Multiway replacement selection sort with dynamic reservoir

T. C. Ting and Y. W. Wang

School of Information and Computer Science, Georgia Institute of Technology, Atlanta, Georgia 30332, USA

An improved multiway replacement selection sorting algorithm is proposed for generating longer ordered strings. The method has employed both the notions of 'degrees of freedom' and that of 'natural selection'. A dynamic reservoir is proposed which makes fuller use of main memory space of the computer. Extensive simulations were conducted and the results are presented and discussed.

(Received August 1975)

In general, two phases are involved in sorting a file whose size is much larger than the capacity of the main memory of the computer being used. An initial internal sorting phase produces ordered strings of records which are externally merged during a second sorting phase into successively larger strings until a single output string remains.

The number of strings produced by the internal sorting phase is one of the important factors in determining the number of passes necessary for the external phase, which in turn is directly related to the processing time. The external phase uses slow peripheral memory devices, and therefore it usually dominates in terms of total sorting time. Thus, it is desirable to produce strings as long as possible in the internal phase, so that a fewer number of strings are given to the external phase. The multiway replacement selection sorting algorithm (Friend, 1956) is one method widely used for this purpose. The expected length of the strings produced by that method approaches $2p$ when the main memory of the computer used is capable of holding p records. The idea of 'degrees of freedom' was subsequently suggested to improve on replacement selection (Dinsomore, 1965) and it uses m buffers for an m-way merge to produce a single string from more than m pre-ordered blocks. It starts with m blocks, and new blocks can be continually included into a current string so long as the key value of the first record is greater than or equal to the key value of the last record of the just-exhausted block. More recently, a method called the 'natural selection sort' was developed which employs an external reservoir for storing dead records that can not be included in the current string and which resulted in producing longer sequences than the simple replacement selection algorithm (Frazer and Wong, 1973). The length of the strings produced by the method is a function of the size of the reservoir R. For $R = p$, the expected length is ep, where e is the base of natural logarithm (2·718 . . .). For greater values of R, the length of the string is longer. By varying the value of R, one can expect strings of arbitrary preassigned length. This method is an evolutionary improvement over simple replacement selection (Knuth, 1973).

The present method has employed both the notion of 'degrees of freedom' and that of 'natural selection' for further improving the replacement selection method. A dynamic reservoir is proposed to handle the buffers of the input blocks and to store the dead records during the selection process. The method yields longer strings and produces fewer rejected records than does the natural selection algorithm. Furthermore, by including only the keys of the records in a selection tree placed in high speed main memory (thus leaving the whole records to remain in the reservoir), the method permits full utilisation of the main memory space of the computer for producing even longer strings, since the size of the key is but a fraction of the whole record. Therefore, the value of p is greatly increased since the value of p now represents the capacity of the computer's memory which is capable of holding p keys instead of records. However, the present method does require an external reservoir on a direct access medium and the minimum size of the reservoir R is equal to $2p$ records.

Extensive computer simulation has shown that for $R = 2p$ records the expected string length is $4{\cdot}2p$ records when the input data is randomly distributed. For a greater value of R, a longer length can be expected. The expected string length X is a function of parameter R, and the results are expressed graphically in **Fig. 1.**

Description of the method

The present method is an improved replacement selection algorithm which takes advantage of both the idea of 'degrees of freedom' and the concept of 'natural selection'. It uses an external, dynamically organised reservoir on a direct access device for buffering m ordered input data blocks and storing the dead records. A selection tree is structured in the main memory of the computer for performing an m-way merge on the buffered input data blocks. The keys and locations of the first records of input blocks are included in the nodes of the tree. The tree is arranged in a 'heap' which is used to facilitate an efficient merge selection.

Initially, m input data blocks are entered to load the selection tree and the reservoir. The ordered input data blocks are stored in the reservoir in the form of a linked list. The key value and the location of the first record of each input data block is placed in a node of the selection tree. Thus, each node is acting as the header of the list. The number of input data blocks m normally should be equal to the size of the tree p which is the capacity of the main memory space. However, m may be less than p when the input data blocks are long and the reservoir is filled before the tree is fully loaded. This condition may happen when the input data is partially sorted. Therefore, m may be less than or equal to p.

