/k\rfloor$ とおき，$s=0$，$s=1$，$s>1$ の三つの場合に分けて両者を比較しよう．

(1A) $s=0$ の場合．$t'=t-1$，$s'=k-1$ が成り立つので，両者の文字列長は等しく，先頭と末尾の文字だけが異なっている．したがって，定義 1.1 の (1) を満たす．

(1B) $s=1$ の場合．$t'=t$，$s'=0$ が成り立つ．したがって，定義 1.1 の (2) を満たす．

(1C) $s>1$ の場合．$t'=t$，$s'=s-1$ が成り立つ．したがって，定義 1.1 の (1) を満たす．

以上，k が奇数である場合，リスト $L(n,k)$ はグレーコードであることが確かめられた．

(2) k が偶数である場合．はじめに，部分リスト $\langle i-1\rangle\cdot\overline{L(m-i+2,k)}$ の末尾の文字列と部分リスト $\langle i\rangle\cdot L(m-i+1,k)$ の先頭の文字列との境界で定理は成り立つことを示す．補題 3.1 より，部分リスト $\langle i-1\rangle\cdot\overline{L(m-i+2,k)}$ の末尾の文字列は，

$$\begin{aligned}&\mathrm{last}(\langle i-1\rangle\cdot\overline{L(m-i+2,k)})\\&\quad=\mathrm{first}(\langle i-1\rangle\cdot L(m-i+2,k))\\&\quad=\langle i-1\rangle\langle 1\rangle^{m-i+2}\end{aligned}$$

である．一方，部分リスト $\langle i\rangle\cdot L(m-i+1,k)$ の先頭の文字列は，

$$\mathrm{first}(\langle i\rangle\cdot L(m-i+1,k))=\langle i\rangle\langle 1\rangle^{m-i+1}$$

である．両者を比較すると，先頭の和因子と文字列長がそれぞれ異なり，定義 1.1 の (2) を満たしている．

次に，部分リスト $\langle i\rangle\cdot L(m-i+1,k)$ の末尾の文字列と部分リスト $\langle i+1\rangle\cdot\overline{L(m-i,k)}$ の先頭の文字列との境界で定理は成り立つことを示す．補題 3.1 より，部分リスト $\langle i\rangle\cdot L(m-i+1,k)$ の末尾の文字列は，

$$\begin{aligned}&\mathrm{last}(\langle i\rangle\cdot L(m-i+1,k))\\&\quad=\begin{cases}\langle i\rangle\langle k\rangle\langle 1\rangle^{m-i+1-k} & m-i+1>k\text{ のとき}\\ \langle i\rangle\langle m-i+1\rangle & m-i+1\le k\text{ のとき}\end{cases}\end{aligned}$$

である．一方，部分リスト $\langle i+1\rangle\cdot\overline{L(m-i,k)}$ の先頭の文字列は，

$$\begin{aligned}&\mathrm{first}(\langle i+1\rangle\cdot\overline{L(m-i,k)})\\&\quad=\mathrm{last}(\langle i+1\rangle\cdot L(m-i,k))\\&\quad=\begin{cases}\langle i+1\rangle\langle k\rangle\langle 1\rangle^{m-i-k} & m-i+1>k\text{ のとき}\\ \langle i+1\rangle\langle m-i\rangle & 2\le m-i+1\le k\text{ のとき}\\ \langle i+1\rangle & m-i+1=1\text{ のとき}\end{cases}\end{aligned}$$

である．両者を比較すると，$m-i+1>k$ の場合は定義 1.1 の (2) を満たし，$m-i+1\le k$ の場合は定義 1.1 の (1) を満たしている．

k が偶数である場合もリスト $L(n,k)$ はグレーコードであることが確かめられたので，$n=m+1$ の場合も定理は成り立つ．ゆえに，定理は証明された． □

4. $L(n,k)$ の生成アルゴリズム

列挙木は，組合せ的な集合の要素の構造を表し，計算機上で実現するデータ構造に適している．列挙木上の各要素を効率良く生成するアルゴリズムは多くの文献で述べられている [6]．本章で提案するアルゴリズムは文献 [3], [8] で述べられている twisted tree の一種である．列挙木 $T(n,k)$ は組成のリスト $L(n,k)$ の構造を反映しているので，$L(n,k)$ の反転 $\overline{L(n,k)}$ に対応する列挙木を $T(n,k)$ の反転と呼ぶことにする．

