
# 9. セレクション法†

野　下　浩　平††

## 1. はじめに

At a Lawn Tennis Tournament, where I chanced, some while ago, to be a spectator, the present method of assigning prizes was brought to my notice by the lamentations of one of the Players, who had been beaten (and had thus lost all chance of a prize) early in the contest, and who had had the mortification of seeing the 2nd prize carried off by a Player whom he knew to be quite inferior to himself.

——Lewis Carroll (1883)

セレクション (selection, 選択) とは，$n$ 個の数の中で $t$ 番目に大きい数をみつけることである．ここではまず，選択のためのプログラムを紹介する．次に，比較演算の実行回数に関する研究結果を調べる．特に，中央値 (median) を選択する $t=\lceil n/2 \rceil$ の場合と $t$ が定数の場合の2つに注目しよう．

## 2. 選択プログラム

配列 $A$ の添字1から $n$ までの要素に入力の $n$ 個の数が納められているとして，選択問題を解くプログラムを図-1 に示す．このプログラムは，Hoare (1961) によるものを修整したものであり，普通，"FIND" という名前でよばれている．その考え方をみよう．まず，標本として，1つ数 $s$ を選ぶ（4行目）．ここでは，$A[L], \cdots, A[R]$ のうち1つをランダムに選んでいる．次に，$s$ を基準にして，$A[L], \cdots, A[R]$ を並べかえて，$s$ より小さい要素と大きい要素それぞれの集合に "分割" する．すなわち，

$A[i] \geqslant s$　$(L \leqslant i \leqslant r-1)$，$A[r]=s$，

$A[i] \leqslant s$　$(r+1 \leqslant i \leqslant R)$

となるように分割する $(L \leqslant r \leqslant R)$．この分割のための手続き (5-11行目) は，PARTITION とよばれているもので，ソーティング(整列)用の算法QUICKSORT (分割整列法) にも用いられている（溝口「ソート法」参照）．この分割の結果をみると，$t=r-L+1$ ならば，$A[r]$ が目標の数であり，$1 \leqslant t < r-L+1$ ならば，$A[L], \cdots, A[r-1]$ の中に，それ以外は，$A[r+1], \cdots, A[R]$ の中にそれぞれ目標の数があることがわかる．それで，第一の場合であれば，それで終り，第二，三の場合には，対応する範囲の中を探すために，再帰的にこの手順を適用すればよい．(図-1 では，12-13 行目の部分に相当するが，標準的な方法で全体を繰り返し型の構造に変換している．)

```
 1  {n≥1, 1≤t≤n}
 2  begin L:=1; R:=n; a[n+1]:=-∞;
 3    repeat {L≤R, 1≤t≤R-L+1}
 4      f:=random (L, R); s:=A[f]; A[f]:=A[L]; A[L]:=s;
 5      l:=L; r:=R+1;
 6      repeat repeat l:=l+1 until A[l]≤s;
 7             repeat r:=r-1 until A[r]≥s;
 8        if l<r then
 9          begin w:=A[l]; A[l]:=A[r]; A[r]:=w end
10      until l≥r;
11      A[L]:=A[r]; A[r]:=s;
12      if t<(r-L+1) then R:=r-1
13        else begin t:=t-r+L-1; L:=r+1 end
14    until t=0
15    {A[r] が目標の数である}
16  end
```

図-1　プログラム FIND

この算法の能率について調べよう．計算時間を評価するために，ここでは，2つの数の比較演算 (6, 7行目) に着目して，その実行回数を勘定しよう．簡単のため，入力の $n$ 個の数は，どれも互に相異なる大きさをもつものとする．

さて，$n$ 個の数の大小順の並び方は，$n!$ 通りあるが，入力としてどの並び方もランダムに $(1/n!$ の等確率で) 与えられるものと仮定する．各入力ごとに比較回数が定まるので，その '平均比較回数' $\bar{F}_t(n)$ が自然に定義される．

算法 FIND の平均比較回数について，次のことが証明できる：

† Algorithms for Selection by Kohei NOSHITA (Department of Computer Science Denkitusin University).

