分　　類　(1)*

淵　一　博**

A. 序　　論

分類（sorting）という操作は，データ処理の重要な部分をなしている．1955年 J.C. Hosken[1] は，「分類は，パンチ・カードを用いる大きな理由である．現在までのところ，それは，磁気テープ，およびその他の場合には高速な組織を使わない大きな理由であることが多かった」と述べている．口真似をすれば，今では，分類が重要なのは磁気テープを用いるからであるといえようか．事実，翌56年には米英それぞれから二編の分類に関する報告が出て[2,3]，両者共磁気テープによる分類を主要な関心事としつつ分類手法一般について述べているのである．

このようにカードから磁気テープへのファイル媒体の転換というような質的変化が起る際に分類の問題が強く意識されるようである．わが国でも“カードかテープか”という論争が続いているけれども，その際分類が争点の一つにとり上げられている．諸外国に比べ約 10 年遅れて出発したわが国の計算機工業が，磁気テープの設置をようやく自己の課題とし始めたという事実を考えるとうなずけないことではない．

しかも，データ処理の実践が，国内の技術の上だけに立たず，同時に先進国から直輸入された装置を使っても行われているというわが国のお定りの特殊事情が，問題をいくらか複雑にしているようである．

A-1　ファイル媒体の推移と分類

データ処理組織の発達は，オフ・ライン装置による個別処理からオンライン装置による集中処理に進んでいるように思われる．それはまたファイル媒体のカード→磁気テープ→ランダム・アクセス・ファイルへの推移に対応しているようである．分類についていえばカードによる分類はオフラインであり，磁気テープでは，オフラインの分類機を使うものと，オンラインで操作するものとがある．時分割同時制御方式の実用化は後者に有利な影響を及ぼすが，これについては後で述べよう．ランダム・アクセス・ファイルまでいけば完全にオンライン操作になるであろう．

A-2　分類とは何か

データ処理における“分類”は，「情報の**項目**（item）を，それに含まれるある**キー**（key）に関する規則に従って，ならべること」と定義されている（ACM Grossary[11] より）．すなわちデータ処理で“分類”と使うときは，“ならべること”に重点があるのである．日常用語としての“分類”は“分ける”ことに重きがある，sorting でも同じニュアンスなので，〔2〕，〔16〕，〔10〕などでは正確を期するには“ならべかえ”（rearrangement, re-ordering）を使う方が適切だ*としているが，sorting の方がよく使われるのは慣習によるのであろう．

A-3　ファイル，項目，キー

普通データ処理はファイルを中心にして行われる．ファイルというのはある目的に沿って集められた記録あるいは項目の集合である．項目というのはある単一の対象に関連する情報をまとめて一単位としたものである．項目の中のある部分がキーとして用いられるが，それは固定しているものではない．用途により項目の中どの部分を使ってもよい．キーは一般に文字の集合であるが，文字にも照合順序（collation sequence）が与えられていて，それにより大きい順または小さい順にならべるのが普通である**．

分類がなぜ必要かということ，どの程度必要かということはデータ処理全体の観点からみる必要がある[10]．逐次式ファイル（sequential file—その代表は磁気テープである）ではその性質上，もしデータが順序にならんでいなければ，その取出しが困難（不経済）になる．また親ファイル（master file）と同じ順序に取引ファイル（transaction file）の項目が並んでいなければ，その処理の際，項目を合せるのに行きつ戻りつせ

---

* Sorting 1, by Kazuhiro Fuchi (Electrotechnical Laboratory, Tokyo)

** 電気試験所電子部

* rearrangement by sorting（分類によるならべかえ）という用例がある．これは分ける方の意で後に述べる radix sorting のことである．

** 以下の節で例解をするときは，キーは10進数であるとし小さい順にならべるものと仮定する．

ねばならず，非能率的である．逐次ファイルでは順序どおりに処理するということによってのみその最大の欠点（呼出し時間が長いということ）をかくせるからである．

他方，即時呼出しファイル（random access file）では，データの呼出しはほぼ均等であるから，分類の必要性は減るわけであるが，それでも能率的なデータ取出しには順序に並んでいることが望ましくなってくる．

分類の操作は一般にかなり時間のかかるものであるから，いつでも分類しておけばよいということにはならない．処理の他の部分での得と分類時間の損をうまく調整しなければならない．例えば，分類をいつどれくらいまとめて行うのがよいかということが問題になる[9]．