A 'heap' is arranged after the selection tree is loaded and an m-way merge is performed by using the tree selection method. The node on the top of the selection tree is selected and the record immediately linked by the node is output. The key value and the location of the next record in the same input data block are entered to the just-vacated node. The selection tree is then readjusted to maintain a valid 'heap'. The process repeats until one of the input data blocks is exhausted.

A new input data block is read into the reservoir. If the key value of the first record of the newly entered block is greater than or equal to the key value of the just-output record the data

Dr. Ting is now on leave at National Science Foundation, Washington DC, and Mr. Wang is now at Chung-san Research Institute, Taipei, Taiwan, Republic of China.

Downloaded from https://academic.oup.com/comjnl/article/20/4/298/393615 by guest on 19 April 2024

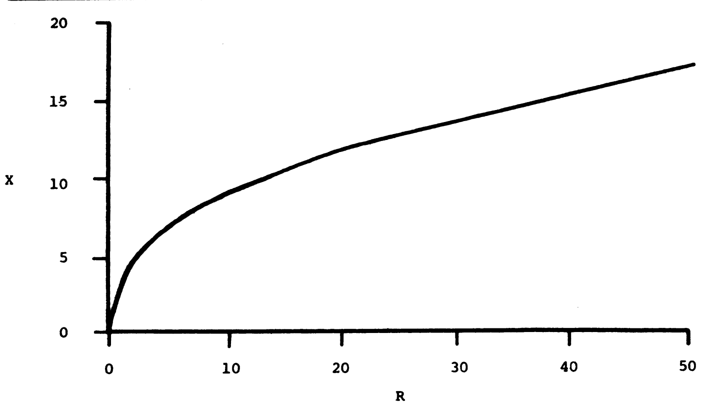


Fig. 1 Expected sequence length X as a function of parameter R

block is appended to the selection tree. Otherwise, the beginning record of the data block is considered a dead record, and it is linked to a list of dead records in the reservoir. The next record is treated as the first record and it is tested to determine whether the remaining data block can be appended to the selection tree. The same process repeats until either the remaining data block is appended to the tree or the entire data block is placed on the dead record list. In the latter case, another input data block is entered. The process continues until the reservoir is full, and at this point the size of the selection tree is reduced by one. The selection process continues with an $(m - 1)$ way merge, and eventually terminates when the selection tree is empty.

After a new data block is appended to the selection tree, if $m < p$ and the reservoir is not full m will be increased by one and an additional input data block will be entered. This input process will be continued until the $m = p$ or the reservoir is full. The selection process will then be continued.

During the next run, for the generation of the next string, the records in the dead record list are used as input data until all of them have been processed. These records either are included in the output string or are again returned to the dead record list. The additional input data are then read and are either appended in the output string or rejected until the reservoir is once again full, the selection tree is once again empty, and so on. This process continues until finally the whole process ends when the input data file is completely exhausted and the reservoir is completely empty.

Input data blocks

The file to be sorted can be regarded as a set of records in natural sequences. There are two types of natural sequences of records—namely, ascending and descending sequences. An ascending sequence is a set of linearly ordered records in which the key value of the preceding record is less than or equal to the key value of the current record. Similarly, a descending sequence is a set of linearly ordered records in which the preceding record is greater than or equal to the current one. The type of sequence is actually determined by the order of the first two records of the sequence. A sequence is terminated when the subsequent record indicates an order other than the present one. The natural sequences are recognised and switched to ascending sequences during the input function of the method. These natural sequences are treated as input data blocks and they are stored as linked lists in the buffer areas in the external reservoir on a direct access memory device. A link reference is added at the end of each record as the link to the next record. A null link reference is placed within the last record as the link to no record.

The minimum length of the input data block is two records. The length of the input data block can be long when the input data is partially sorted, and the present method takes advantage of the phenomenon.

