SORTING TECHNIQUE SIMPLIFIED

MERGING
SORTING

HAL H DAVIES AUGUST 1 1963

ABSTRACT

SORTING TECHNIQUES SIMPLIFIED

by Hal H. Davies

July 15, 1963

Direct Inquiries to: Hal H. Davies
Cleveland IBM Datacenter
2925 Euclid Avenue
Cleveland 15, Ohio
Prospect 1-6050

"Sorting Techniques Simplified" is an analysis of internal sorting and merging techniques used in computer sorting. Simplified examples are used with each method of sorting covered. There is no attempt to evaluate the technique as to efficiency or speed, only in explaining how the technique works. The techniques are divided into two major categories:

1. Internal Sorting
2. Merging

Section I

INTRODUCTION

Much of the literature about sorting deals either with a specific sort or in the evaluation of a particular sorting technique. In this paper, the mathematical evaluation is intentionally missing in order to facilitate the understanding of "how it works". There are, however, some generalized statements about various methods to help place their usage in a program or in a generalized sort.

An effort has also been made to make examples independent of a particular device. In some of the merge examples tapes are assumed as input/output devices. This is not to exclude usage of any other device.

The Glossary of Terms is a collection of terms found in IBM and other sorting materials to help the inexperienced understand the particular technique or evaluation being studied. These are not necessarily used in this paper.

Generalized sorts can be arbitrarily divided into three sections:

A. Assignment Phase

The function of this phase is to establish the parameters for the particular sort. Read and evaluate control cards, determine blocking factors and record length, and calculate "G". The control fields (keys) are determined and program modifications made to comparing areas. The assignment phase turns a generalized sort package into a specific sort.

B. Internal Sort Phase

Internal sort develops string of records. "G" is a term used to describe the memory size in number of records that can be accommodated in the internal sorting phase of the program. It is usually desirable to have the output, or sequence strings, from this phase to be as long as possible to minimize the time necessary in the next phase. In the examples of techniques used in internal sorting all numbers used are keys (control fields). These are limited to two digit numeric numbers to facilitate understanding of the technique. In actual usage most generalized sorts will handle alphabetic or special characters in the keys. All examples assume an IOCS that will bring records one at a time whether blocked or unblocked to the sort program. In some cases a file may be ordered (not in sequence) when the technique starts.

2. Internal Sort Phase (continued)

The number series used is continued through all examples. It is thought to be a random series, but this is not a necessity. It is not used to show the best or worst of the techniques, only to serve as an example. Keeping the same series may help the reader learn and understand the method.

3. Merging Phase

Merges start with 16-sequence strings of records developed in the internal sort phase. In this phase the strings are brought together in each pass developing longer and longer strings until in the final merge pass (sometimes called a separate phase) records are merged into one string, which is the output of the sort program.

Letters have been placed in their relative position to facilitate understanding, i.e. some techniques use a combination of other techniques or area modification of another technique. These must follow in sequence. The sequence is not to indicate merit or order of usage of a method.

In this paper many ideas and examples have come from the papers presented at the Association of Computing Machinery Sort Symposium, November 1952 and recorded in Communications of the Association of Computing Machinery, Volume 6, Number 5, May 1953. A special vote of gratitude goes to D. G. Corbitt of the University of Toronto, Canada for his paper "Sorting on Computers".

It is hoped that this paper can help the reader in a basic understanding of the techniques used in sorting.

Section II

INTERNAL SORTING

In writing of a sort there are many factors that determine techniques to use. Some of these factors are:

1. Number of compares needed to place string in sequence.
2. Available memory size (program steps needed as opposed to size of G). The larger the G the less time is needed to merge.
3. Balance of Input Output speed to processing speed.
4. Features of the computer used.

The decision of what technique to use in a sort comes down to speed (running time considerations). It is desirable to maximize the length of the original strings and minimize the number of compares, but keep the processing time close to equal to the Input Output speed of the computer. A good method in one situation may be a poor method in another. Generalized sorts must also be able to handle a wide variety of situations. Specific sorts can be tailored to a situation. Hence no single method could be singled out as best for all.

A. Exchange Techniques

Exchange starts with a full "G" and proceeds to exchange records until the desired sequence is achieved.

(1) Simple Exchange (Bisection)

Method:

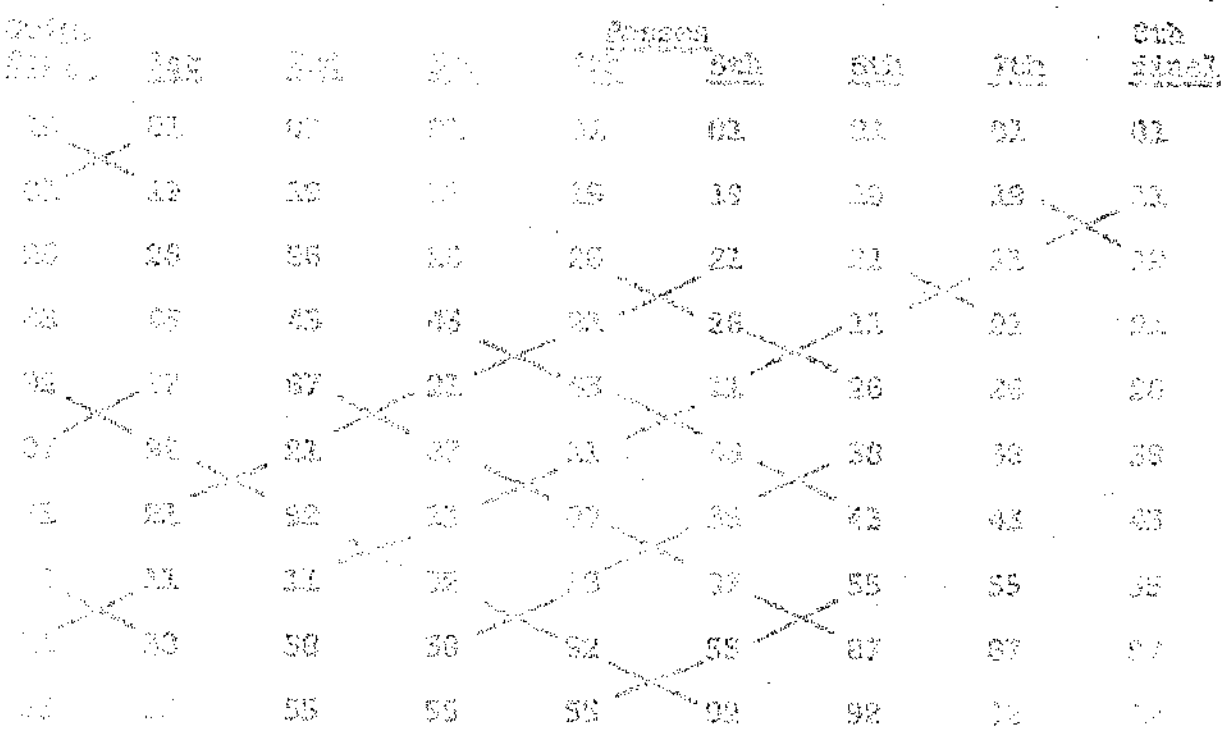
Each key is compared with the following key. If the first is greater, the two are exchanged. If the first is smaller, the compare continues 2nd with 3rd etc. In the case of a switch no attempt is made to use the higher of the two in the next compare. The next compare is with the following two records. If there is a switch between Record 1 and Record 2, Record 3 and Record 4 constitute the next compare. This process is continued until the end of the G area is reached. Each scan of the entire G area constitutes a pass. The passes are continued until no exchanges are made on a pass. The records are then in sequence.

