Vol. 27 No. 5 情報処理学会論文誌 May 1986

確率的アルゴリズムにおける効率・正解率間のトレードオフ関係について†

渡辺 治††

確率的アルゴリズムとは，乱数を利用して非決定的に計算を行うアルゴリズムの総称である．乱数を用いて計算過程を決めていくので，同じ入力に対してもつねに同じ答えを出すとは限らない．しかも誤って答える場合もある（この確率を誤り率と呼ぶ）．このような確率的アプローチにおいて，効率と正解率（＝1－誤り率）間のトレードオフ関係が予想される．しかしその関係が証明された例はほとんどない．本稿では，集合操作の基本演算でありソーティングと深くかかわりのある同一要素判定問題において効率・正解率間の詳しいトレードオフ関係を示す．すなわち(i)同一要素判定問題を解く確率的アルゴリズムを示し，そのアルゴリズムにおける効率・正解率の関係を調べる，(ii)そのアルゴリズムが効率・正解率の面から最適であることを証明し，(i)で得られた関係が本当のトレードオフ関係であることを示す．

1. まえがき

P≠NP などの予想から，非決定的計算を効率よく決定的に行うことはたいへん難しい（不可能である）と考えられている．そこで乱数を利用して擬似非決定的計算を行う方法が提唱され，効率のよいアルゴリズムがいくつか考案された[2),10)]．このような計算方法を総称して確率的アルゴリズムと呼ぶ．確率的アルゴリズムは乱数を用いて計算過程を決めてゆくので，同じ入力に対してもつねに同じ答えを出すとは限らない．しかも誤って答える場合もある（この確率を誤り率と呼ぶ）．しかしそのために従来の（決定的）アルゴリズムでは得られないような効率（計算速度）を得られる場合がある．このような確率的アルゴリズムの中でよいアルゴリズムとは正解率（＝1－誤り率）が高く効率のよいものであると考えるべきであろう．当然のことながらこの両者—効率と正解率—の間にはトレードオフの関係が予想される．しかしそのことが証明された例はほとんどない[11)]．本稿では同一要素判定問題を例にとり効率と正解率の間の精密なトレードオフ関係を示す．

同一要素判定 (element uniqueness problem) とは，与えられた集合の中に同一要素があるか否かを判定する問題である．これは，二つの集合の共通部分を見つける問題などと同種の問題であり，集合操作に関する基本的問題の一つである．本稿では全体集合が全順序集合であるという条件のもとでこの問題を考える．全順序集合における同一要素判定アルゴリズムは，ソーティング・アルゴリズムと密接な関係にあり，任意のソーティング・アルゴリズムは同一要素判定アルゴリズムに手直しすることができる．そこでFord-Johnsonのソーティング・アルゴリズム[3)]から比較回数が約 $n\log n$ 回の同一要素判定アルゴリズムが得られる（ただし n は入力される要素数．また $\log n$ の底は本稿全体を通して2とする）．しかも，この問題での比較回数の下限も約 $n\log n$ であることが知られているので[1),6)]，決定的手法ではこのアルゴリズムがほぼ最適である．

この問題に対する確率的アプローチについてはManber らによって研究され[5)]，つぎのようなことが知られている：

（1）もし正解率のことを考えに入れなければ，非常に少ない比較回数で同一要素判定を行うことができる（与えられた集合の中からランダムに要素を一組取りだして等しいかどうかを判定すればよい：比較回数は1回で済む）．

（2）しかし正解率をある定数 ε 以上にしようとすると，確率的アルゴリズムでもやはり $\Omega(n\log n)$ 回の比較が必要である．

これも正解率と効率の間のトレードオフ関係のひとつといえよう．しかし正解率の下限と比較回数の間のもっと精密な関係を求められないだろうか？ この問いは確率的手法の能力・性質を知るといった理論的な意味を持っているが，実際的な側面も持っている．たとえば $1\cdot10^9$ 回以下の比較回数のもとで $1\cdot10^8$ 個のデータの中から同一要素を見つける問題を考えてみよ

† The Time-Precision Tradeoff of Probabilistic Algorithms by OSAMU WATANABE (Department of Information Science, Tokyo Institute of Technology).