The structure of the selection tree

The selection tree is a 'loser-oriented binary tree' which can be viewed as a priority queue of the smallest item out first (Knuth, 1973). Each node of the tree consists of two fields, the key and the location reference. By using the location reference, each node can be viewed as a header of the input data block with the key value and the location of the first record included in the node.

The tree is allocated in consecutive main storage spaces. The relationship between the father node and its sons are determined by the locations in storage. The father node of the node located in L is placed at $\lfloor L/2 \rfloor$ location and the two sons of the node at L are positioned at $2L$ and $2L + 1$. The tree is arranged

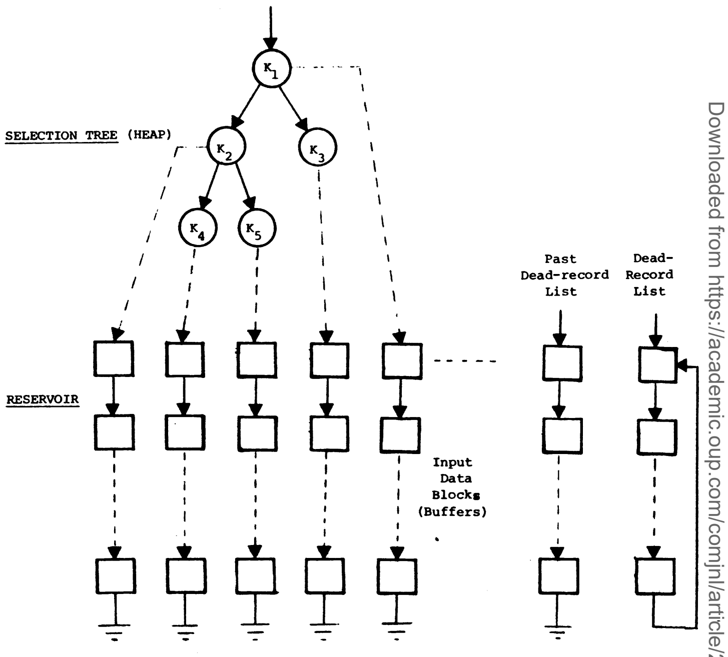


Fig. 2 The data structures of the selection tree and the reservoir

Table 1 The tabulated results

R	X	Y
1	2·24	0·00
2	4·17	0·00
3	4·96	0·14
4	5·55	0·51
5	5·92	1·23
6	6·31	1·56
7	6·76	2·16
8	7·16	2·74
9	7·52	3·02
10	7·96	3·37
11	8·23	3·79
12	8·67	4·11
13	8·91	4·34
14	9·24	4·88
15	9·67	5·68
16	9·88	5·94
17	10·02	6·78
18	10·42	7·14
19	10·73	8·02
20	10·97	8·89
25	12·12	12·03
30	13·21	16·47
40	14·82	24·36
50	16·68	32·43

Downloaded from https://academic.oup.com/comjnl/article/20/4/298/393615 by guest on 19 April 2024

so that the key value K of the father node is always smaller than or equal to the key values of its two sons. In other words, the tree is organised so that

$$K_{\lfloor j/2 \rfloor} \leqslant K_j \text{ for } 1 \leqslant \lfloor j/2 \rfloor < j \leqslant m$$

where m is the size of the tree. It implies that the node with the smallest key value appears at the root of the tree,

$$K_1 = \min(K_1, K_2, K_3, \ldots, K_m) .$$

It also implies that $K_1 \leqslant K_2$, $K_1 \leqslant K_3$, $K_2 \leqslant K_4$, $K_2 \leqslant K_5$, . . ., etc. Initially, the keys of all first records of the input data blocks are loaded into the nodes of the tree in sequential manner. An efficient algorithm which was suggested by Floyd (1964) is used to create the selection tree. A top down selection procedure is used to perform an efficient m-way merge by utilising the main concept suggested by Williams (1964) in his 'heapsort'.

The organisation of the reservoir

An area of direct access storage space capable of holding at least $2p$ data records is used as an external reservoir for the method, where p is the number of keys that can be handled in the main memory of the computer used. Initially, the entire reservoir is organised as a list of available space.