B. 分類の基本操作

最初に分類をするために今まで考案された基本的な操作を説明しよう．これらの操作は特定の組織に固有のものではない．しかし逆に特定の組織をとれば，その特殊な性格によって，選べる方法が限定されるし，有利，不利もでてくるものである．磁気テープ組織はその制限が最もきびしいものである．

B-1 比較の回数と情報量[16,15]

項目の配列についてはそのエントロピー(情報量)を考えることができる．簡単化してのべれば次のようになる．今項目数がN個あるとし配列についての情報は何も得られていないとする．ある項目がどの場所にあるかを知るには$\log_2 N$ビットの情報が要る（N個の場所を区別できなければならないから）．各項目に同じ情報が要るもの* とすれば全体で$N\log_2 N$ビット要る．比較により1ビットの情報が得られるから必要回数は，$N\log_2 N$回である．情報処理装置の情報を扱う速度はほぼ一定だから，分類をするのに要る時間がほぼ

$$T\sim N\log_2 N$$

の方法が望ましいといえる（特に項目数の多いとき）．しかし項目数があまり多くないときは比例常数の関係からN^2に比例するものでも時間的に有利になることがあり得る．

すでにある程度配列について情報がえられているとき，または部分的に望みの順序に整列しているときはエントロピーが減少しているわけで，分類に要する操作の数は減る．

項目の分布についてその確率的性質が分っている場合（上では一様分布を仮定していたわけである．この場合が計算は最も容易である），エントロピーが計算できて，分類手法の良し悪しを判定する目安を立てることができる．正確な情報理論的考察を経ないにしても，分布の性質を知り，それを意識的に利用することは非常に大事である．これは与えられたデータ処理の性質を洞察し，最も能率的な組織を設計することに通じる．

B-2 基数法

基数法（radix sorting, divergent sorting, あるいは rearrangement by sorting）は最もよく知られた方法である．パンチ・カード分類機で主として使われるのはこの方法である．

キーがp進法n桁からなるとする（キーが文字からなるときはその照合順序を考え適当な修正を加えればよい）．この方法で基本的なのは，キーのある桁に注目して，項目をp個の類に“ふり分ける”操作である．

この操作を繰返して分類を遂行するわけであるがそのやり方には大きく分けて二つのものがある．

その一つは MSD* 法とも呼ぶべきもので，まずキーの MSD についてみて，項目をp個の類に分ける．次いでその各類について，2番目の桁についてみて“ふり分け”をする．次はその類（第二次の類）について“ふり分け”をする．これを LSD まで続けていき，最後に“回収”すれば，全項目が整列する．

他は，LSD 法とでも呼ぶべきもので，まずキーの LSD についてみて“ふり分け”を行う．次いで，このp個の類を“回収”し順に重ねる．これが一つのサイクルを形成する．次のサイクルは下から2桁目について行う．これを MSD まで繰返すわけであるが，注意を要するのは，それまでのサイクルでできた順序をくずさないように操作しなければならないことである（例えばカード分類機では機構的にそうなっている）．n回のサイクルの終りには，全項目が整列している．

基数法では，各項目についてn回の“ふり分け”が必要である．そこでこの方法による時間は，全項目数をNとすると，nNに比例することになる．一方，キーが重複しないとすれば，$N\leq p^n$である．キーの（可能な）数p^nに対してNがはるかに少いとき項目の密

---

* 順列の数を考えて情報量を $\log N!$ とすれば上の概算よりやや少くなるが主要な項はやはり $N\log N$ である．

* MSD は最上位の桁（most significant digit）の略，LSD は最下位の桁（least significant digit）の略である．

度が低い（thin population），N が p^n に近いかそれより多いとき（項目が重複しているときにありうる）密度が高い（thick population）という表現をすることにすれば，基数法は密度が高いときはよいが，密度が低いとき不利であるといえる．まとめて，

$$T\sim nN,\ n\sim\log_p N$$

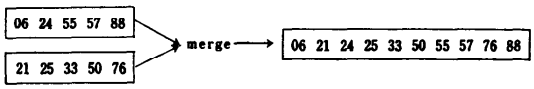
と書いておこう．

次に問題なのは項目の分布にかたよりがあるときである．すなわち p 個の類に分けたとき，ある類に属する項目が非常に多くなりすぎるときである．各類を入れるスペースには一応制限があるわけだから，その制限を越すおそれがあるときは，その対策を立てておかなければならない．もし分布についてあらかじめそういう事態が起ることがわかっていれば“ふり分け”方を修正することを考慮しておくか，あるいはあふれが起ったときの措置を講じておかなければならない．

