列挙の基本と基礎的なアルゴリズム

Enumeration Fundamentals and Basic Algorithms

岡本吉央

Abstract

列挙問題とは，与えられた条件を満たすものを漏れなく，重複なく出力する問題である．本稿では，列挙問題を解くためのアルゴリズム設計技法として基礎的なものを紹介する．まず，列挙アルゴリズムの効率の良さの測り方を導入する．その後で，部分集合列挙問題を例題として，分割法，グレイコード，逆探索法という三つの設計技法を紹介する．最後に，効率の良い列挙アルゴリズムを設計することが難しい列挙問題とその理由について言及する．

キーワード：列挙アルゴリズム，遅延，分割法，グレイコード，逆探索法

1. 列挙アルゴリズム

列挙問題（enumeration problem）とは，与えられた条件を満たすものを漏れなく，重複なく出力する問題である．出力順に条件を課す場合もある．例えば，「$\{1,2,3,4,5\}$ の部分集合で，その和が 6 となるものを列挙せよ」という問題の出力は $\{1,2,3\}$，$\{1,5\}$，$\{2,4\}$ である．また，「『コアラ』という文字列を含む Web ページを PageRank の大きな順に 10 個出力せよ」という問題も列挙問題である．

列挙アルゴリズム（enumeration algorithm）とは列挙問題を解くアルゴリズムである．列挙問題に限らず，アルゴリズム設計を行うときに注意すべき項目が二つある．一つは正当性である．すなわち，アルゴリズムが満たすべき性質を保証することである．二つ目は効率性である．すなわち，アルゴリズムの費やす資源量（時間，空間）を見積もることで，その効率を評価することである．

列挙アルゴリズムにおける正当性は出力すべき対象を全て，漏れなく，重複なく出力することである．出力順に関する条件がある場合はそれを満たすことも要求される．例に挙げた「$\{1,2,3,4,5\}$ の部分集合で，その和が 6 となるものを列挙せよ」という問題を考えよう．次のアルゴリズムは自明なものであるが，その正当性は直ちに分かる．

- ステップ 1　$\{1,2,3,4,5\}$ の全ての部分集合に対して，その和を計算する．
- ステップ 2　和が 6 であれば，その部分集合を出力する．そうでなければ出力しない．

しかし，このアルゴリズムの効率は良くない．ステップ 1 では $\{1,2,3,4,5\}$ の部分集合を全て扱っていて，それは $2^5=32$ 個ある．その一方，出力は 3 個しかない．

よって，正当性を満たしつつ，効率の良いアルゴリズムをどのように設計するのか，ということが列挙アルゴリズム設計における要点である．

列挙アルゴリズムの効率を測るための尺度を導入する．列挙アルゴリズムに対する入力のサイズを n，そのときの出力の数を N とする．理論的に，効率の良いアルゴリズムは「n に関する多項式時間アルゴリズム」として形式化されることが多いが，列挙アルゴリズムは出力を漏れなく行うため，その計算量が N を下回ること

本会誌では，①文部省（文部科学省）学術用語集電気工学編，②本会編の改訂電子情報通信用語辞典，③本会編のエンサイクロペディアハンドブックに基づき用語を統一している．
本稿中の「同型」は上記②に従うと「同形」であるが，著者の希望により「同型」で掲載した．

岡本吉央　正員　電気通信大学大学院情報理工学研究科情報・通信工学専攻
E-mail　okamotoy@uec.ac.jp
Yoshio OKAMOTO, Member (Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo, 182-8585 Japan).
電子情報通信学会誌　Vol.95 No.6 pp.477-483 2012 年 6 月

はできない．そして，出力の数Nが入力サイズnの多項式で抑えられる保証はないので，時間計算量がnに関する多項式で抑えられる保証はない．したがって，計算量はnとNに依存する形で算定する．

- 出力多項式時間列挙アルゴリズム（output-polynomial enumeration algorithm）とは，時間計算量がnとNに関する多項式で抑えられる列挙アルゴリズムである．
- ならし多項式時間遅延列挙アルゴリズム（amortized polynomial-time-delay algorithm）とは，時間計算量がnに関する多項式，Nに関する一次式で抑えられる列挙アルゴリズムである．
- 最悪時多項式時間遅延列挙アルゴリズム（worst-case polynomial-time-delay algorithm）とは，次の三つがどれもnに関する多項式で抑えられる列挙アルゴリズムである．