列挙木 $T(n,k)$ の根から葉までの経路で深さ i に位置する節点を $v(i)$，$v(i)$ のラベルを $a(i)$ として，根 $v(0)$ のラベル $a(0)$ を除いた節点のラベルの列 $a(1)\ldots a(r)$，$r\le n$，が組成である．$T(n,k)$ 上のラベル列 $a=a(1)\ldots a(r)$ とその次列 $a'=a'(1)\ldots a'(r')$ を比較すると，異なるラベルの一方は a が与えられた節点の中で弟をもつもののうち，最も深い節点に付けられ，他方は $a(r)$ 若しくは $a(r+1)$ である．そこで，$v(i)$ から根に向かう先祖の中で，初めて弟をもつ先祖の深さを記憶する $d(i)$ を定義する．$d(i)$ を用いると，a と a' の異なるラベルの一方は $a(d(r))$ となる．

$T(n,k)$ の節点に与えられたラベルに関して，次のことがいえる．節点 $v(i-1)$ を根とする部分木が反転していなければ，$v(i-1)$ の子に与えられるラベルは，先頭の子に 1，次の子に 2，のように順に増加していくが，反転している場合は k または $n-\sum_{j=1}^{i-1}a(j)$ から順に減少していく．各 $x=0,1,\ldots r-1$ に対して，組成に対応する経路上の深さ x の節点 $v(x)$ を根

とする部分木を $T(x)$，$v(x)$ の子に対するラベル付けを $\mathrm{dir}(T(x))$ と表し，

$$\mathrm{dir}(T(x)) = \begin{cases} 1 & \text{ラベル付けが増加のとき} \\ -1 & \text{ラベル付けが減少のとき} \end{cases}$$

とすると，補題 3.1 から次の二つの補題が導かれる．

[補題 4.1] 組成 $a(1)\ldots a(r)$ に対して，$i = d(r)$ とする．このとき，$\mathrm{dir}(T(i)) = \mathrm{dir}(T(i-1))$ であることと $a(i)$ が奇数であることは同値である．

[補題 4.2] 組成 $a(1)\ldots a(r)$ に対して，$i = d(r)$ とする．このとき，$i+1 \le x \le r-1$ を満たす各整数 x に対して，$\mathrm{dir}(T(x)) = \mathrm{dir}(T(i-1))$ であることと $a(i)+a(i+1)$ が偶数であることは同値である．

$T(n,k)$ 上のある組成と次の組成との相異なるラベルの位置は，定理 3.1 の証明の中で述べられている．$T(n,k)$ 上の任意の組成を $a = a(1)\ldots a(r)$，その次列を $a' = a'(1)\ldots a'(r')$，$a(i)$ と $a'(i)$ が与えられている節点をそれぞれ $v(i)$，$v'(i)$ とおく．また，組成 a に対応するラベルの部分和を $s(i) = a(1)+\cdots+a(i)$，次列 a' に対応するラベルの部分和を $s'(i) = a'(1)+\cdots+a'(i)$ とおく．定理 3.1 の証明の中で示されたとおり，$a(i)$ と $a(r)$，a の文字列長 r を制御することにより，a' を求めることができる．はじめに，$\mathrm{dir}(T(i-1)) > 0$ の場合を考える．$a(i)$ が奇数の場合，a' の文字列長 r' に関して，

$$r' = \begin{cases} r-1 & a(r) = 1 \text{ のとき} \\ r & a(r) > 1 \text{ のとき} \end{cases}$$

が成り立ち，先頭と末尾のラベル $a'(i)$ と $a'(r')$ について，それぞれ $a'(i) = a(i)+1$ と

$$a'(r') = \begin{cases} a(r-1) & a(r) = 1 \text{ のとき} \\ a(r)-1 & a(r) > 1 \text{ のとき} \end{cases}$$