Input data blocks are stored in linked lists with their headers placed in the nodes of the selection tree. These linked lists constitute the input data buffers.

Two additional lists are organised for handling the dead records; one for storing the current dead records and one for holding the past dead records. The list of past dead records is used as input during the current run and it is organised in the form of a simple list. The current dead record list is used to accept the rejected records during the run and it is organised in the form of a circular list. The reason for the circular list is that it is convenient to put an entire circular list onto the past dead record list.

The relationship between these data structures is illustrated in **Fig. 2.**

The algorithm

The following flowchart was chosen to illustrate the algorithm for the present purposes. However, it does not represent an optimal implementation of the algorithm, since many detailed steps have been omitted so that the main ideas can be easily understood.

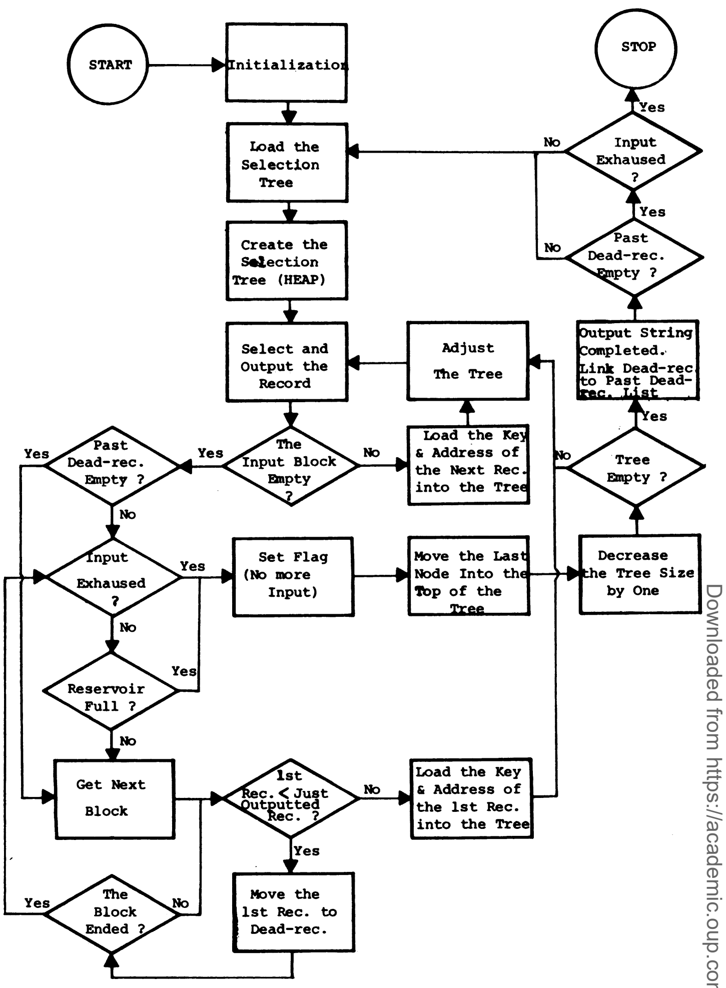
Experimental results

An extensive computer simulation was conducted for testing the method. A large number of pseudorandom integer numbers were used as input data. The values of m were $m = 2^k$ for $k = 5, 6, 7$ and the values of R were $R = m, 2m, 3m, \ldots$ $20m, 25m, 30m, 40m, 50m$. In each run, more than 100 output sequences were produced. Average sequence length was computed for the 5th through 100th sequences in order to eliminate the effect of initial convergence to the limiting distribution. The results are tabulated in **Table 1.** The expected sequence length $X =$ the average length$/m$. When $R < 2m$, there are some records to be rejected more than one time before merging into the output sequence. A count of the number of records which are again returned to the dead record list was kept during each run. The average number of returned records Y is also tabulated in Table 1, and is expressed in Y equal to average number of returned records$/m$. It is of interest to note that for $R < 2m$, the expected length X can be approximated

$$X = R + 2 - Y .$$

Conclusions

The present method is an improved replacement selection which produces longer sorted sequences than other methods.