†† 東京工業大学理学部情報科学科

う（決定的手法では少なくとも約 $2.7 \cdot 10^9$ の比較が必要）．上の（1），（2）からは，この条件のもとでどのくらいの正解率が望めるか，どうすれば最も正解率が高いかなどの問いに答えられない．さらに精密なトレードオフ関係に関する考察が必要である．

本稿では（1），（2）よりさらに精密にトレードオフ関係を調べ，n，正解率，比較回数の間のより詳しい関係を与える．すなわち任意の p に対し，（i）p 以上の正解率を得るための確率的アルゴリズムを示しその比較回数を p と n の関数として求める，（ii）その正解率を得るために必要な比較回数の下限を求め，ここで示したアルゴリズムが（定数倍を除いては）最適であることを示す．

2. 問題の定義と計算モデルの設定

本稿で取り扱う問題を正確に述べると次のようになる．

（サイズ n の）同一要素判定問題：

集合 Ω を全順序集合とし，任意の二要素間の“大小比較”を行うことができるものとする．この集合 Ω の元からなる n 項ベクトル $(a_1, \cdots, a_n)$ が与えられたとき，ある i, j $(i \neq j)$ において $a_i = a_j$ であるか否かを判定する問題．□

したがって以下では“YES（ある i, j で $a_i = a_j$ である）”あるいは“NO”と答えるだけのアルゴリズムを考える．しかしこれからの議論は同一要素を“見つける”アルゴリズムにも適用できる．

これからの議論を簡単にし正確にするために，適当な計算モデルを設定する．ソーティングなどのように比較演算が本質的である場合，アルゴリズムの効率をデータ間の比較回数によって評価することが多い[4),6),7)]．ここでも“データ間の比較”をアルゴリズムの他の要素と切り離し，データ間の比較回数によって計算コストを評価できるようにモデル化する．

比較演算アルゴリズムのモデル：

各アルゴリズムは，データ用の変数 $x_1, x_2, \cdots, y_1, y_2, \cdots, z_1, z_2, \cdots$ およびその他の変数を用いる．問題のサイズが n のときには，$x_1, \cdots, x_n$ を入力変数と呼びアルゴリズムの実行開始時にある値が入っているものとする（$x_1, \cdots, x_n$ に入っている値を入力値と呼び n 項列ベクトルで表す）．計算の過程でデータ用変数に対してできる操作はデータ用変数間の代入と，データ用変数間の“大小比較”のみとする．□

アルゴリズムの計算コスト：

二つのデータ用変数間の“大小比較”の回数．□

実際のプログラムでは，$x_1, x_2, \cdots$ などは変数としてとられるのではなく，ディスク上の領域に割り当てられると考えた方が妥当であろう．するとデータ変数間の比較演算は「x_i と y_j をメモリー上にとってきて比較する」といった手続きになり，比較演算の時間が実行時間の多くを占めることになろう．

確率的アルゴリズムといっても従来の（決定的）アルゴリズムとあまり違わない．ただ，確率的アルゴリズムでは積極的に乱数を利用するという点が異なる．乱数は次のような乱数発生関数を用いてつくる．

random(m)＝1 以上 m 以下の整数よりランダムに選んだ数．

普通このような関数は疑似乱数発生プログラムにより実現されるが，ここでは random(m) は一様乱数であると仮定する．確率的アルゴリズムでは乱数を用いるため，一組の入力値 $\boldsymbol{a}$ に対しいつも同じ計算過程をとるとは限らない．したがって入力値 $\boldsymbol{a}$ に対して出力する値や入力値 $\boldsymbol{a}$ に対する計算コストを一意に決めることができない．そこで次のような値によって確率的アルゴリズム X の入力値に対する振舞い方を表す．

$p_X(\boldsymbol{a}) = Pr\{X$が入力値 $\boldsymbol{a}$ に対して“YES”と出力する$\}$*，

$c_X(\boldsymbol{a}) = \max\{\gamma$ での比較回数 $|$ γ は $\boldsymbol{a}$ に対する X の計算過程$\}$．

ここで $p_X(\boldsymbol{a})$ や $c_X(\boldsymbol{a})$ は入力値に対する値であり，入力値全体に対する確率であったり入力値全体に対する最悪時計算コストではない点に注意されたい．