†† 電気通信大学計算機科学科

$\bar{F}_t(n)=O(n)$

もっと詳しくは，$\bar{F}_t(n)$ が $n$ と $t$ の関数として完全に表わすことができる (Knuth (1971))．特に，$t=\lceil n/2\rceil$ の場合，すなわち中央値を選択する場合，次が成立つ：

$\bar{F}_{\lceil n/2\rceil}(n)\simeq 2(1+\ln 2)n=3.39n.$

ここで，等号 $\simeq$ は，$n$ より低次の項を無視するという漸近的な意味で解釈する．($n$ を固定した時，$\bar{F}_t(n)$ は，$t$ がほぼ $n/2$ で最大になる．)

以上，算法 FIND の平均的な比較回数に関する結果をみたが，最も運の悪い入力データに対しては，すなわち‘最悪の場合’には，比較回数がほぼ $n^2/2$ になり，極めて能率が悪い．

なお，FIND を一層実用的なプログラムとするための能率向上の工夫がいろいろ発表されている．多くの場合，QUICKSORT に対する工夫がそのまま応用できる (Knuth (1975), Nozaki (1977) 等を参照)．

Floyd-Rivest (1975) は，FIND よりもっと速いプログラム“SELECT”を発表した．この算法は，標本として，FIND のように1つの数 $s$ を取出すのではなく，いくつかの数からなる集合 $S$ を取出し ($|S|=o(n)$)，その中から適当な2つの数 $u$, $v$ を選択する．そして，$u$ と $v$ を基準にして，配列 $A$ を3つの範囲に分割することにより，次に探す範囲として，大きさが大幅に減っているもの (大きさ $o(n)$) を (ほぼ確率1で) 指定することができる．以上のような考え方に基づく算法 SELECT について，平均比較回数 $\bar{S}_t(n)$ が次のようになる：

$\bar{S}_t(n)\simeq n+\min(t, n-t).$

特に，$\bar{S}_{\lceil n/2\rceil}(n)\simeq 1.5n$ となる．

実際の計算実験でも，中央値を選択する問題に対して，計算時間に関して，SELECT が FIND より，例えば $n=10000$ で，2倍以上速くなることが報告されている．

## 3. 選択算法の比較回数

選択問題を解くのに要する比較回数について調べよう．まず，比較回数という言葉の意味をはっきりさせるために，選択算法を決定木 (decision tree) によって定義する．$n=4$, $t=2$ に対する決定木の例を 図-2 に示す．この例より，一般的な定義を考えることは容易であろう．決定木の各節点には，比較によって判明した数の間の大小関数を表わす半順序集合を対応させる．なお図では，次に比較する2つの数 $x, y$ を点線で示している．比較の結果，$x>y$ であれば，左の子，$x<y$ であれば，右の子がそれぞれえられることになる．特に，根は，大小順序関係のない入力の $n$ 個の数の集合に対応しており，葉は，目標の $t$ 番目の数が一意的に指定できるものに対応している．(なお，どんな算法も無駄な比較は行わないものとする．)

さて，決定木が表わす算法 $\alpha$ の最悪の場合の比較回数 $V_t^\alpha(n)$ は，この決定木の高さで定義される．また，入力として，$n!$ 通りの並び方のどれかがランダムに与えられるものと仮定した時，算法 $\alpha$ の平均の場合の比較回数 $\bar{V}_t^\alpha(n)$ は，次の値として定義される：

$(\Sigma[\text{入力}(x_1,\cdots,x_n)\text{に対する葉の高さ}])/n!.$

ここで，$\Sigma$ は，$(1,2,\cdots,n)$ の順列 $(x_1,\cdots,x_n)$ すべてについての総和を表わす．

図-2 の算法 $\alpha$ については，$V_2^\alpha(4)=5$, $\bar{V}_2^\alpha(4)=100/24=4.17$ である．すぐわかることであるが，$V_2^\beta(4)=\bar{V}_2^\beta(4)=4$ となる別の算法 (決定木) $\beta$ を作

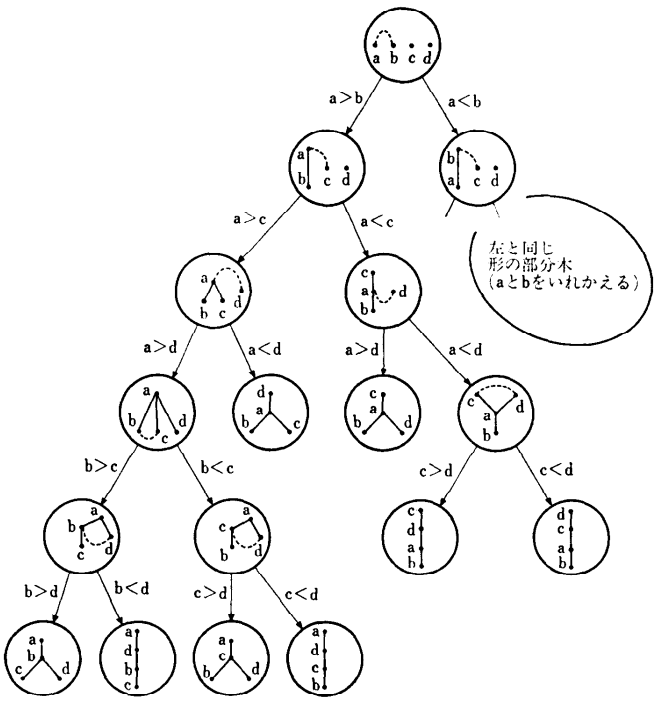


図-2 選択の決定木の例 ($n=4, t=2$)

ることができる.