記憶装置内でこの方法を使うときは，“ふり分け”のためにほぼもとのデータの量と同じ，N項目分のスペースが要ることも注意しておくべき点の一つである．

B-3 併合法（sorting by merging）

この方法の基本となる**併合**（merge）の操作というのは，いくつかの整列した列から，それらを一緒にした一つの整列した列をつくりだすことである．この整列している列のことを**連糸**（strilg）とも呼ぶ．すなわち p 個の連糸からそれらを合せた1個の（長い）連糸をつくり出すことであるといえる．この場合 **p 重併合**（p-way merge, p-fold merge）と称する．2重併合の例を**第1図**に示す．



第1図 2-way merge

p 重併合の場合，入力の p 個について比較し，そのうち最小のものを出力として次々に出す．出力された項目についてはそれが属していた連糸から次々に補充をしていけばよい．この点で併合の操作は逐次式ファイルに適している．

この併合を使った分類法は次の二つの手順からなる（以下はもっぱら2重併合法についてのみ述べる）．N 個の項目はいくつかの連糸からなるが，それを，

（2） 二つのブロックに分ける．

（2） 両ブロックから連糸を読んで新しい連糸を形成していく．

これを繰返せば，連糸の長さはだんだん長くなり遂にはNに達する．すなわちN項目全部が一つの連糸になるがこれは全体が整列したことを意味する．

連糸の考え方には二つある．一つは，最初各項目が長さ1の連糸であると考える．これによると上に述べたサイクルを n 回繰返せば，連糸の長さは 2^n でありしたがって，

$$n=\lceil\log_2 N\rceil$$

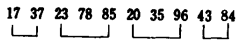
$\lceil x\rceil$ は x より小でない整数

回のサイクルで分類が完結する（straight merge）．

しかし，そう考えより，すでにデータ内に存在する（自然な）連糸を利用する方が自然で賢明な考えである（第2図）．すなわち，最初 S 個の連糸があるとすると1回のサイクルで連糸数が半分になるから，

$$n=\lceil\log_2 S\rceil$$

回で完了する．実際のデータではすでに部分的な分類がなされていることも多いから，この方がずっと有利である（probabilistic merge）．



第2図 自然な連糸

連糸の分布については，一様な確率分布の場合について非常に詳しい計算がなされている[12,13,14]．これによると連糸の長さは平均2である*．したがって一様な確率分布の場合でも，ほぼ，

$$n=\log_2 N-1$$

である．

一回のサイクルに要する時間は項目数に比例するから，この方法で分類するのに要する時間は，

$$T\sim N(\log_2 N-1)$$

である．これは項目数にのみ関係し，キーの寸法にはよらない．

この方法を内部記憶装置内で使うときには $\frac{1}{2}N$ 項目分のスペースが要る．

なおこの併合法については磁気テープ分類法の節で詳細に扱う．

B-4 選択法（sorting by selection）

これは簡単で，N項目中最小のキーを持つものを選び，それを別の場所に移す．次に残りの $N-1$ 項目の

* これは極めて初等的な考察でもうなずける．つながるか，つながらないかの確率は半々だから．

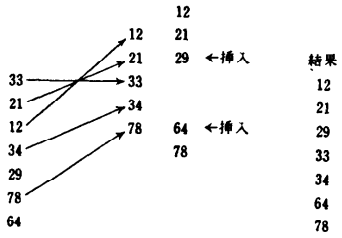
中の最小を選び，前に移したものの次に移す．これを繰返すわけである．最小のものを見出すために必要な比較回数は，N, $N-1$, ……であるから全体として

$$\frac{1}{2}N(N-1)$$

となる．

B-5 挿入法（sorting by insertion）

この方法は，すでにできた列の中の適当な場所をさがしてそこにその項目を挿入する方法である．すなわち，今までのどの項目より小なら，その列の上に加え大ならば下に加える．また中間であればそれより大または小の項目を一つづつずらして，空いた所に挿入する（**第3図**）．



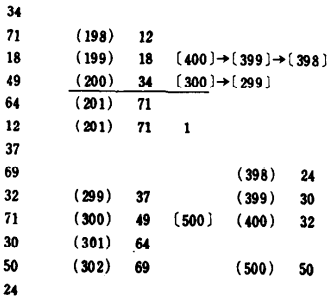
第3図　挿入法

この場合の比較および移動の回数は，乱数の場合，今まである項目の約半分と考えてよいから全体で

$$\frac{1}{4}N(N-1)$$

となる．

この場合，実際に移動をさせないやり方もある[10]．これは nesting 法とも呼ばれるが，**第4図**に例を示すように，挿入すべきところに“張出し”をつける．例えば49は34と71の間であるがそれを（300）に入れ張出した先をインデックスとして(200)の所に〔300〕と入れておく．このようにして実際には移動を行うことなく最後のものまで行き，それからインデックスを