START
Initialization
STOP
Load the Selection Tree
Input Exhausted ?
Create the Selection Tree (HEAP)
Past Dead-rec. Empty ?
Select and Output the Record
Adjust The Tree
Output String Completed. Link Dead-rec to Past Dead-rec. List
Past Dead-rec. Empty ?
The Input Block Empty ?
Load the Key & Address of the Next Rec. into the Tree
Tree Empty ?
Input Exhausted ?
Set Flag (No more Input)
Move the Last Node Into the Top of the Tree
Decrease the Tree Size by One
Reservoir Full ?
Get Next Block
1st Rec. < Just Outputted Rec. ?
Load the Key & Address of the 1st Rec. into the Tree
The Block Ended ?
Move the 1st Rec. to Dead-rec.
Yes
No

Fig. 3 **The flowchart of the algorithm**

It takes advantage of the 'degree of freedom' concept, and it avoids dead records by using a dynamic, external reservoir. However, it requires extra processing time for handling the additional I/O functions of the reservoir. Nevertheless, as direct access storage devices achieve higher and higher performance in accessing speed and flexibility, the cost for the additional processing time is minimised. Furthermore, the time required for the generation of longer initial sorted sequences is worth spending, since it reduces overall external sorting time by reducing the time needed for the second phase of an external sorting procedure—the slower of the two phases.

The minimum length for an ordered natural sequence in an input file is two records. The average length of the input blocks is then always greater than or equal to two records per block, and the latter condition exists when the input data is perfectly random. The present method uses m degrees of freedom which merges at least m natural input sequences on to the output sequence, and so the resulting output is always more than $2m$ records. Any non-random input data will increase the average length of the output sequences produced by the present method. The m degrees of freedom coupled with the reservoir permits any subsequent record with a key value not less than the mth largest key value of the input data blocks that precede it to be appended to the current output sequence before the reservoir is full. The expected length of the output is a function of the parameters of the sizes of the selection tree and the reservoir, m and R, and of the randomness of the input data file.

The method uses a loser-oriented binary selection tree for performing an efficient internal merge. Only the keys plus the location pointers of the leading records of the input data

Downloaded from https://academic.oup.com/comjnl/article/20/4/298/393615 by guest on 19 April 2024

blocks are included in the nodes of the selection tree, instead of the whole records. In almost all sorting applications, the size of the node will be only a fraction of the size of the whole record, and so the method permits effective use of computer's main memory. It makes full advantage of the increasing capacity and speed of direct access storage when limited main memory capacity is available. Such situations are commonly found in multiprogramming environments and in minicomputer applications, for example. The ability to change the size of the reservoir to vary the expected length of the output sequences offers greater flexibility for fully utilising memory space for different kinds of external sorting applications.

References

Dinsomore, R. J. (1965). Longer Strings from Sorting, *CACM*, Vol. 8, No. 1

Floyed, R. W. (1964). Algorithm 245: Treesort 3, *CACM*, Vol 7, No. 12

Frazer, W. D. and Wong, C. K. (1972). Sorting by Natural Selection, *CACM*, Vol. 15, No. 10

Friend, E. H. (1956). Sorting on Electronic Computer Systems, *JACM*, Vol. 3, No. 3, July.

Knuth, D. E. (1973). *The Art of Computer Programming;* Vol. 3, Sorting and Searching, Addison-Wesley, Reading, Mass.

Williams, J. W. J. (1964). Algorithm 232: Heapsort, *CACM*, Vol. 7, No. 6

Book reviews

The ANSI/SPARC DBMS Model, edited by D. A. Jardine, 1977; 225 pages. (*North Holland*, US$24.00)

Systems for Large Data Bases, edited by P. C. Lockemann and E. J. Neuhold, 1977; 224 pages. (*North Holland*, US$24.50)

These books record the proceedings of two conferences dealing with data base management systems. The first reports the Second SHARE working conference on Data Base Management Systems, held at Montreal in April 1976; the second reports the Second International IFIP conference on Very Large Data Bases, held at Brussels in September 1976.