① アルゴリズムの実行開始から一つ目の出力が得られるまでの時間．
② 任意の$i\in\{1,\ldots,N-1\}$に対して，i番目の出力が得られてから，$i+1$番目の出力が得られるまでの時間．（この時間は列挙アルゴリズムの遅延（delay）と呼ばれる．）
③ N番目の出力が得られてから，アルゴリズムが停止するまでの時間．

例えば，$O(N^2n^3)$という時間計算量を持つ列挙アルゴリズムは出力多項式時間であるが，ならし多項式時間遅延とは限らない．そして，$O(Nn^4)$という時間計算量を持つ列挙アルゴリズムは出力多項式時間であり，なおかつ，ならし多項式時間遅延でもある．簡単に分かることであるが，最悪時多項式時間遅延アルゴリズムはならし多項式時間遅延アルゴリズムであり，ならし多項式時間遅延アルゴリズムは出力多項式時間アルゴリズムである．

出力多項式時間列挙アルゴリズムは「効率の良い」列挙アルゴリズムであると考えられるが，それを得ることは容易であることが多い．そのため，列挙アルゴリズム設計の研究では，ならし多項式時間遅延アルゴリズムや最悪時多項式時間遅延アルゴリズムを得ることが主眼になる．

空間計算量に関して効率の良いアルゴリズムは多項式空間という概念で形式化される．すなわち，多項式空間列挙アルゴリズム（polynomial-space enumeration algorithm）とは，使用作業領域がnの多項式で抑えられるものである．列挙アルゴリズム設計においては，出力した対象を作業領域に保持するか，あるいは保持せずに捨ててしまうか，ということを設計者が定める必要があるので，そこに注意する．出力を作業領域に保持すれば，それをアルゴリズムの中で再利用できるが，そうでなければできない．例えば，アルゴリズムの実行中にある対象を出力しようとしたとき，それが既に出力されたものであるかどうかを確認して，重複を回避することは，今までに出力された対象を全て作業領域に保持しておけば，一つ一つ照合することで可能となる．しかし，こうすると，使用作業領域がnの多項式で抑えられるとは限らなくなる．そのため，多項式空間列挙アルゴリズムを構築するときには，出力の重複を省くために上記のような単純な戦略を用いることができなくなるという重要な点を押さえる必要がある．

本稿では以下，次の項目を順次説明する．まず，効率良い列挙アルゴリズム設計法を紹介するために，例題として部分集合列挙問題を紹介する．その問題に対して，分割法，グレイコード，逆探索法という三つの手法を適用する．これらはどれも，ならし多項式時間遅延多項式空間列挙アルゴリズムである．続く章では，これらの手法の得失を論じる．その次の章では，列挙問題の難しさについて論じる．

2. 部分集合列挙問題

部分集合列挙問題とは，自然数nが与えられたとき，$\{1,2,\cdots,n\}$の部分集合を全て出力する問題である．例えば，$n=3$のとき，出力する対象は$\{1,2,3\}$の部分集合全てであり，それは$\emptyset$，$\{1\}$，$\{2\}$，$\{3\}$，$\{1,2\}$，$\{1,3\}$，$\{2,3\}$，$\{1,2,3\}$である．これは最も基本的な列挙問題の一つであり，設計法の基礎を説明するためにも適している．以降，集合$\{1,2,\cdots,n\}$を単に$[n]$という記号で表す．

また，細かいことであるが，入力nは1進表現で与えられるとして，すなわち，$O(n)$という計算量は入力サイズの多項式であるとみなす．

3. 分割法による列挙

分割法（binary partition method）の基本的な考え方は場合分けである．集合$[n]$の任意の部分集合は次の二つのどちらか一方だけを満たす．

① nを要素として持たない．
② nを要素として持つ．

これは当然のことであるが，この観察が次のアルゴリズム設計指針を与える．すなわち，nを要素として持たない部分集合を全て出力し，nを要素として持つ部分集合を全て出力すれば，列挙が完了するのである．そのため，「nを要素として持たない部分集合の列挙」と「nを要素として持つ部分集合の列挙」という部分問題をど