Comments:

This method is used in scientific type programs where a specific group of fields are separated by a fortran "DO" loop. The number of instructions generated is small, but the number executed is large. This method is highly dependent upon distribution of the keys. In the case where the low key is last in the file or a reverse sequence, it requires "n" passes to order file. If the file contains just a few keys reversed, 2 or 3 passes will be sufficient.

Example #2:

Here 13 - 21, 92 - 87, and 31 - 38 are exchanged in the first pass. All other pairs are in sequence. In the 2nd pass only the 93 - 21 are exchanged. A 3rd pass we'll chosen would be necessary to determine that all are in sequence.



Bidirectional Exchange

Method:

The technique here is very similar to the simple exchange. An attempt is made to move high keys to the end and the low keys to the start more quickly. On the first pass the compare starts with the first two records and compares follow as long as records are in sequence. The difference occurs on an exchange. The high of an exchange is compared with the next record. This comparing continues until a higher record at the end of a is found. In the meantime all records passed are shifted up one position. A key can only move up one position but down many. The second pass functions in reverse placing the smaller keys closer to the start of the "a". A key exchanged can move down one but up many. The following passes alternate, odd number pass ascending, even number passes descending.

Test:

Less program steps are required but a small number of compares and exchanges is needed than in simple exchange. Consequently, fewer times on a reverse sequence file 6-1 pass are necessary. Probably the largest usage would be in sorts that are necessary within a program rather than in a generalized sort.

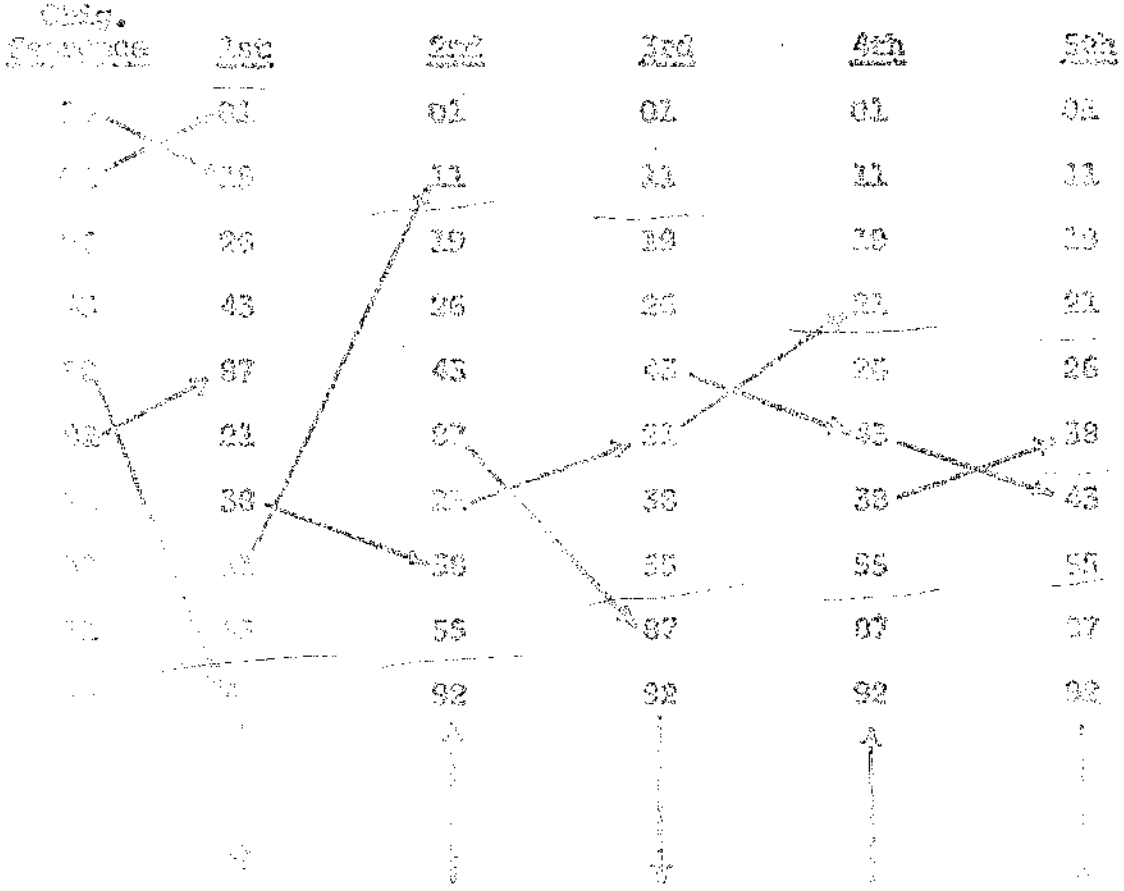
Example:

The first pass ascending descend, recognizes that the 19 and 11 are out of sequence, exchanges them and compares the 19 to the 35. The 19 is then put in position which is executed. Comparing continues until the 98 has to be compared. In this case of sequence the 97 is shifted to 1. The 92 is found for the next sequence and is higher than 11, 97, 11 and 35. The 92 is then placed in the last 6 position with the other records, shifted up one position each.

In the second pass the compares start with the 92 and function in the opposite direction. The first out of sequences are the 11 and 30. The 23 drops down and the 11 moves up to the second position. A sixth pass not shown is needed to check for completion of sequencing.

Appendix #2

PERMUTATION EXCHANGES



Arrows indicate direction of pass.

Numbers by numbers show movement of keys.

[illegible] Binary Radix Exchange

Method:

[illegible] is started on the beginning of "U", binary bit position of key. The first one bit encountered is exchanged with the first zero bit encountered [illegible] the scan from end of "G". The scan is continued and exchanges are made until the beginning scan meets the end scan. An indicator is then placed at this position to establish the [illegible] of the file for following passes. Above the indicator all positions are "0", below all are "1". The next high order [illegible] [illegible] indicators [illegible] to the previous pass [illegible] the number of items in the pass.

The passes continue until the [illegible] bit positions have been scanned. At this point all records are in sequence. Each record [illegible] [illegible] will have an indicator set.

Uses:

Binary radix exchange is particularly adapted to the binary computers. The length of the key can be the limiting factor in that the passes are equal to the number of bits in the control field.

Example [illegible]:

In the first pass all numbers less than 64 fill the first group. 64 and 99 fill the second. The scan continues to decrease by the power of 2. The indicators are placed at 64 on first pass, 32 and 96 second, 16, 48 etc., [illegible] until the last pass where there is an indicator placed on all [illegible] except equals. (The brackets show an exchange (always [illegible] with [illegible]). The V indicates bit position scanned. The lines [illegible] [illegible] areas.

Example 63

(4) Merge Exchange

Method

This technique endeavors to quickly bring low keys to the front of "G" and high keys to the end. For best results "G" should be a power of "2". The first pass compares the first record with first record in the second half of "G". If in sequence they are unchanged; if out of sequence, they are exchanged. This is continued with the second record in the first half with the second record in second half. This proceeds until the last record in the first half is compared with the last in the second half.

The next pass uses a G quartered i.e., first record in first quarter, to first in second quarter, to first in third quarter, to first in fourth quarter. These records are placed in sequence. It may take more than one exchange to complete the sequencing. The process is continued until a section contains one record.