3. 同一要素判定アルゴリズムの概略

ソーティング・アルゴリズムから同一要素判定アルゴリズムを作れることはよく知られている（同一要素対はソートの過程で必ず見つかる[7)]）．そこで Ford-Johnson のソーティング・アルゴリズム[3)]を用いると，ほぼ $n\log n$ 回の比較で同一要素の判定ができる．しかもそれがほぼ最適であることが証明されている[1)]**．

ところがある w $(w < n\log n)$ に対し，w 回より多く比較できないと仮定した場合，このアルゴリズムは使えない．というより下限の議論からつねに正しい答えを出すアルゴリズムを作ることができない．そこで

---

* $Pr\{\alpha\}$ は α という事象の起きる確率．

** Manacher のアルゴリズム[4)]の方が Ford-Johnson のものより比較回数が少ない．しかしその差は僅かであり後者の方が単純なので，ここではそれを使う．

ランダム化の手法をとる．最も単純な方法は，m個($m\log m<w$)をn個中よりランダムに選んでくる，そのm個に対しソーティングを行う，その過程で同一要素判定を行う，といったアルゴリズムである（以下，この方法をアルゴリズムAと呼ぶ）．

このアルゴリズムは，同一要素を持たないすべての入力値に対しいつも正しい答え（"NO"）を出力する．しかし同一要素を持つ入力値に対してはつねに"YES"を出力するとは限らず，その正解率は最悪の入力値（一組の同一要素しか持たない入力値）の場合でほぼ$(m/n)^2$である．

アルゴリズムAではm個を独立に選んできたのだが，k番目の元と$k+m-1$番目の元の間にある元を選んできたらどうだろうか（k自体はランダムに選ぶ）．これが本稿で提案するアルゴリズムの基本的考え方である．このアイデアを使ったのが次に示すアルゴリズムである．

Algorithm B
begin
$k\leftarrow$random(n);
(*) $\begin{cases} z_0\leftarrow$'$k$th best ($k$th smallest)' in $\boldsymbol{x}$*;
$z_1\leftarrow$'$(k+m-1)$th best';
$(y_1,\cdots,y_m)\leftarrow$elements of $\boldsymbol{x}$ such that $z_0\leqq x_i\leqq z_1$; $\end{cases}$
sort $\boldsymbol{y}$ and find a pair of variables having the same value;
if $\boldsymbol{y}$ contains such a pair
then "YES" else "NO" fi
end.

このアルゴリズムでも同一要素を持つ入力値に対していつも"YES"と答えるとは限らない．しかし正解率は最悪の入力値の場合でm/nとアルゴリズムAより大きい．そのかわりz_0, z_1を選び，z_0とz_1の間にある要素を選ぶために余計な比較が必要となる．ところが'kth best（n個中でk番目に小さい値）'を求めるには約$3n$回の比較でよいことが知られている[8]．またソーティングと同様，z_0が求まった時点でx中のどの変数がz_0より大きな値（あるいは小さな値）を持っているかが分かっているはずだ．そこで，$\boldsymbol{y}$を選ぶために必要な比較回数はたかだか$6n$回である．したがって$6n$回の前処理が大きく響かないようなwに対しては有効である．

もしwがさらに小さいときは，次のようにすればよい：適当なn'個をn個のデータよりランダムに選んできて，そのn'個のデータに対してアルゴリズムBを実行する（以下，この方法をアルゴリズムCと呼ぶ）．

4. 各アルゴリズムの解析

3章で示したアルゴリズムB，Cは，同一要素判定問題に対するほぼ最適なアルゴリズムである（証明は5章で述べる）．そこで本章では，アルゴリズムB，Cの計算コストと正解率の解析を行い両者の関係を調べる．また比較のため単純なアルゴリズム（アルゴリズムA）の解析も同時に行う．

1章でも述べたように，n（問題のサイズ），w（比較回数の上限），p（正解率の下限）の関係を求めることが本稿の目的である．しかしそれらの関係を一般式の形で扱うと複雑になるので，ここではwを代表的なnの関数としたときのpとnの関係を示す（n，w，p間の関係式の一般形については文献9)を参照されたい）．また以下では計算をやりやすくするため$n>2^{20}$として議論を進める．ただし，その事実を用いて式を簡単化する場合にはつねに断わってから行うことにする．