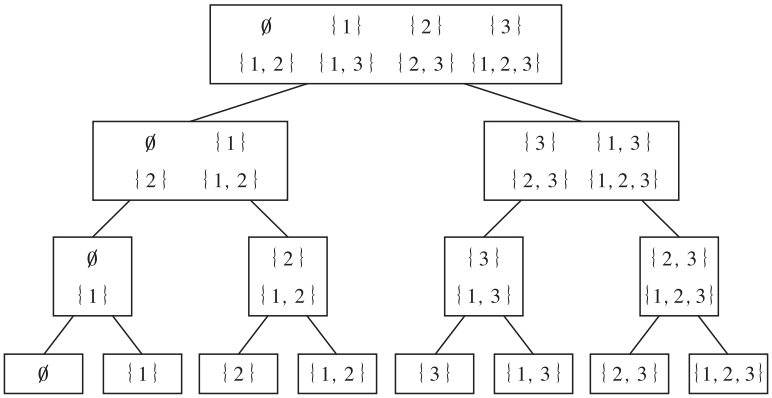

図1　分割法による部分集合列挙（$n=3$）

う解けばよいか考えればよい．

まず，「n を要素として持たない部分集合の列挙」であるが，n を要素として持たない部分集合はすなわち $[n-1]$ の部分集合である．すなわち，問題に再帰的な構造があることが分かる．一方，「n を要素として持つ部分集合の列挙」であるが，n を要素として持つ部分集合から n を取り除くと $[n-1]$ の部分集合が得られ，逆に，$[n-1]$ の部分集合に n を追加すると n を要素として持つ部分集合が得られる．すなわち，$[n-1]$ の部分集合が列挙できればよい．

この再帰構造から次の再帰アルゴリズムが得られる．

部分集合列挙アルゴリズム（分割法）
入力　自然数 n
出力　$[n]$ の部分集合全て
ステップ1　$A(\emptyset, n)$ を呼び出して停止

アルゴリズム　$A(X, i)$
入力　自然数 $i \in \{0, \cdots, n\}$，集合 $X \subseteq \{i+1, \cdots, n\}$
出力　$\{X \cup Y \mid Y \subseteq \{1, \cdots, i\}\}$ の要素全て
ステップ1　$i=0$ ならば，X を出力して停止
ステップ2　$i>0$ ならば，$A(X, i-1)$ と $A(X \cup \{i\}, i-1)$ を呼び出して停止

再帰呼出しが作る再帰計算木と，それに対応する分割の例を図1に示す．

アルゴリズム A の正当性は上の議論から直ちに分かる．空間計算量はどうだろうか？　これは再帰アルゴリズムであるが，呼出し $A(X, i)$ において，再帰の深さは i である．再帰計算木の各ノードで保持すべき情報はそのノードから根に戻るパスとアルゴリズムを呼び出した際の引数である．これらは $O(i)$ 領域しか必要としない．すなわち，使用作業領域は $O(n)$ である．時間計算量はどうだろうか？　アルゴリズムの呼出し $A(X, i)$ は再帰計算木の葉において出力を必ず行い，再帰計算木の内部ノードは必ず二つ子を持つ．そのため，再帰計算木の辺の数は再帰計算木の葉の数に比例し，それは出力の数に比例する．一つの部分集合を出力すること自体に $O(n)$ 時間かかるため，$A(\emptyset, n)$ の時間計算量は $O(Nn)$ となる．よって，このアルゴリズムはならし線形時間遅延線形空間アルゴリズムである．

分割法では，列挙すべき対象全体の集合を仮想的に分割して，小さくなった問題を再帰的に解く．問題の再帰構造や部分問題の解きやすさがアルゴリズム設計の鍵となる．

4. グレイコードによる列挙

グレイコード（Gray code）による列挙では，対象を局所的に変更することを繰り返し，出力を次々に得る．部分集合列挙問題において，集合 $X \subseteq [n]$ に対する局所的変更として，1要素の追加と1要素の除去を考える．すなわち，二つの集合 $X, Y \subseteq [n]$ に対して，X から Y が局所的変更によって得られるとき，そのときに限って，$|X \triangle Y|=1$ である．ただし，$X \triangle Y = (X \cup Y) \setminus (X \cap Y)$ は対称差を表す．集合 X から Y が局所的変更によって得られるとき，Y から X も局所的変更によって得られることに注意する．