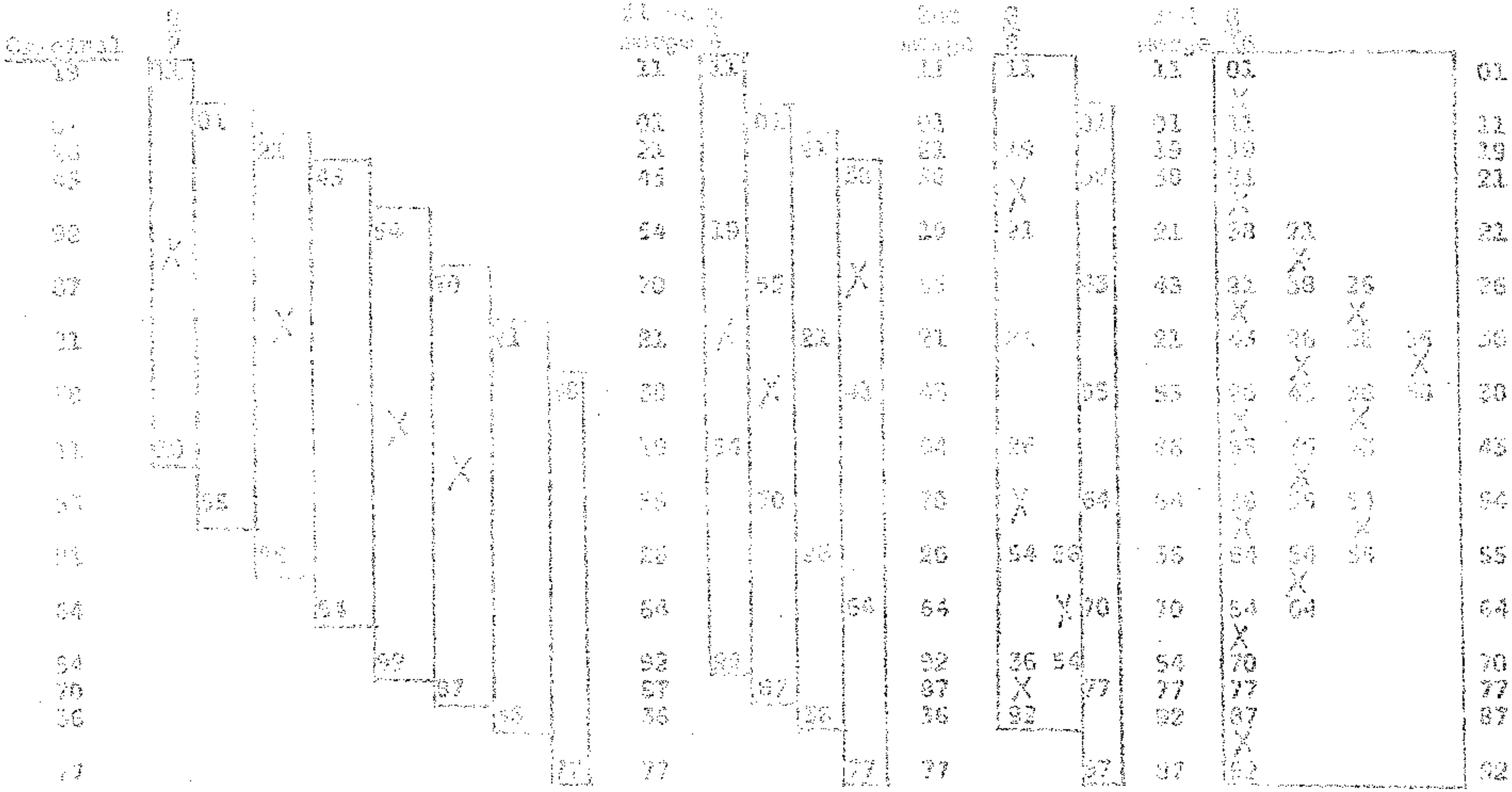
Pass	Section
1	G/2
2	G/4
3	G/8
4	G/16
.	.
.	.
n	G/G

Use:

This technique tends to decrease the number of compares to sort a "G". However, the number of exchanges runs higher than most other methods. Heavy use of indexing registers may be required to make this approach practical. Sizeable improvement is shown in the sift method. There is a tendency to have multiple exchanges on the final pass.

Example #4:

In the first pass with a G equal to 16, the 73 is compared with the 12 and exchanged. The 81 is compared with the 55. No exchange occurs. etc. Exchanges are marked with an "X". In the third pass a double exchange occurs. The 34 is compared to the 25 and exchanged. 34 is then compared with the 42 no exchange. The 92 is compared with the 81 and exchanged. At this point sequence is still not correct. The 92 and [illegible] must still be exchanged. Many multiple exchanges occur in the final pass.



5. [illegible]

[illegible]

[illegible] is basically the merge exchange technique with the factors [illegible]. The formula used for the factor is [illegible] + 1. The [illegible] portion only is used. [illegible] is divided by 2 to find the [illegible] until factor is equal to 1. i.e.:

G = [illegible] factors are [illegible]
G = 16 factors are 9, [illegible], 1
G = [illegible] factors are [illegible], 1

[illegible]

[illegible] above factors, less compares and less exchanges are needed [illegible] the same. Frank and Lazarus (ACM January 1960) feel [illegible] technique is effective with a "G" up to 256.

[illegible]

[illegible] pass 1 = 9, pass 2 = [illegible], pass 3 = 2, pass 4 = 1. The 19 [illegible] compared with 56. The [illegible] change occurs at [illegible] compared with [illegible]. In this example the effect of adding an additional pass with a factor of [illegible] shown. The effect in this case is to reduce the number of exchanges and the [illegible] of G. Using the factor "2" has not at this time been proven. This case and [illegible] are [illegible] to indicate some value may be there.

Example #5

Or by adding a factor 2 between 3 and 1

B. Radix Variations

This method examines each digit of a key to determine sort. Binary, radix exchange is included since exchange happens the records are unchanged. The card sorter is the best example of radix sorting.

(1) Radix or Distribution Sort (The Card Sorter Method)

Method:

The key is examined starting with the units position in the first pass and the record placed in the designated pocket or reserved memory area determined by the value of this key position. Records are then collected from the pockets keeping records in same order as they entered the pocket.

The next pass considers the next digit position to the left, again placing records in designated pockets. Another collection is made. After two complete passes, the high order digit has been examined and the collection made. Records are now in key sequence.

Usage:

The computer difficulties in this method center around two elements. 1) The variable pocket size and 2) the length of the key. However, this might be used in modified form to "break down" using a high order digit to facilitate use of another technique. The records in each pocket is then sorted independently of the rest.

Example #1:

The key (19) units digit (9) is examined and placed in the "9" pocket. Next the (02) is examined and placed in the "2" pocket. This is done for all records. Finally the (77) is examined and placed in the "7" pocket. The records are then collected, being the first in first out method. In the first collection the (01) is taken from pocket "1" first, then (21), (11) and finally the record (31).

Method: Sorting is pre-counting on a high order digit to decrease what will be sorted.

Example #6
RADIX OR [illegible]
(POCKET SORT)

Morning	Pocket #	
[illegible]	70	0
[illegible]		
[illegible]	21, 11	1
[illegible]	21, 01	
92	92	2
[illegible]		
[illegible]	43	3
[illegible]		
[illegible]	[illegible]	4
[illegible]		
[illegible]	55	5
[illegible]		
[illegible]	36, 26	6
[illegible]		
[illegible]	[illegible]	7
	[illegible]	8
	[illegible]	9

[illegible] Units
Position of Key

Evening	Pocket #		
70	01	[illegible]	01
01			11
21	19, 11	1	19
11			21
21	26, 21, 21	2	21
92			26
43	36, 36	3	[illegible]
63			[illegible]
54	[illegible]	4	[illegible]
55			54
26	55, 54	5	55
36			64
07	64	6	70
77			77
[illegible]	[illegible]	7	87
19			92
1st collection	[illegible]	8	2nd col.
	92	9	

[illegible] Tens
Position of Key

C. Merge Techniques

Merges use a full "G" to sort and natural or forced strings are merged until a final string is formed.