4.1 ソーティングおよび選択のアルゴリズムの計算量について

各アルゴリズムの解析に入る前に，それらの中で利用されているソーティングおよび選択（'kth best'をもとめる）アルゴリズムについて知られている事実を述べる．

ソーティング・アルゴリズム：

Ford-Johnson のアルゴリズムでn個をソートする時に必要な比較回数の上限$F(n)$は，$F(n)=n\log n-cn+\frac{1}{2}\log n+O(1)$（ただし $1.329<c<1.415$）であることが知られている[3),4)]．このうち$\frac{1}{2}\log n+O(1)$の項は$n>2^{20}$のときnに対する比率が非常に小さいので無視する．またcはこの範囲内でどう決めてもこれからの話に大きな違いが生じないので，式を簡単にするために$c=1.4$とする．つまり，以下では$F(n)=n\log n-1.4n$と考えて話を進めることにする．□

選択アルゴリズム：

Schönhageらの提案したアルゴリズムが現在最も比較回数が少ない[8)]*．そのアルゴリズムでn個の中からt番目に小さい値（$1\leqq t\leqq n$）を求めるのに必要な比較回数の下限を$V_t(n)$とすると，すべてのt（$1\leqq t\leqq n$）

* 誤解を生じないときは，データ用変数もベクトル$\boldsymbol{x}$, $\boldsymbol{y}$などで表す．

* 元木はさらに効率のよいアルゴリズムを提案したが[9)]，やはりその差が僅かなので Schönhage らのアルゴリズムで考える．

に対し

$$V_t(n)\leqq 3n+\left(\frac{\log n+\log k+6}{k}\right)n+V_{t'}(8k^2+2k+4). \tag{4.1}$$

(ただし，k は $1\leqq k\leqq n$ を満たす任意のパラメータ) 適当な k を定めて $n>2^{20}$ である仮定のもとに $V_t(n)$ を解析すると $V_t(n)<3n+0.296n$ が得られる (付録(1)参照)．以下では $V_t(n)=3.3n$ として話を進める．ただし $V_t(n)$ は漸近的に $3n$ に近づくので n が十分大きければ $V_t(n)=3n$ としてもよい．たとえば $n>2^{50}$ では $V_t(n)<3.002n$ となる．□

4.2 アルゴリズムA, B, Cの解析

サイズ n の入力値全体を D_n，そのうち同一要素を持つ入力値全体を L_n とそれぞれ表記する (D_n は n 項整数列ベクトルの全体)．アルゴリズム A, B, C はどれも D_n-L_n の元に対しては正しく "NO" と答える．そこでアルゴリズム X (XはA，B，C) の正解率の下限を求めるには，L_n の元 $\boldsymbol{a}$ に対して "YES" と答える確率 $p_X(a)$ の下限を求めればよい(これを $p_X(n)$ または単に p_X で表す)．さらに L_n のなかでも同一要素の組がひとつしかない入力値が最悪であるので，$p_x(n)$ を求めるときはそのような入力値だけ (その全体を M_n とする) について調べる．各アルゴリズムで必要な比較回数の入力値全体に対する下限を $c_X(n)$ あるいは c_X と記述する．

アルゴリズムAの解析：

3章で述べたアルゴリズムAにおいて，ちょうど m 個の x_i を $\boldsymbol{x}$ の中から選ぶのは難しい (同じ x_i が重複して選ばれる可能性がある)．しかしたとえば次のようにしても，ほぼ同じ正解率が得られる．

```
for m times do
  j←random(n);
  if j is not chosen before then choose x_j fi
od;
```

以下，アルゴリズムA (およびC) では，この方法を用いていると考え解析を進める．

まず正解率の下限 p_A を求めてみよう．入力値 $\boldsymbol{a}$ は M_n の元でしかも $a_1=a_2$ であるとする．このとき $p_A(\boldsymbol{a})$ は

$$\begin{aligned} p_A(\boldsymbol{a}) &= Pr\{a_1, a_2 \text{ が選ばれる}\}, \\ &= {}_mP_2\times Pr\{t \text{ 回目に } a_1 \text{ が選ばれる}\} \\ &\quad \times Pr\{t' \text{ 回目に } a_2 \text{ が選ばれる}\}, \\ &= m(m-1)/n^2. \end{aligned}$$