第4図　Nesting 法

頼りにして全データを回収する．

挿入法は，別にN項目分のスペースがなくても，それ自身の中の交換で実行できるから1項目分だけのスペースですむようにすることができる．

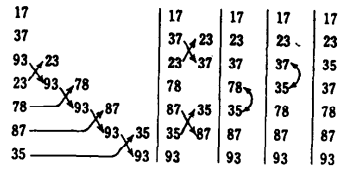
B-6 交換法（sorting by exchanging）

この方法は，互に隣り合った2項目を比較し，順序が逆であればそれらを交換するという操作からなりたつ．これを一端から他端まで続ける．何度も繰返し，交換が一つも起らないようになれば全項目が分類されたことになる（**第5図**）．

この交換法も一項目分のスペースがあればよい．時間はほぼ，

$$T\sim N^2$$

である．



第5図　交換法

この方法はこれだけでは操作が簡単であるけれども，かなり時間のかかるものである．これには C-3 で述べるような改良法がある．

C. 内部分類

内部分類（internal sorting）とは，主記憶装置内部で行われる分類のことである．主記憶装置内のデータについては呼び出しが即時（random access）と考えてよいこと，アドレスによってデータが扱え，内部だけの操作であるからプログラミングも比較的容易であるなどの利点がある．一方，容量が限られている*ため，大きなデータを数多く扱うことができないのが欠点である．

そこで内部分類においては，必要なスペースの量というのが，どの分類法を選ぶかについて大きな要因となってくる．また項目数が限られているため，所要時間が $N\log N$ に比例するものの方がよいとは必ずしも云えない．

さて，計算機の性質（万能であるという）から，B に述べた方法はすべて内部分類に使うことができる．あるいは内部分類で方法のテストをすることもでき

* 計算機の記憶容量は（プログラムの分も含めて）せいぜい数万字であるのに対し，事務データ処理のファイルが数百万字に及ぶのは珍しくない．

る．逆にプログラムによつて自由にアイデアを実行に移すことができるので，内部分類において最も種々な方式が考案されている．

しかし実際のデータ処理では，内部分類では間に合わないので，どうしても外部記憶装置を使わなければならない(オフラインでの分類は別にして)．が外部記憶装置を使うにしても，内部記憶を介して使うのであるから，内部分類を適当に組合せることにより，能率を上げることが可能である．

そのような補助的役割の他に内部分類について発展させられた手法には別の意味がある．というのは，現在ファイル媒体として最もよく使われている（わが国ではこれから使われようとしている）磁気テープは，その特異な性格のため，採用される手法もおのずから限定されざるを得ないが，次の段階のファイル媒体としての即時呼出しファイルに進めば，それらは再び大きな意味を持つことになるだろうということである．これについては後でも少し触れたいと思う．

以下では幾分特殊な，というよりいくつかの原理を組合せたような内部分類法について述べることにする．

C-1　基数交換法 (radix exchange)[5]

この方法は基数法と交換法の特徴をとり入れたものである．一種の2分法を用いているので，キーは2進数で表わされていると考える（2進法または，語の中のビットに関する演算のできる機械は有利である)．

原理を**第6図**で説明しよう．まずキーの MSD を頭からみて，初めて1になるところまで進む．次にやはり MSD を下から上にみていって初めて0が現われるところまで進む．ここで両者を交換する．次にまた1が現われるまで下に進み，他方0が現われるまで下から上に進む．ここでまた交換をする．上からみていった点と下からみていった点がぶつかったところでパスが終る．

キー(10進)	Pass No. 0	Pass No. 1	Pass No. 2	Pass No. 3	Pass No. 4	結果
7	0111	0111	0010	0001	0000	0
5	0101	0101	0000	0000	0001	1
1	0001	0001	0001	0010	0010	2
9	1001	0011	0011	0011	0011	3
0	0000	0000	0101	0101	0100	4
4	0100	0100	0100	0100	0101	5
8	1000	0110	0110	0110	0110	6
2	0010	0010	0111	0111	0111	7
6	0110	1000	1000	1000	1000	8
3	0011	1001	1001	1001	1001	9