It would be wrong, of course, to infer from the titles given to the proceedings that the Montreal conference dealt exclusively or comprehensively with the ANSI/SPARC model, or that the Brussels conference concentrated on problems arising from very high data volumes. Perhaps predictably, both conferences had contributions on access control, end-user interfaces, the conceptual model/schema, and the various competing data models. On the last of these topics, the neat Montreal tutorial paper by Tsichritzis and Lochovsky is marred (but not rendered unintelligible) by the omission of all the diagrams referred to in the text; Tsichritzis appears again at Brussels as co-author of a comprehensive and concise taxonomy of data models.

The Montreal proceedings open with an expository paper on the ANSI/SPARC model by Beatrice Yormark and include considerable treatment and discussion of the three schema level approach, with particular emphasis on the nature, function and form of the conceptual schema. In addition to the topics already mentioned there is a useful and readable account by Frank Manola of the Codasyl DDLC's activities since the 1973 Journal of Development, indicating the current state of the language specifications and future directions for the DDLC's work. Each paper is accompanied by an edited version of the discussion which followed it; panel discussion sessions also are included.

With fifteen papers to Montreal's nine, the Brussels conference was able to range more widely to include, *inter alia*, contributions on a methodology for choosing an efficient internal schema; DIAM II's general model for access methods; the development of a special purpose hardware processor for performing the primitive operations on relations; communication of data between different DBMSs; the conversion of applications programs as a consequence of data base changes; user extensions to the Peterlee Relational Test Vehicle; a deductive capability for data management; and the British Library bibliographic retrieval system, MERLIN. Discussion reports are not included in the proceedings.

Typographically, the Montreal proceedings have been edited into a uniform style and they bristle with (mostly trivial) misprints; the Brussels proceedings have been prepared directly from author supplied copies.

A number of the contributors are well known in the data base field, and both books contain much that is of interest to those concerned with current developments in data base technology. These conferences demonstrate again how much material is still in the melting-pot and how much melting is still to be done; despite some persuasive arguments to the contrary, one is left with an intuitive respect for those individuals at Montreal who expressed a fear of premature standardisation in the data base area.

J. Inglis (London)

CAMAC Instrumentation and Interface Standards, IEEE, 1977; 216 pages. (*John Wiley*, £15·00)

Although the computer industry has succeeded in standardising some common parameters such as magnetic tape track formats, it has not established any degree of uniformity in computer input/output buses, even between different models produced by the same manufacturer. Where many different types of instrument need interfacing to an online computing system, such a diversity causes difficult problems for instrument makers, computer manufacturers and the ultimate user.

In an attempt to diminish these problems, several attempts to define a standard bus have been made, probably the most successful being the CAMAC (Computer Automated Measurement and Control) standard dataway proposed by the Committee for European Standards on Nuclear Equipment. The originators in 1969 were concerned with the problem of connecting a variety of measuring and monitoring equipment to a controller which often comprised hardwired logic. It was soon realised that a minicomputer and later a microprocessor would provide a more powerful and more flexible controller, so that many later installations replaced the controller by a computer interface.

The original specification included mechanical and electrical details of the units, and provided for a single crate containing 25 'stations' but this concept was extended in 1972 to allow multiple crate systems. All of these schemes used parallel data transmission, which is fast but involves many lines. Where instruments are well separated geographically, cable costs can be drastically reduced by using serial transmission. The standard for this was defined in 1974.

This book collects together the IEEE standards which defined the CAMAC modular instrumentation and interface system and their associated digital highways, together with a supplement giving the IEEE recommended practice for block transfers in CAMAC systems.

The CAMAC system is now being increasingly applied to online industrial computer installations and over a thousand different instruments are available with CAMAC interfaces. This book gives much detail about the signal levels, timing and protocol and system organisation and would be an invaluable guide to any user involved in an online control system requiring a variety of sensing and monitoring instruments.

J. C. Cluley (Birmingham)

Downloaded from https://academic.oup.com/comjnl/article/20/4/299/[illegible] by guest on 09 April 2024