入力値 $\boldsymbol{a}$ は最悪な入力値のひとつであるから，正解率の下限 $p_A(n)=m(m-1)/n^2$ となる．

つぎに計算コスト c_A について考える．データ用変数間の比較は選ばれた (m個の) データ間のソーティングのところでしか行われていない．したがって $c_A(n)=F(m)=m\log m-1.4m$ である．□

アルゴリズムBの解析：

アルゴリズムBにおいても $\boldsymbol{y}$ の選び方 (文(*)) をもう少し詳しく考えると次のようになる．

```
k←random(n);
if k>n/2 then
  k0←k−m/2+1;
  z←elements of x which are larger than
     'k0th best' in x;
  (this is done during the selection of
   k0th best in x)
  y←elements of z which are smaller than
     '(m−1)th best' in z;
else
  k0←k+m/2−1;
  z←elements of x which are smaller than
     'k0th worst' in x;
  y←elements of z which are larger than
     '(m−1)th worst' in z;
fi;
```

この $\boldsymbol{y}$ を選ぶ過程で同一要素が見つかる場合もある．その場合にはただちに "YES" と答えて停止する (同一要素が見つからない場合は，$\boldsymbol{y}$ に選ばれる要素の数は $m-1$ 以下であることに注意)．

正解率の下限 p_B を求めよう．入力値 $\boldsymbol{a}$ は M_n の元で，ソートすると $(a_{i_1},\cdots,a_{i_n})$ となり，このうちの j_0 番目と j_0+1 番目の要素が等しいものとする．この二つの要素がともに $\boldsymbol{y}$ に選ばれれば ($k-m/2+1\leqq j_0$, $j_0+1\leqq k+m/2-1$ ならば) その同一性を検出できる．また $j_0+1=k-m/2+1$ あるいは $j_0=k+m/2-1$ のときでも，$\boldsymbol{y}$ を選ぶ過程で $a_{j_0}=a_{j_0+1}$ を検出できる．したがって

$$\begin{aligned} p_B(\boldsymbol{a}) &= Pr\{k-m/2\leqq j_0\leqq k+m/2-1\}, \\ &= m/n. \end{aligned}$$

これが正解率の下限 $p_B(n)$ になる．

また計算コスト c_B は次のようになる．

$$\begin{aligned} c_B(n) &= V_{k'}(n)+V_{m-1}(n-(k-m/2))+F(m), \\ &\leqq 4.95n+m\log m+0.25m. \end{aligned}$$

□

アルゴリズムCの解析：

アルゴリズムAおよびBの解析から p_C と c_C は容易に求まる．

$$p_C(n)=\frac{n'(n'-1)}{n^2}\cdot\frac{m}{n'}=\frac{m(n'-1)}{n^2},$$

$$c_C(n)=4.95n'+m\log m+0.25m$$

ただし n' ($1\leqq n'\leqq n$ なる適当な値) は，付録(3)より $(m\log m+0.25m)/4.95$ と定める．□

4.3 比較回数と正解率の関係

各アルゴリズムに対し m と n を用いて正解率の下限と計算コストを表したので，これらから m を消去すれば問題のサイズが n の場合の正解率の下限と計算コストの関係を示すことができる．しかし m を消去しようとすると不必要に複雑になるので，c_X の上限 w がそれぞれ $6n$ と $n/2$ である場合を例にとり，p_X と n の漸近的関係を調べることにする．

$w=6n$ の場合：

$4.95n$ 回の前処理を行うことができるのでアルゴリズムBが使える．しかもアルゴリズムCでは $n'=n$ となり，アルゴリズムBと同じになる．そこでアルゴリズムA，Bについて調べればよい．

まず p_A を評価しよう．m には，$c_A(n)=m_0\log m_0-1.4m_0=6n$ となる m_0 を使うと p_A が最も高くなることは明らかだろう．ほぼ $m_0=6n/(\log n+2.2)$ であるから（付録(2)），それを $p_A=m(m-1)/n^2$ に代入すると