この局所的変更を用いて，無向グラフを定義できる．頂点集合は列挙するべき $[n]$ の部分集合全体であり，二つの部分集合 X, Y の間に辺があることを，X から Y が局所的変更によって得られること（すなわち，$|X \triangle Y|=1$ であること）とする．上記注意より，これを無向グラフとみなしてよいことが分かる．図2に $n=3$ のときの例を示す．このグラフを Q_n と表記することにする．

グレイコードによる列挙では，Q_n のハミルトン道に沿って出力すべき対象を一つずつ見つける．（復習：グラフのハミルトン道とは，その頂点をちょうど一度ずつ訪問する道のことである．）すなわち，ハミルトン道の

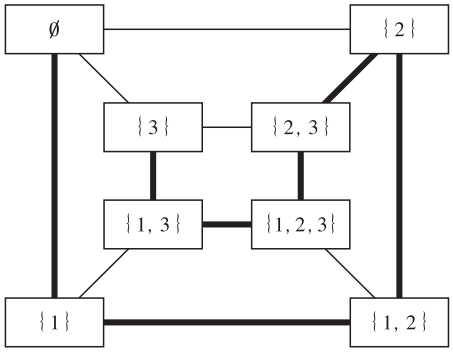

図2　グレイコードによる部分集合列挙（$n=3$）

訪れた頂点に対応する部分集合を順に列挙するのである．図2では，∅から始まるハミルトン道を太線で示している．

この列挙を効率良く行うためには，確認すべきことが二つある．

① 実際に，Q_nがハミルトン道を持つこと．
② そのハミルトン道に沿って列挙をする際，Xという部分集合を出力した後に出力すべき部分集合を効率良く見つける方法を与えること．

まず，Q_nが∅を始点として$\{n\}$を終点とするハミルトン道を持つことをnに関する帰納法で証明しよう．$n=1$のときは簡単に正しいと分かるので，$n>1$とする．このとき，nを要素として含まない部分集合全体が誘導するQ_nの部分グラフはQ_{n-1}であるので，帰納法の仮定より，その部分グラフには∅を始点として$\{n-1\}$を終点とするハミルトン道P_1が存在する．また，nを要素として含む部分集合全体が誘導するQ_nの部分グラフはQ_{n-1}と同型である．なぜなら，そのような部分集合からnを除去するとnを要素として含まない部分集合が得られ，nを要素として含む部分集合X, Yの間に辺があるとき，そのときに限り，$X\backslash\{n\}$と$Y\backslash\{n\}$の間に辺があるからである．すなわち，再び帰納法の仮定より，この部分グラフには$\{n\}$を始点として$\{n-1, n\}$を終点とするハミルトン道P_2が存在する．

いま，グラフQ_nにおいてP_1とP_2という道が得られたが，これらを結ぶことでQ_nのハミルトン道を構成してみる．すなわち，∅を始点とし，P_1に沿って$\{n-1\}$まで歩く．そして，$\{n-1\}$と$\{n-1, n\}$は辺で結ばれているので，その辺に沿って，$\{n-1\}$から$\{n-1,n\}$に移動する．そして，P_2を逆にたどって$\{n-1, n\}$から$\{n\}$に移動する．これで，∅を始点として$\{n\}$を終点とするQ_nのハミルトン道が得られた．図2のハミルトン道もこの手順によって得られたものである．これで証明が完了した．

では，二つ目の確認事項を見よう．上の証明をじっくり眺めると，ある観察が得られる．すなわち，上の証明で構成したQ_nのハミルトン道において，集合Xの次に訪れられる集合Yは次の公式で与えられる．Xの要素数が偶数の場合，$Y=X\triangle\{1\}$であり，Xの要素数が奇数の場合，$Y=X\triangle\{1+\min X\}$である．

上の観察に基づいて，次の列挙アルゴリズムが得られる．

部分集合列挙アルゴリズム（グレイコード）
入力　自然数n
出力　$[n]$の部分集合全て
初期化　$X:=\emptyset$
反復　$X=\{n\}$を出力するまで，以下を繰返し
ステップ1　Xを出力
ステップ2　Xの要素数が偶数ならば，$X:=X\triangle\{1\}$
ステップ3　Xの要素数が奇数ならば，$X:=X\triangle\{1+\min X\}$