(1) Two-Way Merge

Method:

Keys are first compared in pairs placing each element in the pair in sequence. In the second pass two strings of two are merged to form a string of four. This is accomplished by comparing the first of each string and placing the lower in the larger string. Next the key left after the first compare is compared to the next key in the other string. Again the low is placed out. This is continued until the two strings form one string equal to the sum of the length of the original strings. The first pass of the 2-way merge is a simple exchange first pass.

Uses:

The standard 2-way merge is probably the most used internal sort in generalized sorts. Merging is not affected by adverse sequences within the "G". The programming is straight forward and number of steps are minimized. For a sort that is designed to handle a wide variety of situations the 2-way merge is a good technique.

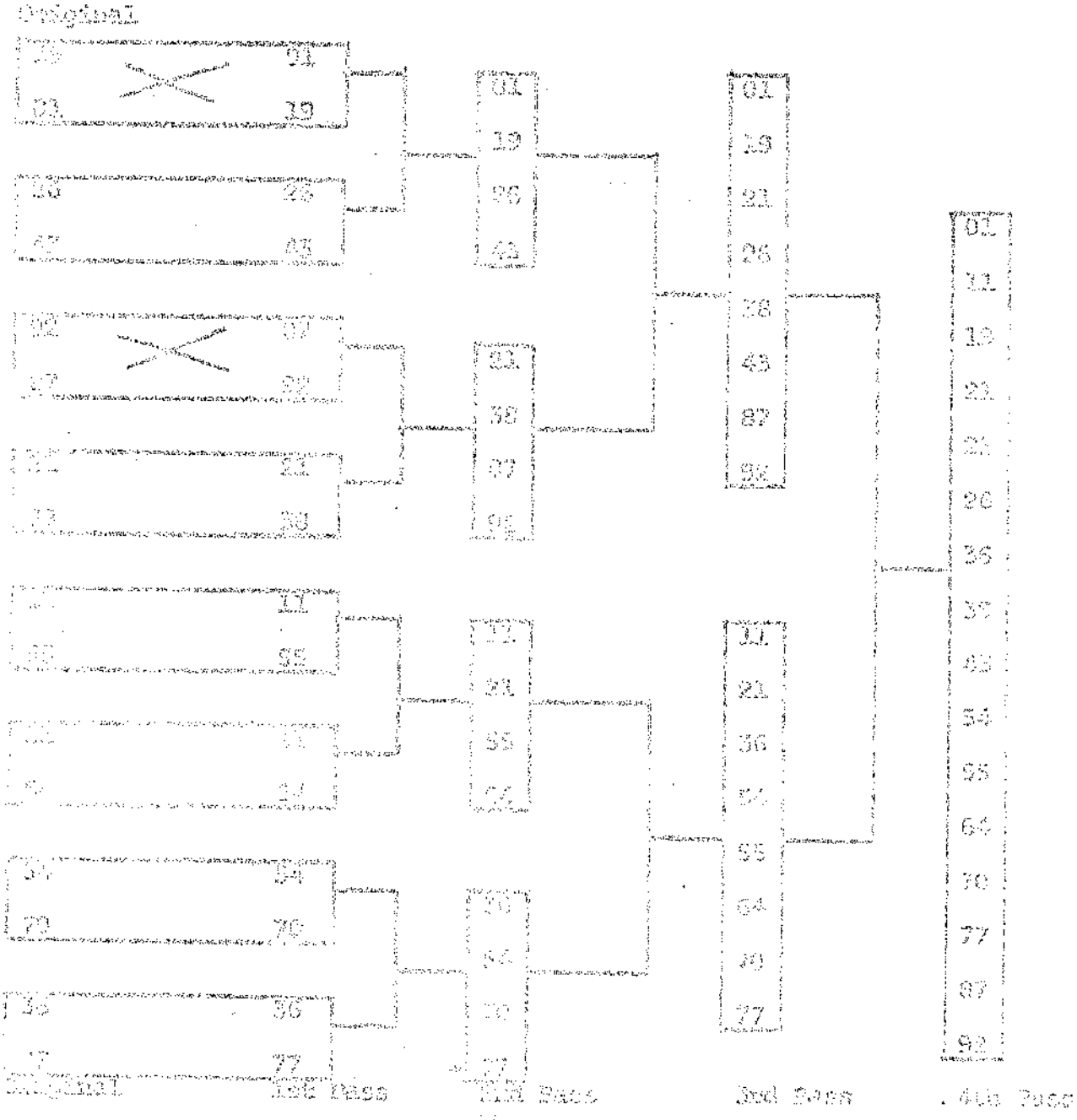
Example (1):

2-way merge builds strings of 2 using a single exchange then these strings are merged. The third set of strings of 2 is used.

Compares	Result String
1. 11 (1st) to 21 (2nd)	11
2. 55 (1st) to 21 (2nd)	21
3. 55 (1st) to 64 (2nd)	55
4. - Copy 64 (2nd)	64

All comparing proceeds in a similar manner.

Example 27



11) Variable String Merge

Method:

Instead of the exchange used in the first pass of the 2-way merge, the variable string merge scans for natural sequences in the ascending or descending sequence. This ability is somewhat dependent upon the natural subdivision of the computer word. When the strings are first merged, all succeeding merge passes operate in almost all cases though the strings to be merged are not necessarily equal in length.