$$p_A(n)=\left(\frac{10}{\log n}\right)^2+O\left(\frac{1}{(\log n)^2}\right).$$

つまり p_A は漸近的に $(1/\log n)^2$ に比例して小さくなる．

これに対し，p_B を同様に評価すると

$$p_B(n)=\frac{1.05}{\log n}+O\left(\frac{1}{\log n}\right)$$

が得られる（p_B は漸近的に $1/\log n$ に比例する）．□

$w=n/2$ の場合：

アルゴリズムBは w が小さすぎるため働かないのでアルゴリズムA，Cについて調べる．

前と同様にして

$$p_A(n)=\left(\frac{1}{2\log n}\right)^2+O\left(\frac{1}{(\log n)^2}\right).$$

次に p_C を評価しよう．アルゴリズムCにおける n' と m は，それぞれ $n'=n/9.9$，$m=n/2(\log n+0.25)$ となり（付録(3)），

$$p_C(n)=\frac{1}{2\log n}+O\left(\frac{1}{\log n}\right)$$

が得られる（p_C は漸近的に $1/\log n$ に比例する）．□

さらに一般的に考えると，比較回数の上限 w が n に比例する場合，アルゴリズムB，Cを用いれば漸近的に $1/\log n$ に比例する正解率が得られる．それに対しアルゴリズムAでは正解率は $(1/\log n)^2$ に比例する．

5. 確率的アルゴリズムの限界

本章では同一要素判定問題を解く確率的アルゴリズムの限界について述べる（いわゆる“計算コストの下限”について議論する）．ただし4章と同じ形で結果を示す．すなわち比較回数を制限する値 w を，それぞれ $6n$ と $n/2$ とした場合の正解率の限界（いくら以上の正解率を得ることは不可能か）を示す．

渡辺は，この問題に対する計算コストの下限を一般式の形で示した[11]．そこでは計算モデルとして確率的決定木[4]が用いられている．これは本稿で考えている計算モデルと似ているが，計算コストを“比較回数”と“乱数発生に必要なコスト（1回の random(n) の呼び出しコストを $\lceil\log n\rceil$ と考える）”の和とする点が異なる（この計算コストを h で表す）．文献11)では計算コストの下限を求めるのに次のような補題を用いているが，これらはここでも重要である（証明は長くなるので省く）．

補題 5.1：

X を任意の同一要素判定アルゴリズムとする．ただし $h_X(n)\geqq n$ とする．このとき

(1) $q_n>2^{17}(0<q\leqq 1)$,

(2) $\exp\{p_X(\boldsymbol{a})\mid\boldsymbol{a}\in L_n\}\geqq q$

ならば

$$h_X(n)>\frac{1}{6}q_n(\log q_n-1).$$

（ただし，$\exp\{p_X(\boldsymbol{a})\mid\boldsymbol{a}\in L_n\}$ は $p_X(\boldsymbol{a})$ の L_n 全体における平均）．□

補題 5.2：

X を任意の同一要素判定アルゴリズムとする．ただし X はたかだか n' 個の変数しか見られないものとする．このとき，

(1) $q_{n'}>2^{17}$ $(0<q\leqq 1)$,

(2) $\exp\{p_X(\boldsymbol{a})\mid\boldsymbol{a}\in L_n\}\geqq q$

ならば

$$h_X(n)>\frac{1}{6}q_{n'}(\log q_{n'}-1).$$ □

以上は p_X の平均 h_X との関係であるが，次の補題からこれを p_X と c_X との関係にも拡張できる．

補題 5.3：

任意の同一要素判定アルゴリズム X に対し，次の条件をみたすアルゴリズム Y が存在する．

(1) $c_X(n)=c_Y(n)$,

(2) $\exp\{p_X(\boldsymbol{a})|\boldsymbol{a}\in L_n\}\leqq\exp\{p_Y(\boldsymbol{a})|\boldsymbol{a}\in L_n\}$,

(3) $h_Y(n)=c_Y(n)$.

(すなわち，Yは決定的アルゴリズム)

(略証) アルゴリズムXの random を呼んでいる所で random(m) の値を適当な m' ($1\leqq m'\leqq m$) に置き換えてYを作る．この置き換えによりすべての $\boldsymbol{a}\in L_n$ で $p_Y(\boldsymbol{a})$ が ($p_X(\boldsymbol{a})$ より) 減らなければ問題はない．どんな m' に置き換えても，ある $\boldsymbol{a}\in L_n$ で $p_Y(\boldsymbol{a})$ が減ってしまう場合もある．ところがそんな場合には，$p_Y(\boldsymbol{a})$ が減った分以上は必ず他の $\boldsymbol{b}\in L_n$ で $p_Y(\boldsymbol{b})$ が増えるように m' を選べる．したがって L_n 全体の平均は減少しない．□