第6図　基数交換法

このパスの結果は，MSD が0のものが上に，1のものが下に集まる．これは基数法のふり分けと同じである．ただし場所は別のところでなくスペースは交換のためのものが要るだけである．また分布がかたよっていても総計は変らないのだから何ら支障はない．

次は MSD 0のブロックについて，上から2番目のビットに着目して同じことを行う．以下 B-2 で述べた MSD 法の要領で進んでいく．

この方法の利点は

（1）　分類のために別のスペースがあまり要らない．

（2）　分類時間が項目の数に関係し，キーの長さにあまり関係しない（密度が低い時も)．

（3）　分類時間が分布にあまり依存しない．

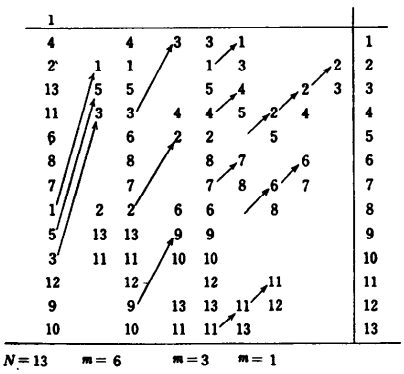
これについては，スペースに対する要求が似たものとして挿入法との比較がある[5]．

	基数交換	挿入
乱配列	82	1457
上の逆	81	1524
整列	63	4
逆配列	65	2974

数字は比較だけの意味しかない．

C-2　Shell の交換法[6,7]

これは交換法の直接の改良である．挿入法のところで述べたように，ある項目が順序の中に入る位置の確率的な平均をとれば中央である．交換法の場合のようにすぐ隣であるという確率は小さい．これが交換法でステップ数の多くなる原因である．そこで Shell は，まず間隔が項目数のほぼ半分であるもの同志を比較し交換し，順次比較するものの間隔を狭めていくことを考えた．その実例を**第7図**に示す．



第7図　Shell の方法（1）

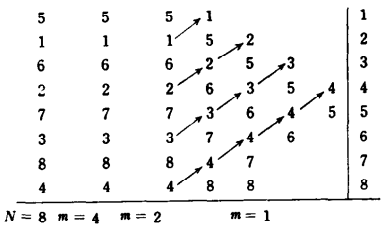
比較の間隔のとり方は次のようにする．i 番目のパスの間隔を $m^{(i)}$ とすると，

$$\begin{cases} m^{(i+1)}=i.p.\left(\frac{1}{2}m^{(i)}\right) \\ m^{(0)}=N \end{cases}$$

（$i.p.$ は整数部* の意）

例でいえば $N=13$ であるから $m^{(2)}=6$，すなわち，1番目と7番目，2番目と8番目，……と比較していく．

このやり方で困るのは $N=2^n$ のときである．このとき，奇数番目の組と偶数番目の組とは，最後の $m=1$ になるまで比較されない．その模様を**第8図**で示す．



第8図　Shell の方法（2）

そこで Frank & Lazarus[7] は，m をつねに奇数に選ぶことを提唱している（**第9図**）すなわち，

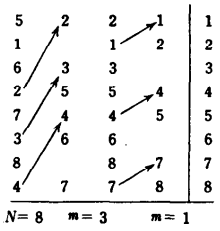
$$m^{(i+1)}=2\,i.p.\left(\frac{1}{4}m^{(i)}\right)+1$$

（7）では，比較の回数を計算し，結局次のような m を選ぶのが最もよいといっている．

$$m^{(i+1)}=2\cdot i.p.\left(\frac{1}{4}m^{(i)}\right)+1 \qquad m^i\leqq 15$$

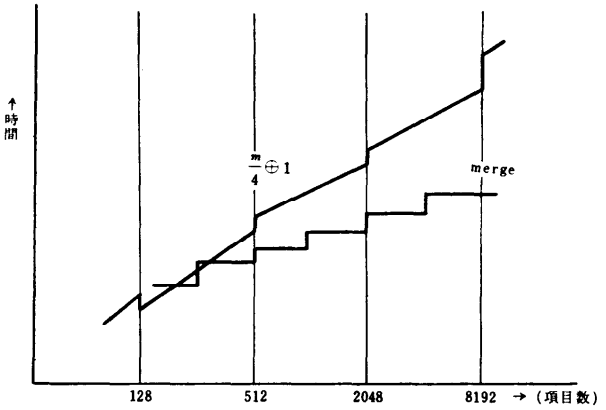
$$m^{(i+1)}=2\cdot i.p.\left(\frac{1}{8}m^{(i)}\right)+1 \qquad m^i>15$$