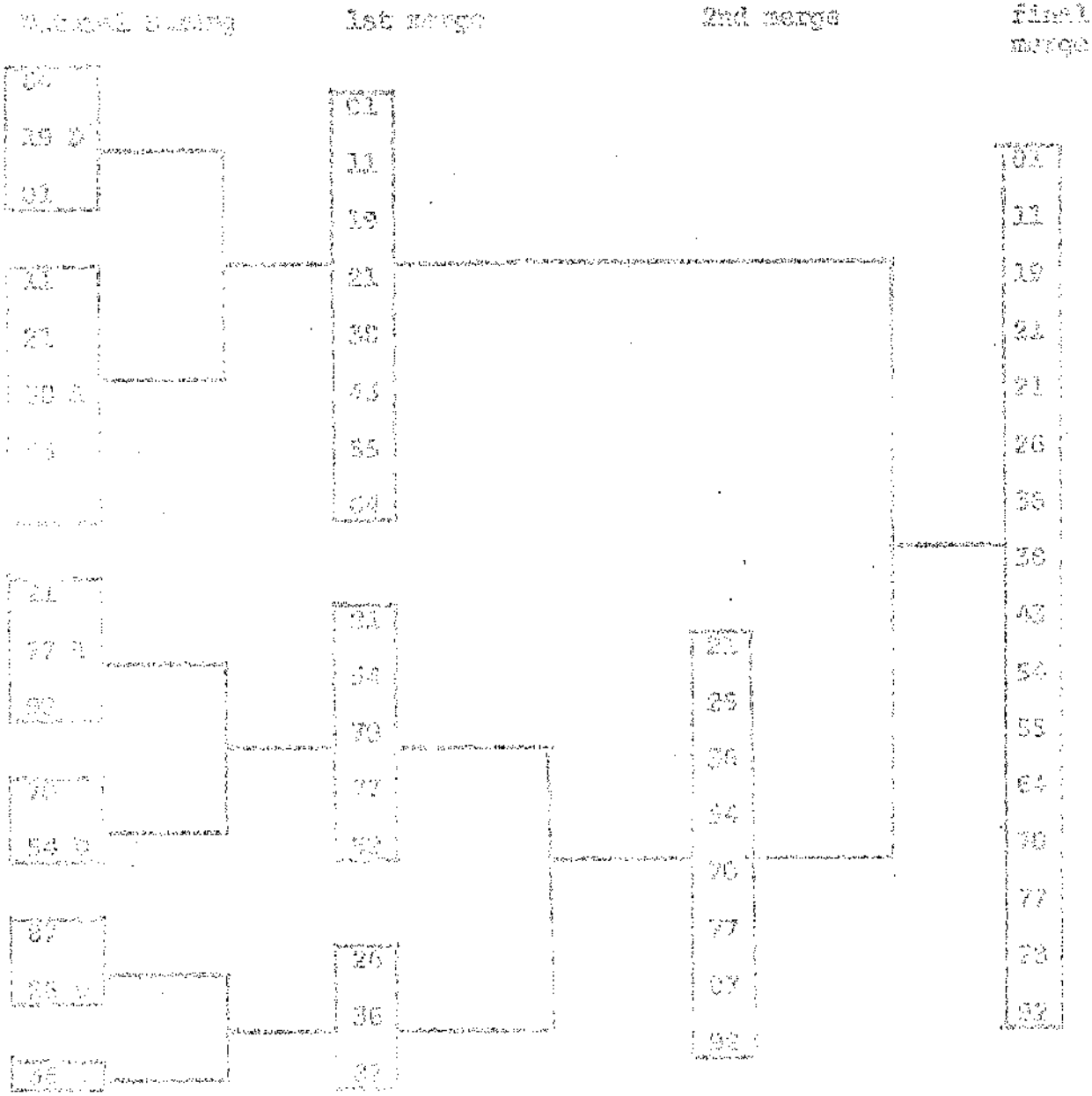
Use:

The use of this technique will take advantage of any natural sequence within a "set" group of records from input. If the "N" of records is in either ascending or descending sequence, it will be written immediately. Existing advantage of any natural string can give a significant time improvement. In the worst situation it is the same as the 2-way merge.

Example #7:

The string used in the other examples has been changed to better show the variable string merge. The first sequence (54, 19, 01) is found to be descending. The next (33, 21, 39, 43, 55) ascending. The last piece in that it is a string of 1. The sequence involved to write the three two strings are:

Sequence	Result String
54 (Hex 36) to 01 (Hex 01)	01
19 (Hex 13) to 21 (Hex 15)	33
33 (Hex 21) to 39 (Hex 27)	39
43 (Hex 2B) to 43 (Hex 2B)	43
55 (Hex 37) to 55 (Hex 37)	55
54 (Hex 36) Copy	54


1st merge
2nd merge
final merge

B. Insertion Technique

These methods start with C empty and fill it by bring in record one at a time.

(1) Insertion

Remarks

The entire "C" are is initialized with a high pad (99) to save the operation of keeping a count of the number of records in "C" at any given time. A count must be used to eliminate an overflow of "C".

Each record is compared to the records in the "C" area in sequence. When a low compare is found the remaining "C" is shifted one position to right and the new record inserted at this point.

Use:

Insertion is a simple straight forward approach to a limited sort. Probably the easiest of all techniques to program. The speed can be bettered in almost all other techniques. If dealing with a reverse sequence the number of compares is equal to "C".

Example #1:

The (19) compares low to (99) shift right and insert at first position.
The (01) compares low to (19) shift right and insert at first position.
The (25) compares high to (01) and (19) and low to (99) etc.

Example #9
Insertion

Input	G Area	
#1 19	(99)	(99)
#2 01	01 19 (99)	(99)
#3 26	01 19 (26) (99)	(99)
#4 43	01 19 26 (43) (99)	(99)
#5 92	01 19 26 43 (92) (99)	(99)
#6 87	01 19 26 43 (87) 92 (99)	(99)
#7 21	01 19 (21) 26 43 87 92 (99)	(99)
#8 38	01 19 21 26 (38) 43 87 92 (99)	(99)
#9 11	01 (11) 19 21 26 38 43 87 92 (99)	(99)
#10 55	01 11 19 21 26 38 43 (55) 87 92 (99)	(99)
#11 21	01 11 19 (21) 21 26 38 43 55 87 92 (99)	(99)
#12 64	01 11 19 21 21 26 38 43 55 (64) 87 92 (99)	(99)
#13 54	01 11 19 21 21 26 38 43 (54) 55 64 87 92 (99)	(99)
#14 70	01 11 19 21 21 26 38 43 54 55 64 (70) 87 92 (99)	(99)
#15 36	01 11 19 21 21 26 (36) 38 43 54 55 64 70 87 92 (99)	(99)
#16 77	01 11 19 21 21 26 36 38 43 54 55 64 70 (77) 87 92 (99)	

(2) Binary Search

Method:

The binary search technique is used as the search method in the framework of the Insertion Method (II, D, 1). The Regenerative technique (II, F, 1) is also involved in the example.

To explain this technique, a full "G" in sequence is assumed and we are adding records to G and taking the low records from "G".

In contrast to the compare method shown in insertion, the binary search with each compare eliminates half of the remaining records in "G". The $\frac{G}{2}$ record is compared first. The $\frac{G}{4}$ or $\frac{3G}{4}$ record next depending on the result of the first compare. This process is continued until the proper location "G" is found. The low record is then placed in output and the "G" shifted to the left. The new record is inserted into its proper location.

If the new record is found to be less than the first record in G but higher than the previous output record, it can be placed in output immediately, leaving G intact. The effect of a sequence break record will be explained in the Regenerative technique (II, F, 1).

Use:

The binary search is a compare technique to be used when finding the sequential location of a record in a group of records. As a sort technique, it uses less compares to find a location but will usually involve more program steps to write. If it is used to initially fill "G", the difficulty of a varying "G" size is encountered.

Example #10:

"G" contains 9 records in sequence. 55 is next record. 55 is compared to 26 $\left(\frac{G}{2}\right)$ high. 55 is compared to 45 $\left(\frac{3G}{4}\right)$ high. 55 compared to 87 $\left(\frac{7G}{8}\right)$ low. 01 is placed in output and records in position 2, 3, 4, 5, 6, and 7 are shifted left one position. The 55 is inserted in position 7. This process is continued until a sequence break forces either a decrease of G size as a Regenerative or the records remaining in G written out and new string started with the sequence break record.

Example #10
Binary Search

C = Compare L = Low
H = High E = Equal

ID	"G" Positions 1	2	3	4	5	6	7	8	9		OP
	01	11	19	21	26	38	43	87	92	Full "G"	
55					(1) C/H		(2) C/H	(3) C/L		Shift left from 71	01
	11	19	21	26	38	43	55	87	92		
21			(2) C/E		(1) C/L					Shift left from 71	11
	19	21	21	26	38	43	55	87	92		
64					(1) C/H		(2) C/H	(3) C/L		Shift left from 71	19
	21	21	26	38	43	55	64	87	92		
54					(1) C/H	(1) C/L	(2) C/L			Shift left from 51	21
	21	26	38	43	54	55	64	87	92		
70					(1) C/H		(2) C/H	(3) C/L		Shift left from 71	21
	26	38	43	54	55	64	70	87	92		
26	(4) C/E	(3) C/L	(2) C/L		(1) C/L					Replace 1	26
	38	38	43	54	55	64	70	87	92		
77					(1) C/H		(2) C/H	(3) C/L		Shift left from 71	38
	43	43	54	55	64	70	77	87	92		

(3) Address Calculation

Method:

In the assignment phase a calculation is made to develop a factor to be used in the address calculation. This factor (F) is developed by dividing the range of keys by "G".

$$\frac{\text{Range}}{G} = F$$

In saving a "G", each key is divided by "F" to develop an effective address. This address is a relative position in "G". (This step need not be executed for all records before the next starts, but can be executed as records are received from input). Starting at first of "G" keys are placed in their address position until the situation arises that a key has an address that had previously been filled. The new record is then compared to the resident record and the lower placed in the address position. Sequentially, all adjacent filled addressed positions are compared to the high of the two. Comparing continues until a higher is found. The remaining adjacent are shifted down one and the key is then inserted.