以上の補題を用いて $w=6n$, $n/2$ のそれぞれの場合での正解率の下限を調べる．

$w=6n$ の場合：

すべての確率的同一要素判定アルゴリズムで，正解率は $36'/\log n$ 未満である ($36'=6\cdot(6+\Delta)$; ただし $\Delta<0.2$)．

(略証) あるアルゴリズムXで正解率が $36'/\log n$ 以上であると仮定すると，次の関係式が成り立つ．

$$\frac{36'}{\log n}\leqq p_X(n)=\inf\{p_X(\boldsymbol{a})|\boldsymbol{a}\in L_n\},$$
$$\leqq\exp\{p_X(\boldsymbol{a})|\boldsymbol{a}\in L_n\},$$
$$\leqq\exp\{p_Y(\boldsymbol{a})|\boldsymbol{a}\in L_n\}.$$

(ただし，Yは補題5.3のY)

アルゴリズムYは少なくともn回以上の比較をしていると仮定できるので，補題5.1が適用でき

$$h_Y>\frac{6\cdot(6+\Delta)}{6\log n}(\log n+\log 36'-\log\log n-1)>6n.$$

また補題5.3より $c_X=c_Y=h_Y$ なので，$c_X>6n$ となり $w=6n$ に矛盾する．

(注) 以上の議論は，$n\geqq 2^{36'}$ のとき意味がある ($n<2^{36'}$ ならば $36'/\log n>1$ となるので無意味)．□

$w=n/2$ の場合：

すべての確率的同一要素判定アルゴリズムで，正解率は $6'/2\log n$ 未満である ($6'=6+\Delta$; $n>2^{20}$ のとき $\Delta<0.2$)．

(略証) あるアルゴリズムXで正解率が $6'/2\log n$ 以上であると仮定し，上と同じように考えると

$$\frac{6'}{2\log n}\leqq\exp\{p_Y(\boldsymbol{a})|\boldsymbol{a}\in L_n\}$$

が得られる．アルゴリズムYではたかだか $(n/2)\cdot 2$ 個の変数しか見られない．したがって補題5.2を用いると ($n'=n$), $c_X=c_Y=h_Y>n/2$ となり矛盾 (ただし，$h_Y>n/2$ を導くとき $n>2^{20}$ を利用)．□

以上の考察からたとえば次のようなことがいえる：比較回数の上限wがnに比例する場合，アルゴリズムB，Cによる正解率の下限は漸近的に $1/\log n$ に比例するが，このような正解率の振舞い方は最適である．つまりどんなに良いアルゴリズムXでも，p_X は漸近的に $1/\log n$ に比例するようになる．

6. あとがき

与えられた入力列中に同じデータがあるかどうかを判定する三種類の確率的アルゴリズム（アルゴリズムA，B，C）を考え，それらの計算コスト（データ間の比較回数）と正解率について解析した．そして漸近的な解析結果から，入力サイズが十分大きい場合にはアルゴリズムB，Cの方が単純な方法（アルゴリズムA）より優れていることを示した．またこの問題に対する確率的手法の限界についても考察し，アルゴリズムB，Cは漸近的にみて最適であることを示した．

理論的な面からみると本稿では，確率的アルゴリズムにおける計算コストと正解率間のトレードオフ関係を調べたことになる．直感的にはたいていの問題でこの関係が生じると予想できるが，その関係があることが証明された問題はほとんどない[11]．アルゴリズムB，Cを作ったことおよびそれらが最適であることを示したことにより計算コストと正解率のかなり詳しいトレードオフ関係を証明したことになる．

本稿では“YES”，“NO”としか答えないアルゴリズムだけを考えたが，各アルゴリズムおよび下限の議論は“同一要素を見つける”問題にも適用できる．またこの問題は“集合間の共通要素を見つける”問題などにも容易に応用できる．さらに同一要素判定問題はソーティングと密接に関係しているので，本稿での考察は確率的ソーティング・アルゴリズムの性質を調べる際に役立つと思われる．