今までの議論から，このアルゴリズムの正当性が導かれる．空間計算量はどうだろうか？　作業領域に保持するものはXのみであり，$X\subseteq[n]$なので，使用作業領域は$O(n)$である．時間計算量はどうだろうか？　反復回数は出力の数Nである．各反復において，Xの要素数の偶奇と$\min X$を計算する必要があるが，これは簡単にできる（詳細は省略）．そのため，各反復における時間計算量はステップ1の「Xを出力」が支配し，$O(n)$となる．すなわち，このアルゴリズムは最悪時線形時間遅延線形空間列挙アルゴリズムである．

このアルゴリズムでは，連続する二つの出力の違いは一つの要素の追加または除去のみである．そのため，その出力の違いのみを出力するという「差分出力」を行うと，各反復における時間計算量は$O(1)$となる．

グレイコードという名称はFrank Grayにちなんでいるが，グレイコードによる部分集合列挙を最初に考案したのが誰であるのかについては議論がある．例えば，Knuthの本[(1)]を参照のこと．その後，他の組合せ的対象に対してもそれに似た手法の列挙アルゴリズムが提案され，それらは「組合せ的グレイコード（combinatorial Gray code）」と呼ばれた．組合せ的グレイコードについてはSavage[(2)]によるサーベイが詳しい．本稿では，組合せ的グレイコードも区別せず「グレイコード」と呼ぶ．

一般に，グレイコードによる列挙アルゴリズムでは，列挙すべき対象全体の上にグラフを定義し，そのグラフのハミルトン道をたどることで列挙を遂行する．上で行った二つの観察に類似する命題を証明する必要があるため，その設計は難しいが，計算量の低いアルゴリズムが得られるため，実用上好まれる．

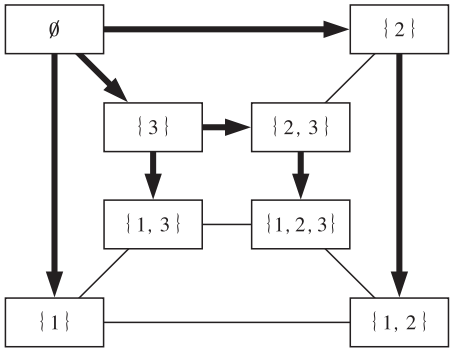

図3　逆探索法による部分集合列挙（$n=3$）　矢印の始点が親，終点が子を表す．

5. 逆探索法による列挙

グレイコードによる列挙アルゴリズムでは，グラフ上にハミルトン道を発見する必要があり，これは容易なことではない．一般に，グラフがハミルトン道を持つとは限らず，もし持っているとしても，それを発見することは難しいかもしれない．

その一方で，今から紹介する逆探索法（reverse search method）はグラフ上の全域根付き木をたどる．連結な無向グラフには必ず全域木が存在するため，存在性に関する懸念は簡単に解消される．アルゴリズム設計の要点は上手にたどれる全域根付き木を見つけることである．

再び，前章で導入した無向グラフ Q_n を考える．すなわち，頂点集合は列挙するべき $[n]$ の部分集合全体であり，二つの部分集合 X, Y の間に辺があることを，$|X \triangle Y|=1$ であることとする．

逆探索法に基づく列挙アルゴリズムでは，Q_n 上の全域根付き木をたどる．ここでは次の全域根付き木 T を定義する．根付き木 T の根は $\emptyset$ とする．根でない頂点 X の親を $X\backslash\min X$ と定義する．例えば，$X=\{1,3\}$ のとき，$\min X=1$ なので，X の親は $\{3\}$ になる．図3に $n=3$ のときの例を示す．

逆探索法では，この根付き木 T の上で深さ優先探索を行い，その訪問順（すなわち，深さ優先探索順）に列挙すべき対象を出力する．それを効率良く行うためには，次の二つが効率良く行えればよい．

根発見　T の根を効率良く発見すること．
子発見　T の頂点 X が与えられたとき，X の子を効率良く列挙すること．（子が存在しない場合は，そうであることを効率良く判定すること．）

この二つが行えれば，T 自身を知ることなく T 上の深さ優先探索が行えるようになる．

部分集合列挙問題に関して，根は $\emptyset$ であるので，根発見は簡単に行える．したがって，鍵となるのは子発見である．集合 X の親は $X\backslash\min X$ であるので，集合 Y の子 Z は $Y=Z\backslash\min Z$ を満たすもの全体である．すなわち，Y の子は，$j<\min Y$ を満たす任意の j に対して $Y\cup\{j\}$ と書ける集合全体である．ただし，便宜上 $\min\emptyset=\infty$ と定めておく．