が成り立つ．$a(i)$ が偶数の場合，$a'(i) = a(i)+1$，$r' = r-1$ が成り立つ．

一方，$\mathrm{dir}(T(i-1)) < 0$ の場合は，前述の a' から a を求めることに対応する．a' から a を求めるとき，$a(r) > 1$ が成り立つことは，定義 1.1 の (1) の条件，すなわち $a(r) = a'(r)+1$ が生じることの必要十分条件である．上記の必要十分条件を a'，r'，k，i を用いて表すことができれば，$\mathrm{dir}(T(i-1)) < 0$ の場合の a と r の更新方法を決定できるので，$a(r) = 1$ を満たす a' の必要十分条件を求めればよい．$a'(i)$ が奇数または $a'(r') = k$ の場合，明らかに $a(i) = a'(i)-1$，$a(r) = 1$，$r = r'+1$ が成り立つ．定理 3.1 の証明の最後に示された k と $a'(i)$ がともに偶数の場合は注意が必要である．$i < r'-1$ が成り立つとき，$a'(r') = k$，$a(r) = 1$ となるので，定義 1.1 の (2) を満たすが，$i = r'-1$ が成り立つときは定義 1.1 の (1) を満たすからである．上記の議論をまとめて条件式を整理すると，「$i = r'$」，「$a'(i)$ が奇数」，「$a'(r') = k$」，「k が偶数かつ $i < r'-1$」のいずれかが成り立つとき，かつそのときに限り，$a(r) = 1$ となる．

$\mathrm{dir}(T(i-1))$ の値は，前述の補題から，「$r = i$」または「$r = i+1$ かつ $a(i)$ が奇数」または「$r > i+1$ かつ $a(i)+a(i+1)$ が偶数」が成立する場合に真，成立しない場合に偽とする論理変数 equaldir を用意し，

$$\mathrm{dir}(T(i-1)) = \begin{cases} \mathrm{dir}(T(r-1)) & \text{equaldir が真のとき} \\ -\mathrm{dir}(T(r-1)) & \text{equaldir が偽のとき} \end{cases}$$

によって決定する．逆に，$\mathrm{dir}(T(r-1))$ の値は，$\mathrm{dir}(T(i-1))$ の値から得られる．

以上の議論をまとめると，$T(n,k)$ の先頭の組成から始めて最後の組成を生成するまで，以下の処理を繰り返せばよい．

1 $i := d(r)$;
2 $\mathrm{dir}(T(i-1))$ を求める；
3 次列に合わせて r と $a(r)$ の値を求める；
4 $a(i) := a(i) + \mathrm{dir}(T(i-1))$;
5 $\mathrm{dir}(T(r-1))$ を求める；
6 次列に合わせて $d(i)\ldots d(r)$ を更新する；
7 組成 $a(1)\ldots a(r)$ を出力する；

2 行目から 5 行目までは本論文中で議論されたので，最後に $d(i)$ の更新処理について述べる．

定義から $d(i)$ は，$T(i-1)$ 上の節点 $v(i)$ から根に向かう先祖の中で，$v(i)$ 自身を含めて初めて弟をもつ先祖の深さを保持している．したがって，4 行目で $a(i)$ を更新したとき，これは $T(i-1)$ の節点で考えると弟に移動したのだが，この節点が末弟ではない限り，$d(i) = i$ であることは変わらないので，$d(i)$ の値を変更する必要はない．一方，この節点が末弟である場合は，定義から $d(i)$ の値は親が保持する $d(i-1)$ の値と同値となるので，以下の代入手続き $d(i) \leftarrow d(i-1)$ を行い，$d(i)$ を更新する．次に，$d(i)$ を更新した後の

```
 1 program gen(input,output);
 2 var a,d:array[0..100] of integer;
 3 var n,k,r,i,rdir,idir:integer;
 4 procedure out(r:integer);
 5 var i:integer;
 6 begin
 7   for i:=1 to r do write(a[i]:2); writeln
 8 end; { of out }
 9 function equaldir:boolean;
10 begin
11   equaldir:=(r=i) or
        ((r=i+1) and ((a[i] mod 2)<>0)) or
        ((r>i+1) and (((a[i]+a[i+1]) mod 2)=0))
12 end; { of equaldir }
13 begin
14   readln(n); readln(k);
15   for i:=1 to n do begin
16     a[i]:=1; d[i-1]:=i-1
17   end;
18   r:=n; d[n]:=n-1; rdir:=1;
19   out(r);
20   while d[r]>0 do begin
21     i:=d[r];
22     if equaldir then idir:=rdir else idir:=-rdir;
23     if idir>0 then begin
24       if ((a[i] mod 2)=0) or (a[r]=1) then r:=r-1
         else a[r]:=a[r]-1
25     end else begin
26       if (i=r) or ((a[i] mod 2) <>0) or (a[r]=k) or
            (((k mod 2)=0) and (i<r-1)) then begin
27         r:=r+1; a[r]:=1
28       end else
29         a[r]:=a[r]+1
30     end;
31     a[i]:=a[i]+idir;
32     if equaldir then rdir:=idir else rdir:=-idir;
33     if ((idir>0) and ((i=r) or (a[i]=k))) or
           ((idir<0) and (a[i]=1)) then begin
34       d[i]:=d[i-1]; d[i-1]:=i-1
35     end;
36     if r>i then begin
37       if r>i+1 then d[r-1]:=r-1;
38       if (a[r]=1) or (rdir>0) then begin
39         d[r]:=d[r-1]; d[r-1]:=r-1
40       end else
41         d[r]:=r
42     end;
43     out(r)
44   end
45 end.
```