参考文献

1) Dobkin, D. P. and Lipton, R. J.: On the Complexity of Computations under Varying Sets of Primitives, *J. Comput. Syst. Sci.*, Vol. 18, pp. 86-91 (1979).
2) 五十嵐善英：確率的アルゴリズムの概観，情報処理，Vol. 21, No. 1, pp. 13-18 (1980).
3) Knuth, D. E.: *The Art of Computer Programming Vol. 3 : Sorting and Searching*, pp. 181-195 and pp. 209-218, Addison-Wesley, Reading (1973).
4) Manacher, G. K.: The Ford-Johnson Sorting

Algorithm is not Optimal, *J. ACM*, Vol. 26, No. 3, pp. 441-456 (1979).
5) Manber, U. and Tompa, M.: Probabilistic, Non-deterministic, and Alternating Decision Trees, *Proc. 14th Ann. ACM Symp. on Theory of Computation*, pp. 214-244 (1982).
6) Motoki, T.: A Note on Upper Bounds for the Selection Problem, *Inf. Process. Lett.*, Vol. 15, No. 5, pp. 214-219 (1982).
7) Reingold, M. E.: On the Optimality of Some Set Algorithms, *J. ACM*, Vol. 19, No. 4, pp. 649-659 (1972).
8) Schönhage, A., Paterson, M. and Pippenger, N.: Finding the Medium, *J. Comput. Syst. Sci.*, Vol. 13, pp. 184-199 (1976).
9) Watanabe, O.: The Relation between Time and Accepting Probability of Probabilistic Simple Decision Trees, *Res. Rep. Inform. Sci.*, C-50, Tokyo Institute of Technology (1983). (この論文の extended abstract は同じ題名で "数理解析研究所講究録, No. 494 (1983)" に掲載).
10) 渡辺 治: 確率的アルゴリズム, 情報処理, Vol. 24, No. 5, pp. 372-377 (1983).
11) Watanabe, O.: The Time-Precision Tradeoff Problem on On-line Probabilistic Turing Machines, *Theor. Comput. Sci.*, Vol. 24, pp. 105-117 (1983).

付　　録

(1) $V_i(n)$ の計算

すべての t $(1\leqq t\leqq n)$ に対し $V_i(n)$ は(4.1)式を満足するので, $f(n)=\max\{V_i(n)|1\leqq t\leqq n\}$ とすると

$$f(n)=3n+\left(\frac{\log n+\log k+6}{k}\right)n+f(8k^2+2k-4).$$

ここで $k=\lfloor\frac{1}{2}(n\log n)^{1/4}/2\rfloor$ とおき, $n>2^{20}$ という仮定のもとで整理すると

$$f(n)<3n+\frac{469}{132}(n\log n)^{3/4}+f\left(\frac{9}{4}(n\log n)^{1/2}\right).$$

この漸化式を解けばよいのだが, 2度目の再帰呼び出しのときは n が十分小さくなっているので少々効率が悪くても評価しやすい選択アルゴリズム[3]を使ったことにすると (そのアルゴリズムでは比較回数$<15n-163$), $f(n)<3n+0.296n$ となる (ただし $n>2^{20}$).

(2) $m\log m-am=b$ の解

ここでは $-1<a<2$, $b>2^{20}$ と仮定する. 上式の解として, たとえば

$$m_1=b/\{\log b-a+1-\log(\log b-a)\}$$

を用いると誤差が小さい ($0.951b<m_1\log m_1-am_1\leqq b$). しかし本稿での議論では b に関する主要項が残っていれば十分であり

$$m_2=b/(\log b-a+1)$$

を解として用いる (この場合でも $m_2\log m_2-am_2\leqq b$ であることに注意).

(3) アルゴリズムCにおける n' の決め方

アルゴリズムCにおいて n' は $1\leqq n'\leqq n$ であれば何でもよい. しかし p_C を最大にするよう n' を決めるべきだろう. パラメータ t $(t>0)$ を使い

$$n'=t(m\log m-0.25m)/4.95$$

とおき調べると, $t=1$ のとき p_C がほぼ最大となる.

(昭和60年7月2日受付)
(昭和61年2月20日採録)

渡辺　治 (正会員)

昭和33年生. 昭和55年東京工業大学理学部情報科学科卒業. 昭和57年同学理工学研究科博士課程を中退し, 同学理学部情報科学科助手となる. 計算の理論, プログラム理論に興味を持っている. 現在, 計算の複雑さの理論の研究に従事. EATCS, LA 各会員.