This process continues until the last key is placed in the "G" area.

Use:

The address calculation method's value is to a large degree dependent on the distribution of the file to be sorted. The element that in most cases is considered good distribution (close to ascending or descending) can be very unfavorable. Take, for example, a large file major and intermediate keys in sequence with many duplicate records only the minor field would be in order. In this case every record in a particular "G" group would develop the same address, completely negating the effect of the calculation.

A good file for this method is one that is randomly distributed. i.e.: in any one "G" most keys develop unique addresses.

A weak part of this approach is the extended use of the divide command. There is a great difference in the speed of divides between machines. Also, do the keys in the file, contain alpha? If they do, division is difficult if not impossible.

This method, for random records uses very few compares as compared with other techniques.

(3) Address Calculation (continued)

Example #14:

The range (00 to 99) is divided by G size (16) giving the resultant factor of (6).

The key is divided by 6 to obtain an address. In division, no remainder is used. The 19, 01, 26, 43, 92 are placed in address without conflict. The 21 with a repeat of the address 3 causes the first compare. The 19 already in address 3 is low and remains the 26 (higher than 21) in address 4 is shifted to address 5 and the 21 placed in address 4. The stops indicate positions where shifts must occur. All keys in "G" and the shifts are shown at this point.

Example #11

Original	Address		1st	2nd	3rd	4th	5th	6th
19	3	(1)	01	01	01	01	01	01
01	1	(2)		11	11	11	11	11
26	4	(3)	19	19	19	19	19	19
43	7	(4)	26	21	21	21	21	21
92	15	(5)		26	21	21	21	21
87	14	(6)		38	26	26	26	26
1st Stop 21	3	(7)	43	43	38	38	36	36
38	6	(8)			43	43	38	38
11	2	(9)		55	55	54	43	43
55	9	(10)				55	54	54
2nd Stop 21	3	(11)			64	64	55	55
64	11	(12)				70	64	64
3rd Stop 54	9	(13)					70	70
70	12	(14)	87	87	87	87	87	77
4th Stop 36	6	(15)	92	92	92	92	92	87
5th Stop 77	13							92
6th Stop Final								

E. Selection Technique

Selections start with a full C and pick one record at a time in desired sequence.

(1) Cascade Selection

Method:

"C" is first divided into sections of 4 keys. C sizes of 4, 8, 16, 32, 64 etc. work best. The lowest of each group of 4 is picked. The lowest of 4 groups is then picked etc. until the lowest in the "C" is found. Picking the lowest is called initializing "C" i.e.: that is to pick the low of all groups of 4 through all levels. After a low is found the program goes back to group from which the low was chosen and picks the new low of the group. Only the compares effected by the new pick are executed to find the next low.

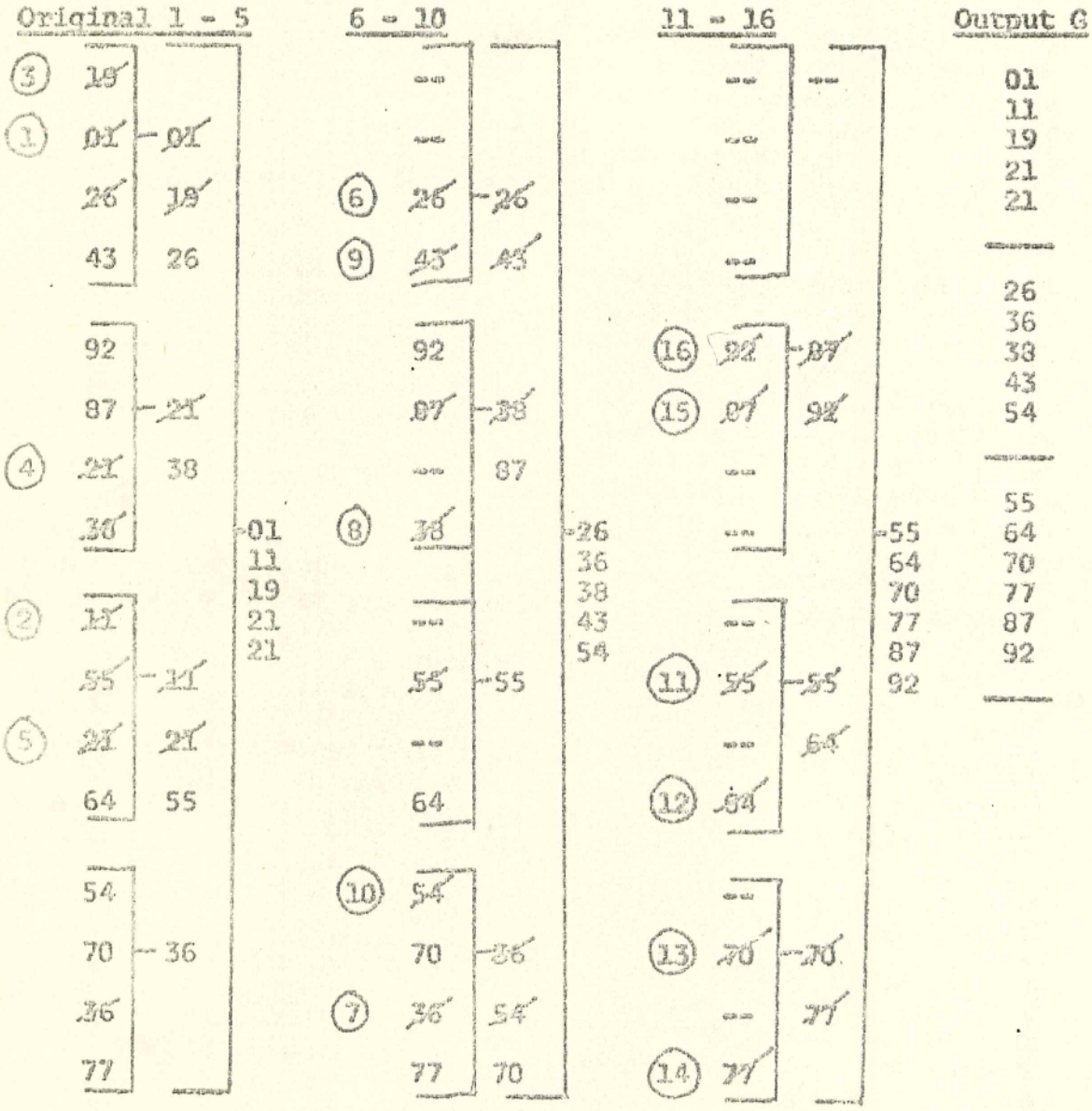
Uses:

The major advantage of this method is that once "C" is initialized (the first low found) all following selections are made with a minimum of compares. Bad sequence has no effect upon the time or number of steps.