図 2　完全な列挙プログラム
Fig. 2　Complete enumeration program.

$d(i-1)$ のとるべき値について考える．$a(i-1)$ が与えられている節点，すなわち $T(i-1)$ の根が末弟ではない場合，$d(i-1)$ の値を変更する必要はないが，$T(i-1)$ の根が末弟である場合は，将来 $a(i-1)$ が更新されるときに備えて $d(i-1) \leftarrow i-1$ とする．以上，$d(i)$ の更新手続きに関して，次の補題を得る．

[補題 4.3]　ラベル $a(i)$ の更新後，$d(i) \leftarrow d(i-1)$ と $d(i-1) \leftarrow i-1$ の代入手続きを終えたとき，

$$d(x) = \begin{cases} x & x = j, \ldots, i-1 \text{ のとき} \\ j & x = i \text{ のとき} \end{cases}$$

が成立する．j は $a(i)$ のラベルをもつ節点から根に向かう先祖の中で，初めて弟をもつ先祖の深さである．

(証明)　帰納法による．　□

前述の 3 行目で $a(r)$ を更新しているので，上記の議論と同様に $d(r)$ についても更新する．補題 4.3 から，深さ r の節点で $d(r)$ を参照すると，常に目的のラベルをもつ節点の深さを得ることができる．

$d(i)$ の初期値は $d(i) = i$, $i = 0, \ldots, n$, である．また $L(n,k)$ の先頭の組成 $a(1) \ldots a(r)$ は補題 3.1 によって与えられるので，$\langle 1 \rangle^n$, $r = n$, である．$L(n,k)$ を生成する完全な列挙プログラムを図 2 に示す．本章の最後に次の定理を与える．

[定理 4.1]　リスト $L(n,k)$ の各組成を $O(1)$ 時間で生成することができる．

5. リスト $L'(n,k)$ の定義

本章では，擬似的な和因子 $\langle 0 \rangle$ を導入し，$a > 1$ の和因子 $\langle a \rangle$ を $\langle a \rangle \langle 0 \rangle^{a-1}$ のように定義する．例えば，最大の和因子をたかだか 3 であるような自然数 3 の組成は $\langle 1 \rangle\langle 1 \rangle\langle 1 \rangle$, $\langle 1 \rangle\langle 2 \rangle\langle 0 \rangle$, $\langle 2 \rangle\langle 0 \rangle\langle 1 \rangle$, $\langle 3 \rangle\langle 0 \rangle\langle 0 \rangle$ のように，各組成の文字列長が n となり，かつ組成の総数は k 段フィボナッチ数列に一致する．各組成の文字列長を等しくすることにより，列挙アルゴリズムを構築する際，文字列長を制御する部分を省略できる．このような組成のリストに対するグレーコードを次のように定義する．最大の和因子がたかだか k であるような組成のリスト A 上の各組成 $\langle a_1 \rangle \langle a_2 \rangle \ldots \langle a_n \rangle$ と次列 $\langle a'_1 \rangle \langle a'_2 \rangle \ldots \langle a'_n \rangle$ について，(1) $i \neq p, q$ に対して，$a_i = a'_i$, (2) $a_p = a'_p + 1$ かつ $a_q = a'_q - 1$ を満たす n 以下の異なる正整数 p と q が存在するとき，リスト A はグレーコードである．

和因子 $\langle 0 \rangle$ を導入した組成のリストを $L'(n,k)$ とおき，$L'(n,0) = (\lambda)$, $L'(n,1) = (\langle 1 \rangle^n)$ のように定義する．$k > n$ について，$L'(n,k) = L'(n,n)$ のよう