このような決定木によって，選択算法を定義すれば，すべての選択算法という言葉の意味もはっきりするようになる．次の記法を導入しておこう：

$$V_t(n)=\min_{\alpha} V_t^{\alpha}(n),$$

$$\bar{V}_t(n)=\min_{\alpha} \bar{V}_t^{\alpha}(n).$$

すなわち，$V_t(n)$ と $\bar{V}_t(n)$ はそれぞれ，最悪の場合と平均の場合の必要十分な比較回数を表わしている．この値を実現する算法は，最適の算法ということになる．

次に，$V_t(n)$ と $\bar{V}_t(n)$ のそれぞれに関する諸結果をみていこう．ここでは，選択問題の研究の中心的な課題である $t=\lceil n/2\rceil$ の場合と $t$ が定数の場合について調べる．特に，$n$ が大きくなる時の漸近的な様子に関心をもつ．(なお，$t$ が $n$ に較べて小さいが，$n$ に依存するような場合の中間的な問題を扱う課題も研究されているが (Motoki (1982))，ここでは省略する.)

まず，中央値選択問題を取り上げる．$V_{\lceil n/2\rceil}(n)$ に対して良い上界を与える算法としては，

Blum-Floyd-Pratt-Rivest-Tarjan (1973) と
Schönhage-Paterson-Pippinger (1976)

による2つの算法がよく知られている．前者は，選択問題が $n$ に線型的な計算時間で解けることを示した最初の算法であり，その上界は，

$$V_{\lceil n/2\rceil}(n)\leq 5.43n$$

となっている．後者は，主要項として，現在最も良い上界を与える算法である．

$$V_{\lceil n/2\rceil}(n)\leq 3n.$$

この上界は，最適のものではないであろうと予想されているにも拘らず，もう長い間改良されていないし，現在のところ，あまり改良の手掛りがないようである．なお，Yao (1974) による有名なある予想命題が証明できれば，上界が $2.5n$ の算法が作ることができるが，その命題の証明も成功していない.

下界については，Pratt-Yao (1973) による次の結果がよく知られている：

$$V_{\lceil n/2\rceil}(n)\geq (7/4)n.$$

この下界については，いろいろな証明法が知られている．(下界の証明法の例としては，Blum 等 (1973), Nozaki (1973), Schönhage (1974), Kirkpatrick (1974, 81), Hyafil (1976), Noshita (1976), Fussenegger-Gabow (1979) がある.)

現在まで，次に証明できそうである下界 $2n$ を目標にして証明が試みられているが，成功していない．なお，Yap (1976) が下界 $(11/6)n$ をえたとの報告が一部の人々に配布されている.

以上みてきたように，上界と下界の間隙を狭くすることが差当りの課題である．最終的には，$2n$ が最適の主要項であろうという人もいれば，$2.5n$ の下界の証明を試みている人もいて，長い間の多くの努力にも拘らず，最適の主要項を決める見通しはたっていないといえる.

次に，$t$ が定数である場合の $V_t(n)$ について調べよう．$t=1,2,3$ の場合，$V_t(n)$ の値は，すべての $n$ に対して完全に決定されている.

$$V_1(n)=n-1,$$

$$V_2(n)=n-2+\lceil\log_2 n\rceil,$$

$$V_3(n)=\begin{cases} n-3+2\lceil\log_2(n-1)\rceil \\ \quad 5\cdot 2^{k-1}+1<n\leq 4\cdot 2^k+1 \text{ の時} \\ \quad (k\geq 1, n\geq 5), \\ n-4+2\lceil\log_2(n-1)\rceil \\ \quad 4\cdot 2^k+1<n\leq 5\cdot 2^k+1 \text{ の時} \\ \quad (k\geq 0, n\geq 6). \end{cases}$$