Example #12:

The lows selected in initialization are 01, 21, 11 and 36, from these the 01 is selected as the low. The 01 is replaced by the next low in the first section (19) and the second selection is repeated (19, 21, 11, 36), from which the (11) is selected. The (11) is then replaced by the 21 from the third set (19, 21, 21, 36). From this the 19 is picked as the third pick. This process is continued until all records from all quads have been picked. In example #12 the selections are arbitrarily divided into packs, 1-9 are shown in the first column, 6-19 in the second column and 12-16 in the third. The numbers in circles indicate the keys picked. Keys picked in the first column are eliminated in the second etc

Example #12
Quadratic Selection



(2) Replacement Selection (Tournament Sort)

Method:

"n" is set to a number that is a power of 2 i.e. 2, 4, 8, 16, 32, 64 etc. C is filled with records. The first winner is found in the initialization of C. Pairs of records are compared and the address of the winner is placed in the first level table. Pairs of level 1 winners are compared and the winners addresses are placed in the level 2 winners table. This process in continued to standard tournament style until the winner of the final compare is placed in output.

The winner's address is used for the next record from input. The portion of the tournament affected by this change is played (one compare at each level) to find the second winner.

The input record is always checked with the last output record. If the input record would cause a sequence break an indicator is set causing a "bye" in this set. When the indicators are set on all records the string is completed and processing of the next output string starts.

After the first winner is found the number of compares to find the next is equal to the number of levels in the tournament.

n = number of compares
$2^n = C$

Records are only moved from input to tournament and from tournament to output. While tournament is progressing they hold their same position in memory. The record address is used and compares made through indexing.

Use:

This technique requires a small amount more memory for winners tables and program steps, but the low number of compares necessary to find each winner places it in strong contention as one of the fastest techniques. Sequence does not adversely affect the number of compares but it does affect the string length generated. When these adverse sequences do occur other regenerative techniques are affected in the same manner.

Using random records string lengths averaging 2C are formed*. When complete reverse sequence occurs string length is equal to C or C + 1, depending on whether the new record is brought in before the first winner is placed in output or not. If all records are within C records of their sequence in output, a single string will result.

*Frank, Kenneth A. "Internal and Tape Sorting Using the Replacement-Selection Technique." ACM Sort Symposium November 1957

(3) Replacement Selection (Tournament Sort) continued

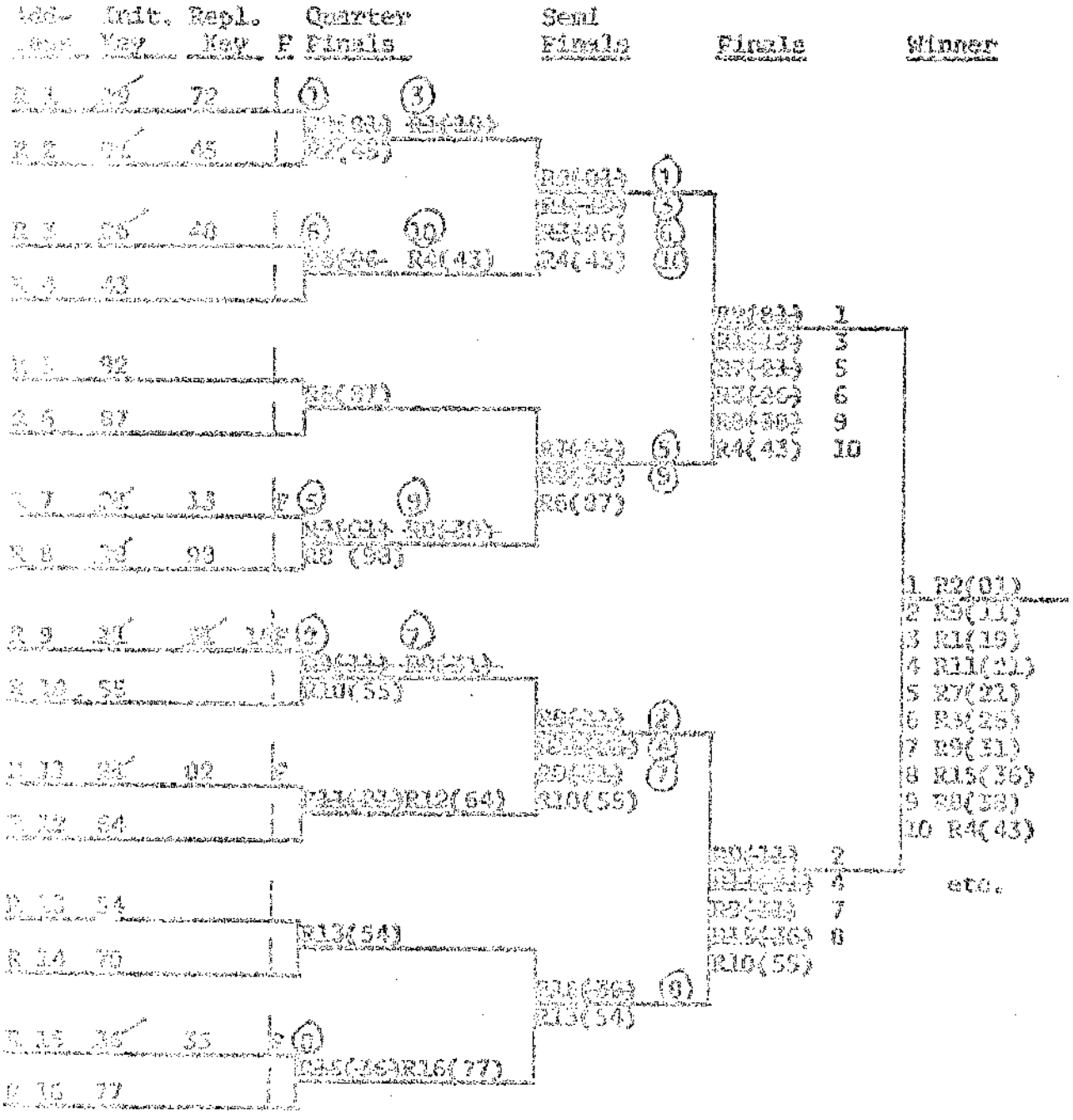
Examples 13 A, 13 B, 13 C, 13D:

Due to the difficulty of picturing the dynamic aspects of this method several examples are used. Example #13A shows the tournament through the outputting of 10 records. No new input record has been entered to replace 24. The tournament (string) ends when all record positions are forfeited (F). To show the steps in the development of example #13A, example #13B, example #13C, example #13D are used. Example #13B shows the status of the tournament on initialization. Quarter final, semi-final, and finals cables are filled. 2(01) is selected as the winner (first output record). Example #13C shows the number of compares to determine the second winner 8(11). The key (45) has replaced (01) in R2. Example #13D shows the second forfeit (F) record 6 + 5 (the fifth record in after 6 was initially filled) a (13). The last output record at this point was 8(91), or higher than the input (13), therefore a forfeiture (F) is caused and the tournament continues.

The end of this string is not shown, but would occur when the last of the record addresses becomes a forfeiture. Tournament is again initialized to start the next string.