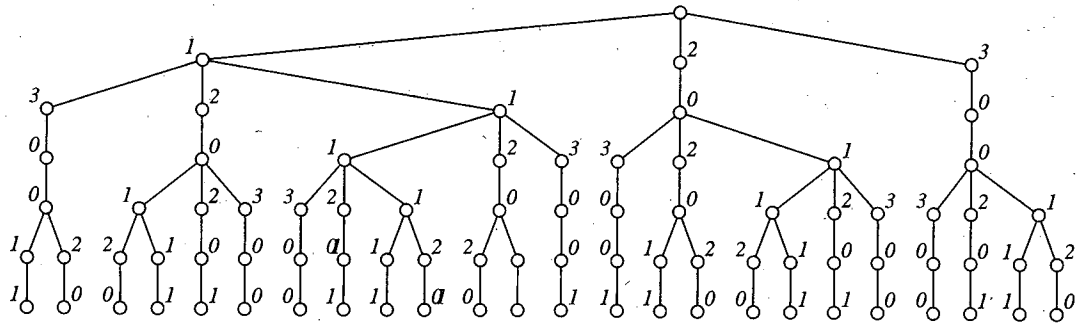

図 3　列挙木 $T'(6,3)$
Fig. 3　Enumeration tree $T'(6,3)$.

に定義する．一方，$1 < k \leq n$ について，

$$\begin{aligned} L'(n,k) = \langle 1\rangle \cdot \overline{L'(n-1,k)} \circ \cdots \\ \circ \langle i\rangle\langle 0\rangle^{i-1} \cdot \overline{L'(n-i,k)} \circ \cdots \\ \circ \langle k\rangle\langle 0\rangle^{k-1} \cdot \overline{L'(n-k,k)} \end{aligned}$$

のように k 個の部分リストの連結として定義する．このとき，$L'(n,k)$ のすべての部分リストが反転するようにリストを連結する．例として，リスト $L'(6,3)$ に対応する列挙木 $T'(6,3)$ を図 3 に示す．

[補題 5.1]　$\mathrm{first}(L'(n,k)) = A_1A_2\ldots A_j$ とすると，$A_i = \langle 1\rangle\langle k\rangle\langle 0\rangle^{k-1}$，$i = 1,2,\ldots,j-1$，が成り立ち，末尾のブロック A_j について，

$$A_j = \begin{cases} \langle 1\rangle\langle k\rangle\langle 0\rangle^{k-1} & s = 0 \text{ のとき} \\ \langle 1\rangle & s = 1 \text{ のとき} \\ \langle 1\rangle\langle 1\rangle & s = 2 \text{ のとき} \\ \langle 1\rangle\langle s-1\rangle\langle 0\rangle^{s-2} & s > 2 \text{ のとき} \end{cases}$$

が成り立つ．s は n を $k+1$ で割ったときの余りである．一方，$\mathrm{last}(L'(n,k)) = B_1B_2\ldots B_j$ とすると，$B_i = \langle k\rangle\langle 0\rangle^{k-1}\langle 1\rangle$，$i = 1,2,\ldots,j-1$ が成り立ち，末尾のブロック B_j について，

$$B_j = \begin{cases} \langle k\rangle\langle 0\rangle^{k-1}\langle 1\rangle & s = 0 \text{ のとき} \\ \langle 1\rangle & s = 1 \text{ のとき} \\ \langle s\rangle\langle 0\rangle^{s-1} & s > 1 \text{ のとき} \end{cases}$$

が成り立つ．s は n を $k+1$ で割ったときの余りである．

(証明)　n に関する帰納法による．　□

[定理 5.1]　リスト $L'(n,k)$ はグレーコードである．

(証明)　n に関する帰納法で証明する．明らかに $n = 1$ について定理は成り立つ．$n \leq m$ について定理は成り立つと仮定し，$n = m+1$ について考える．$i < k$ について，$L'(m+1,k)$ を

$$\begin{aligned} L'(m+1,k) = \langle 1\rangle \cdot \overline{L'(m,k)} \circ \cdots \\ \circ \langle i\rangle\langle 0\rangle^{i-1} \cdot \overline{L'(m-i+1,k)} \\ \circ \langle i+1\rangle\langle 0\rangle^{i} \cdot \overline{L'(m-i,k)} \circ \cdots \\ \circ \langle k\rangle\langle 0\rangle^{k-1} \cdot \overline{L'(m-k+1,k)} \end{aligned}$$