ここで，$V_1(n)$ については，明らか，$V_2(n)$ については，Kislitsyn (1964) による (Knuth (1973) 参照)．$V_3(n)$ については，この10年位調べられてきたが，最終的に，Aigner (1982) により確定されている．一般の $t$ ($\geq 4$) については，Hadian-Sobel (1969) の上界式

$$V_t(n)\leq n-t+(t-1)\lceil\log_2(n-t+2)\rceil$$

がよく知られている (Knuth (1973))．

この上界式を改良する方法や下界の証明は，いろいろな人々により試みられている (例えば，Yao (1974), Kirkpatrick (1974), Yap (1976), Hyafil (1976), Noshita (1976))．現在のところ，これらの研究により，$V_t(n)$ は，ほぼ

$$n-t+(t-1)\log_2 n$$

であることが知られており，最も良い上界と下界の差は，約 $t\log_2 t$ であり，$n$ に依存しない定数の差まで縮められている.

したがって，$t$ が定数である場合の $V_t(n)$ を定める課題は，もうほとんど解決されているといえる．それでも，差当り，$V_4(n)$ の値を完全に決定することが次の課題であろう.

次に，平均比較回数 $\bar{V}_t(n)$ について調べよう.

まず，中央値選択問題の上界として，

$$\bar{V}_{\lceil n/2\rceil}(n)\leq 1.5n$$

である．これは，**2** の Floyd-Rivest の算法が与える上界である．

下界については，やはり Floyd-Rivest (1976) が次式を証明している：

$$\bar{V}_{\lceil n/2 \rceil} \geq 1.375n.$$

次に，$t$ が定数である場合，

$$\bar{V}_t(n) \leq n + c_1 \ln \ln n$$

$$\bar{V}_t(n) \geq n + c_2 \ln \ln n$$

である．ここで，$c_1, c_2$ は，（$t$ に依存する）適当な正の定数である．上界式は，Matula (1973)，下界式は，Yao-Yao (1982) による．

## 4. ま と め

整列問題が $\theta(n \log n)$ の計算量をもつことはよく知られているが，ここでみたように選択問題は，$\theta(n)$ の計算量をもつ．大雑把には，選択問題は，入力データを一通りながめるだけの手間で解くことができるといえる．このことを利用して，他のいろいろな計算問題を解く算法の設計に，線型時間選択算法が部分手続きとして応用されている．例えば，グラフの最小木 (minimum spanning tree) を求める高速算法がそのようなものである (Yao (1975))．

また，実用的なプログラムという見方からは，**2** でみたように，選択専用の算法は，整列算法を流用するのに較べて格段に速い．

最後に，選択問題に関する研究課題としては，**3** で述べた中央値選択の比較回数の上界または下界の改良がなんといっても最も大きいものである．その他，実用に直接関連する課題としては，最悪の場合でも線型時間で走る算法で，現在の普通の計算機で十分速いプログラムとして実現できるものを開発するというものがある．この場合，作業用記憶場所をあまり使わない (in situ) 算法を作ることがその次の課題となろう．

## 参 考 文 献

数が多いので，主なものだけをあげる．その他のものについては，次の Knuth の本で調べられたい．最近の結果については，雑誌 J. ACM, SIAM J. on Computing, Info. Proc. Letters, J. CSS などに数多く載っている選択問題の論文をもとにして捜されたい．

1) Knuth, D. E.: The Art of Computer Programming, Vol. 3 (Sorting and Searching), Addison-Wesley (1973, 1975 (second printing)).
2) Hoare, C. A. R.: Partition, Quicksort and Find—Algorithm 63-65, C. ACM, Vol. **4**, No. 7, pp. 321-322 (1961).
3) Floyd, R. W. and Rivest, R. L.: Expected Time Bounds for Selection, C. ACM, Vol, **18**, No. 3, pp. 165-172 (1975).
4) Blum, M., Floyd, R. W., Pratt, V. R., Rivest, R. L. and Tarjan R. E.: Time Bounds for Selection, J. CSS, Vol. **7**, No. **4**, pp. 448-461 (1973).
5) Schönhage, A., Paterson, M. and Pippinger, N.: Finding the Median, J. CSS, Vol. **13**, No. 2, pp. 184-199 (1976).
6) Kirkpatrick, D. G.: A Unified Lower Bound for Selection and Partition Problems, J. ACM, Vol. **28**, No 1, pp. 150-165 (1981).
7) Yao, A. C. and Yao, F. F.: On the Average-Case Complexity of Selecting the $k$th Best, SIAM J. on Computing, Vol. **11**, No. 3, pp 428-447 (1982).

（昭和 58 年 2 月 21 日受付）