以上の手続きを基にして，実際，次のアルゴリズムを設計できる．

部分集合列挙アルゴリズム（逆探索法）
入力　自然数 n
出力　$[n]$ の部分集合全て
ステップ1　$B(\emptyset)$ を呼び出して停止

アルゴリズム　$B(X)$
入力　集合 $X\subseteq[n]$
出力　Q_n 上の根付き木 T における X の子孫全体（X も含む）
ステップ1　X を出力
ステップ2　$i:=\min X$ とする
ステップ3　$i=1$ ならば停止
ステップ4　そうでなければ，$j:=1$ として以下を繰返し
ステップ4-1　$j=i$ または $j>n$ ならば停止
ステップ4-2　そうでなければ $B(X\cup\{j\})$ を呼出し
ステップ4-3　$j:=j+1$

このアルゴリズムの正当性は今までの議論から分かる．空間計算量はどうだろうか？　これは再帰アルゴリズムであるが，再帰計算木の深さは高々 n である．再帰計算木の各ノードでは X を保持する必要があり，そのために $O(n)$ の空間を用いる．したがって，空間計算量は $O(n)$ である．時間計算量はどうだろうか？　無向グラフ Q_n の頂点数は N であるので，木 T の辺数は $N-1$ である．すなわち，再帰計算木の辺数は $N-1$ である．再帰計算木の各ノードにおいて出力をするので，総計算時間は $O(nN)$ となる．したがって，このアルゴリズムはならし線形時間遅延線形空間列挙アルゴリズムである．より詳細な解析を行えば，最悪時線形時間遅延アルゴリズムであることも分かるが，省略する．

上のアルゴリズムは再帰の形式で記述されているが，再帰を用いない記述も可能である．しかし，その際には，根発見，子発見という手続きの他に，「親発見」という手続きも必要となる．

一般に，逆探索法では，列挙すべき対象全体の上のグラフを考え，その全域根付き木を深さ優先順にたどることで列挙を遂行する．重要な点は全域根付き木を上手に定義することと，その上での根発見，子発見，親発見と

いう手続きを効率良く行うことである．つまり，「上手に定義する」というのは，この三つの手続きができる限り効率良く行えるように定義する，という意味である．

逆探索法を明示的に提案したのは Avis と Fukuda[3] である．彼らは超平面アレンジメントのセル列挙問題や凸多面体の頂点列挙問題を考える中で逆探索法に至った．

グレイコードによるアルゴリズムの説明の中で差分出力について簡単に触れた．逆探索法においても差分出力を援用できる．しかし，連続する二つの出力の差分が小さいことは保証されない．これを解消するために，深さ優先探索順ではなく前後順序探索順（prepostorder）という別の順序に基づいて出力を行うと，差分を小さく抑えることができる．実際に，差分は定数となり，よって，差分出力を行うと，遅延は $O(1)$ となる．前後順序探索順については Knuth の本[1] を参照のこと．

6. 手法の得失

ここまで，分割法，グレイコード，逆探索法という三つの手法を紹介した．それぞれの得失をここでまとめておく．

分割法は再帰的な構造を持つ対象の列挙に対して有効な手法である．そして，再帰的な構造を持ちさえすれば，その構造をそのままアルゴリズムに転換できるため，アルゴリズム設計は容易であり，それは利点である．しかし，再帰的な構造がなければ分割法による列挙は難しい．

グレイコードはグラフ上のハミルトン道を探索するため，単純な構造を持っている対象にしか適用できない．しかし，グレイコードが適用できる場合には，高速なアルゴリズムが得られる点は大きな利点である．一般に，グラフ上のハミルトン道を発見することは難しいため，アルゴリズム設計自身も難しい．

逆探索法はグラフ上の全域根付き木を探索するため，比較的複雑な構造を持っている対象に対しても適用でき，それが大きな利点である．しかし，全域根付き木の定義や子発見アルゴリズムの設計といった非自明な部分も多く，それがアルゴリズム設計を難しくしている．