のように k 個の部分リストの連結で表す．帰納法の仮定により部分リストの内部で定理は成立しているので，各部分リストの境界について定理を示せばよい．

部分リスト $\langle i\rangle\langle 0\rangle^{i-1} \cdot \overline{L'(m-i+1,k)}$ の末尾の文字列と部分リスト $\langle i+1\rangle\langle 0\rangle^{i} \cdot \overline{L'(m-i,k)}$ の先頭の文字列との境界で定理は成り立つことを示す．補題 5.1 より，$\langle i\rangle\langle 0\rangle^{i-1} \cdot \overline{L'(m-i+1,k)}$ の末尾の文字列は，

$$\begin{aligned} &\mathrm{last}(\langle i\rangle\langle 0\rangle^{i-1} \cdot \overline{L'(m-i+1,k)}) \\ &\quad = \langle i\rangle\langle 0\rangle^{i-1} \cdot \mathrm{first}(L'(m-i+1,k)) \\ &\quad = \langle i\rangle\langle 0\rangle^{i-1}\langle 1\rangle \cdot \mathrm{last}(L'(m-i,k)) \end{aligned}$$

である．一方，$\langle i+1\rangle\langle 0\rangle^{i} \cdot \overline{L'(m-i,k)}$ の先頭の文字列は，

$$\begin{aligned} &\mathrm{first}(\langle i+1\rangle\langle 0\rangle^{i} \cdot \overline{L'(m-i,k)}) \\ &\quad = \langle i+1\rangle\langle 0\rangle^{i} \cdot \mathrm{last}(L'(m-i,k)) \end{aligned}$$

である．二つの文字列を比較すると，後半の部分列 $\mathrm{last}(L'(m-i,k))$ は等しいので，前半の部分列のみ異なっていることが分かる．$\langle i\rangle\langle 0\rangle^{i-1} \cdot \overline{L'(m-i+1,k)}$ の末尾の文字列の先頭の和因子の値を 1 増やし，$i+1$ 番目の和因子 $\langle 1\rangle$ を $\langle 0\rangle$ に置き換えることによって，$\langle i+1\rangle\langle 0\rangle^{i} \cdot \overline{L'(m-i,k)}$ の先頭の文字列を得る．

したがって，$n = m+1$ の場合も定理は成り立つ．ゆえに，定理は証明された．　□

6. $L'(n,k)$ の生成アルゴリズム

列挙木 $T'(n,k)$ の根から葉までの経路で深さ i に位置する節点を $v(i)$, $v(i)$ のラベルを $a(i)$ として，根 $v(0)$ のラベル $a(0)$ を除いた節点のラベルの列 $a(1)\ldots a(n)$ が組成である．和因子 $\langle 0\rangle$ の導入により，$v(i)$ に与えられるラベル $a(i)$ の最大値は $\min(k, n-i+1)$ となる．ラベル $a(i)>0$ が与えられた節点 $v(i)$ を根とする部分木が反転していなければ，$v(i+a(i)-1)$ の子に与えられるラベル値は，先頭の子に 1，次の子に 2，のように順に増加していくが，反転している場合は k または $n-i+1$ から順に減少していく．深さ i に位置する兄弟間のラベル付けの傾向を記憶する $\mathrm{dir}(i)$ を用意し，ラベル付けが増加である場合は $\mathrm{dir}(i)=1$，減少である場合は $\mathrm{dir}(i)=-1$ とする．また，$v(i)$ から根に向かう先祖の中で，初めて弟をもつ先祖の深さを記憶する $d(i)$ を定義する．