すなわち初めのうちは m の減し方を大きくするのである．この方法を併合法と比較したものを**第10図**にあげる．



第9図　改訂 Shell の方法

* integral part of



第10図　Shell の方法の速度

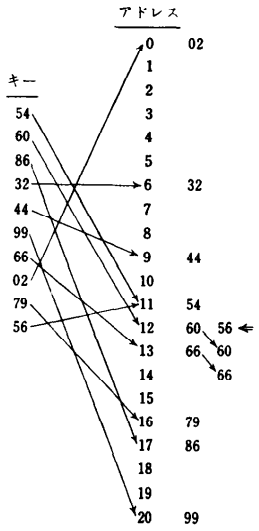
C-3 アドレス計算法（sorting by address calculation）[4]

この方法は挿入法と基数法の特徴をそなえている．後者についていえば基数法で基数を非常に大きくとったことに当る．

まず M 項目（$M\geqq N$）分のスペースを用意する．最初の項目のキーを K_1，キーの最大値を K_m

$$A_1=\left(M\times\frac{K_1}{K_m}\right)+\alpha$$

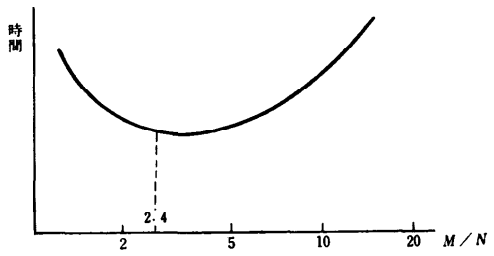
を計算する．そしてこの項目を A_1 番地から入れる．α は用意したスペースの基準番地である．上と同じ式で次々に各項目についてのアドレスを計算し，そのアドレスから項目を入れていく．もしその場所にすでに他の項目が入っていれば挿入法の要領によりその近く



第11図　アドレス計算法

の適当な場所に挿入していく．このようにして全項目について計算し，移し終ったら，作業スペースの頭から順に項目をひろって回収する（**第11図**）．

上にあげたような式をアドレス計算式という．今は一次式を使ったのであった．これはあらかじめ項目の分布が分っていれば，それに合った（一次式でない）式を立てることもできよう．しかし単純さの点でたいてい一次式が使われる．もし分布が一様であれば一次式によってほとんど挿入なしにふり分けが完了する．



第12図 比 率

この方法でMを大きくすればするほど，項目がぶつかる確率は少くなり，したがってふり分けの時間は少くなる．他方Mを大きくすると回収する時間が長くなる．MとNの比を変えて実験した結果は〔4〕によれば**第12図**のようになる．$M/N \fallingdotseq 2.4$ のあたりに時間が最小となる点がある．〔8〕では一様な確率分布という仮定を使って計算をしているが，ほぼ同様な結果が得られている．

参考文献

1) J.C. Hosken, Evaluation of Sorting Methods, EJCC, Nov. 1955
2) D.W. Davies, Sorting of Data on an Electronic Digital Computers, PIEE, March 1956
3) E.H. Friend, Sorting on Electronic Computer Systems, JACM, July 1956
4) E.J. Isaac and R.C. Singleton, Sorting by Address Computation, JACM, July 1956
5) P. Hildebrandt and H. Isbitz, Radix Exchange—An Internal Sorting Method for Digital Computer, JACM, Apr. 1959
6) D.L. Shell, A High Speed Sorting Procedure, Comm. of ACM, July 1959
7) R.M. Frank and R.B. Lazarus, A High Speed Sorting Procedure, Comm of ACM, Jan. 1960
8) I. Flores, Computer Time for Address Calculation Sorting, JACM, Oct. 1960
9) A.S. Douglas, Computer and Commerce 2. Computer Journal, Oct. 1958
10) A.S. Douglas, Techniques for Recording of, and Reference to Data in a Computer, Computer Journal, Apr. 1959
11) Standards, Comm. of ACM, Oct. 1958
12) 木沢誠：テープによるならべかえの理論 I，電試彙報，1959年11月
13) 飯島泰蔵：テープによるならべかえの理論，電試彙報，1960年4月
14) 藤井正友：テープによるならべかえの理論，重複を許す場合について)，電試彙報，1960年5月
15) 喜安喜一，池野信一：分類の情報理論的考察，電気通信学会インホメーション理論研究専門委員会資料，1960年1月19日
16) D.A. Bell, The Principles of Sorting, Computer Journal, July 1958