7. 難しい列挙問題

ここまでは，効率良い列挙アルゴリズム設計を考えてきたが，手持ちの問題によってはどう頑張っても効率良い列挙アルゴリズムが設計できない場合があるかもしれない．これは設計者自身の努力不足であるかもしれないし，そうではなく，問題そのものが持つ複雑性が関連しているかもしれない．ここでは後者の場合がどのように捉えられるか紹介する．

7.1 一つ出力することすら難しい問題

列挙問題は全ての出力を見つける必要があるが，一つ出力することすら難しい問題が存在する．例えば，部分集合和列挙問題を考える．これは，入力として $n+1$ 個の自然数 $a_1, \cdots, a_n, b$ が与えられたとき，$\sum_{i \in S} a_i = b$ となる $S \subseteq [n]$ を全て出力する問題である．

この問題において，出力の数が 1 以上であるか，0 であるかを判定することは NP 完全であることが知られている（これは古典的な部分集合和問題である[4]）．そのため，出力多項式時間アルゴリズムが存在したとすると，P＝NP が導かれる．

このように，対応する判定問題が NP 完全である（あるいは，それよりも難しい）場合には，列挙すら難しいということを肝に銘じる必要があり，列挙の前に「一つ出力することがどれほど難しいか」を考えることが重要である．

7.2 今までの出力に漏れがないかの判定が難しい問題

凸多面集合の頂点列挙問題という近寄り難い問題がある．これは，入力として自然数 d と d 次元空間中の n 個の半空間が与えられたとき，その半空間全体の共通部分（これは凸多面集合である）の頂点を全て出力する問題である．この問題は 7.1 の部分集合和列挙問題とは違い，一つ出力することは多項式時間で可能な問題になっている．

しかし，Khachiyan ら[5] は，この問題の入力と出力の一部が与えられたとき，それ以外に出力すべきものがあるか判定する問題が NP 困難であることを証明した．すなわち，この問題に出力多項式時間アルゴリズムがあるとすれば，やはり P＝NP となってしまうのである．つまり，一つ出力することが簡単であるからといって，列挙が簡単であるとも限らないのである．

7.3 重複判定が難しい問題

重複判定というのは，二つの対象が同じであるかどうかを判定することであるが，それが難しい場合というものがある．例えば，二つの無向グラフが同型であるかどうかの判定は多項式時間で可能かどうか知られておらず，これは計算の理論における大きな未解決問題である．

そのため，同型を除いたグラフの列挙という問題において，今まで出力したグラフを作業領域に保持しておいても，新たに出力しようとするグラフが重複となるかどうかの判定を多項式時間でどのように行えばよいか分からない．

8. ま　と　め

列挙アルゴリズム設計の基礎的手法として，分割法，グレイコード，逆探索法を紹介し，その効率性の評価を行った．列挙アルゴリズムの理論研究においては，ハイパーグラフ極小横断の列挙や有界凸多面体の頂点列挙といった大きな未解決問題が残されており，更なる進展が期待されている．

謝辞　本稿の執筆を御提案頂いた山中克久先生，及び，詳細な御意見をお寄せ頂いた会誌編集委員会に感謝する．

文　献

（1） D.E. Knuth, The Art of Computer Programming Volume 4A Combinatorial Algorithms Part 1, Addison-Wesley, Upper Saddle River, NJ, 2011.
（2） C. Savage, “A survey of combinatorial Gray codes,” SIAM Rev., vol. 39, no. 4, pp. 605-629, 1997.
（3） D. Avis and K. Fukuda, “Reverse search for enumeration,” Discrete Appl. Math., vol. 65, no. 1-3, pp. 21-46, 1996.
（4） R.M. Karp, “Reducibility among combinatorial problems,” in Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds., pp. 85-103, Plenum, New York, 1972.
（5） L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich, “Generating all vertices of a polyhedron is hard,” Discrete Comput. Geom., vol. 39, no. 1-3, pp. 174-190, 2008.

（平成 23 年 12 月 30 日受付　平成 24 年 1 月 24 日最終受付）

岡本　吉央（正員）

平 17 スイス連邦工科大チューリヒ校大学院課程了．同年豊橋技科大情報工学系助手．平 19 同助教．同年東工大大学院情報理工学研究科特任准教授．平 22 北陸先端大特任准教授．平 24 電通大情報・通信工学専攻准教授．Ph.D. 専門は離散数学，離散アルゴリズム，離散最適化．