前章では，$d(i)$ を用いて $T(n,k)$ 上の組成を生成するアルゴリズムを提案したが，$d(i)$ の基本的な動作は，部分木内部の組成を出力するごとに経路上の親から子へ $d(i)$ の値を葉まで下ろすことである．例えば，節点 $v(j)$ から根に向かう先祖 $v(j-1),\ldots,v(0)$ 及び $v(j)$ 自身のうち，初めて弟をもつ先祖を $v(j')$, $j'\le j$, として，組成を生成する過程を考える．$v(j)$ が末弟の場合，$v(j)$ を根とする部分木内部の組成の生成を完了する以前に，$v(j'+1)$ の兄らを根とする部分木から順に $v(j-1)$ の兄らを根とする部分木の内部の組成の生成を完了している．したがって，各先祖の直前の兄を根とする部分木内部の組成の生成を完了するごとに，代入式 $d(i)\leftarrow d(i-1)$ が $i=j'+1,\ldots,j$ の順に実施されるので，$v(j)$ で $d(j)$ を参照すると，目的の先祖の深さ j' を得ることができる．ところが，本章で扱う列挙木 $T'(n,k)$ では，経路上に $\langle 0\rangle$ が与えられた先祖が存在する場合があるので，この先祖で $d(j')$ の値を下ろす動作が途切れてしまう．この結果，$v(j)$ で $d(j)$ を参照しても，目的の先祖の深さを得ることはできない．そこで，組成のラベル列 $a(1)\ldots a(n)$ の各ラベル $a(i)>0$ に対して，$\mathrm{up}(i+a(i)-1)=i$ を定義する．この指標 up を用いて，次式

$$d(i)\leftarrow\begin{cases} d(\mathrm{up}(i-1)) & a(i-1)=0 \text{ のとき}\\ d(i-1) & a(i-1)>0 \text{ のとき}\end{cases} \tag{1}$$

のように $d(i)$ を更新すると，経路上の $\langle 0\rangle$ が与えられた先祖を飛び越えて，目的の先祖の深さを $d(i)$ に得ることができる．

実際に $T'(n,k)$ 上で組成を生成する過程を見ていく．本章で提案するアルゴリズムでは，深さ $n-1$ の節点をアルゴリズムによってたどられる最も深い節点としている．$T'(n,k)$ 上の組成を $a=a(1)\ldots a(n)$, その次列を $a'=a'(1)\ldots a'(n)$, $a(i)$ と $a'(i)$ が与えられている節点をそれぞれ $v(i)$, $v'(i)$ とおく．また，前述の dir, up, d は組成 a に対応するように既に更新されているものとする．組成 a を出力した後，次に更新されるラベルの一方は次式

$$i\leftarrow\begin{cases} d(\mathrm{up}(n-1)) & a(n-1)=0 \text{ のとき}\\ d(n-1) & a(n-1)>0 \text{ のとき}\end{cases}$$

から $a(i)$ を得る．擬似的な和因子 $\langle 0\rangle$ を導入したことにより，更新されるラベルは，$\mathrm{dir}(i)>0$ ならば，$a(i)$ と $a(i+a(i))$, $\mathrm{dir}(i)<0$ ならば，$a(i)$ と $a(i+a(i)-1)$, のように決定できるので，これらのラベルを更新して生成された a' を出力する．このとき，$a'(i)$ に合わせて $\mathrm{up}(i+a'(i)-1)$ を i に変更する．ただし，$i+a'(i)-1>n-1$ ならば $\mathrm{up}(n-1)$ を i に変更する．また，$a'(i)$ に合わせて，$d(n-1)$ の値も $a(n-1)=0$ ならば $d(n-1)\leftarrow\mathrm{up}(n-1)$, $a(n-1)>0$ ならば $d(n-1)=n-1$ のように変更する．$v(i)$ と $v'(i)$ は兄弟なので，$\mathrm{dir}(i)$ を更新する必要はない．

次に，a' を出力した後，残りの組成の生成について述べる．$v'(i)$ が弟をもつ場合，前述のとおり dir, up, d はそれぞれ a' に対応するように更新されているので，a から a' を生成したときと同様に，a' から次の組成を生成すればよい．一方，$v'(i)$ が弟をもたない場合は，式 (1) による $d(i)$ の更新と，次式

$$\mathrm{dir}(i)=\begin{cases} 1 & \mathrm{dir}(i)<0 \text{ のとき}\\ -1 & \mathrm{dir}(i)>0 \text{ のとき}\end{cases}$$

よる $\mathrm{dir}(i)$ の更新を行う．$d(i)$ の更新について，up と d の値は a に対応するように更新されていると仮定しているので，$d(i-1)$ 若しくは $d(\mathrm{up}(i-1))$ は a' を生成する以前に $v(i-1)$ から根に向かう先祖のうち，初めて弟をもつ先祖の深さを保持している．したがって，式 (1) によって $d(i)$ を更新すると，$v'(i)$ から根に向かう先祖のうち，初めて弟をもつ先祖の深さ

```
 1 program gen(input,output);
 2 var a,d,dir,up:array [0..100] of integer;
 3 var n,k,i,j:integer;
 4 procedure out;
 5 var i:integer;
 6 begin
 7 for i:=1 to n do write(a[i]:2); writeln
 8 end;
 9 function min(a,b:integer):integer;
10 begin
11   if a<b then min:=a else min:=b
12 end;
13 begin
14   readln(n); readln(k);
15   for i:=0 to n+1 do begin
16     d[i]:=i; dir[i]:=1; up[i]:=i
17   end;
18   i:=1;
19   while i<=n do begin
20     if a[i-1]=0 then a[i]:=1;
21     if a[i-1]=1 then begin
22       a[i]:=min(k,n-i+1);
23       dir[i]:=-1;
24       up[min(i+k-1,n-1)]:=i
25     end;
26     i:=i+a[i]
27   end;
28   out;
29   if a[n-1]<>0 then i:=d[n-1] else i:=d[up[n-1]];
30   while i>0 do begin
31     if dir[i]>0 then begin
32       a[i+a[i]]:=0; a[i]:=a[i]+1
33     end else begin
34       a[i+a[i]-1]:=1; a[i]:=a[i]-1
35     end;
36     out;
37     up[min(i+a[i]-1,n-1)]:=i;
38     if a[n-1]<>0 then d[n-1]:=n-1
                 else d[up[n-1]]:=up[n-1];
39     if ((dir[i]<0) and (a[i]=1)) or ((dir[i]>0) and
       (a[i]=min(k,n-i+1))) then begin
40       dir[i]:=-dir[i];
41       if a[i-1]<>0 then j:=i-1 else j:=up[i-1];
42       d[i]:=d[j]; d[j]:=j
43     end;
44     if a[n-1]<>0 then i:=d[n-1] else i:=d[up[n-1]]
45   end;
46 end.
```

図 4　完全な列挙プログラム
Fig. 4　Complete enumeration program.

が $d(i)$ に代入される．

以上，組成 a から a' を生成，出力した後，$v'(i)$ が弟をもつ場合の生成過程，弟をもたない場合の生成過程を述べた．$v(i)$，$v''(i)$ を根とする部分木の内部でも，上記の一連の操作を行い，組成 a'' を出力した後，$d(n-1)$ または $d(\mathrm{up}(n-1))$ を参照すれば，a'' と b の異なるラベルのうちの一方をもつ節点の深さを得ることができる．$T'(n,k)$ の最後尾の組成を出力すると，$d(n-1)=0$ となって根の深さが代入される．このとき，組成の生成を終了する．$L'(n,k)$ を生成する完全な列挙プログラムを図 4 に示す．本章の最後に次の定理を与える．

[定理 6.1]　リスト $L'(n,k)$ の各組成を $O(1)$ 時間で生成することができる．

謝辞　有益な御意見を頂いた査読者の方々に深く感謝致します．

文　献

[1] R.P. Grimaldi and D.S. Meadows, "Compositions of integers," Congre. Numer., vol.76, pp.119–125, 1990.

[2] P. Klingsberg, "A Gray code for compositions," J. Algorithms, vol.3, pp.41–44, 1982.

[3] 三河賢治，仙波一郎，"フィボナッチ列の $O(1)$ 時間生成について，" 信学論 (D-I)，vol.J85-D-I, no.2, pp.116–121, Feb. 2002.

[4] F. Ruskey and C.D. Savage, "A Gray code for combinations of multiset," Eur. J. Combinatorics, vol.17, pp.493–500, 1996.

[5] C.D. Savage, "Gray code sequences of partitions," J. Algorithms, vol.10, pp.577–595, 1989.

[6] C.D. Savage, "A survey of combinatorial Gray codes," SIAM Rev., vol.39, pp.605–629, 1997.

[7] S. Skiena, "Compositions," in Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, pp.60–62, Addison-Wesley, Redwood City, CA, 1990.

[8] T. Takaoka, "$O(1)$ time algorithms for combinatorial generation by tree traversal," Comput. J., vol.42, pp.400–408, 1999.

(平成 15 年 12 月 14 日受付，16 年 7 月 14 日再受付)

三河　賢治　(正員)

1995 茨城大・工卒．1997 同大大学院博士前期課程了．2001 同博士後期課程了．2002 茨城大学総合情報処理センター講師，2003 新潟大学総合情報処理センター助手．組合せアルゴリズムの設計と解析の研究に従事．工博．