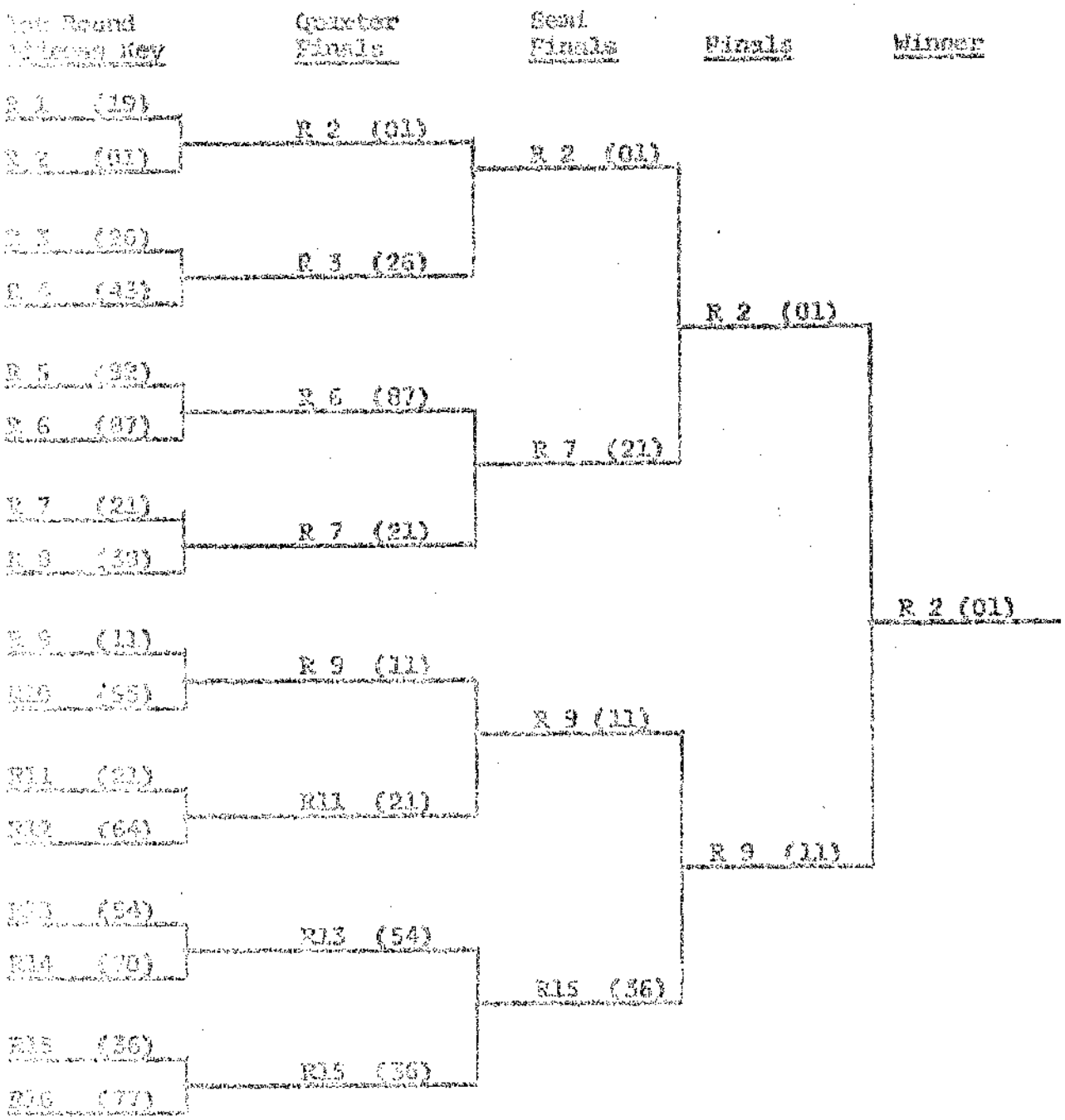
Example #13A

REPLACEMENT - SELECTION, Tournament Sort

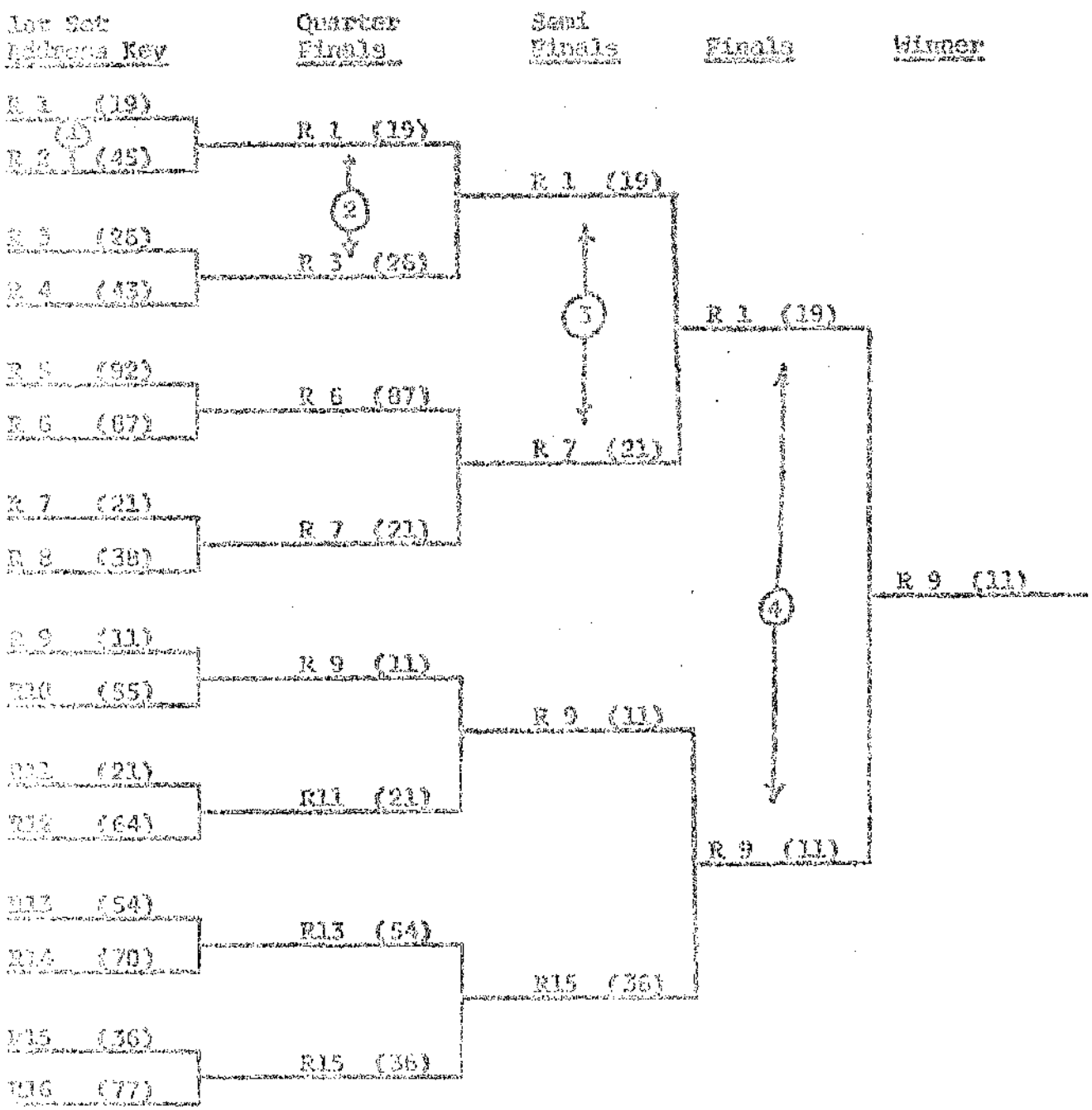


Input after initialization 45, 31, 72, 02, 13, 48, 16, 35, 98 etc.

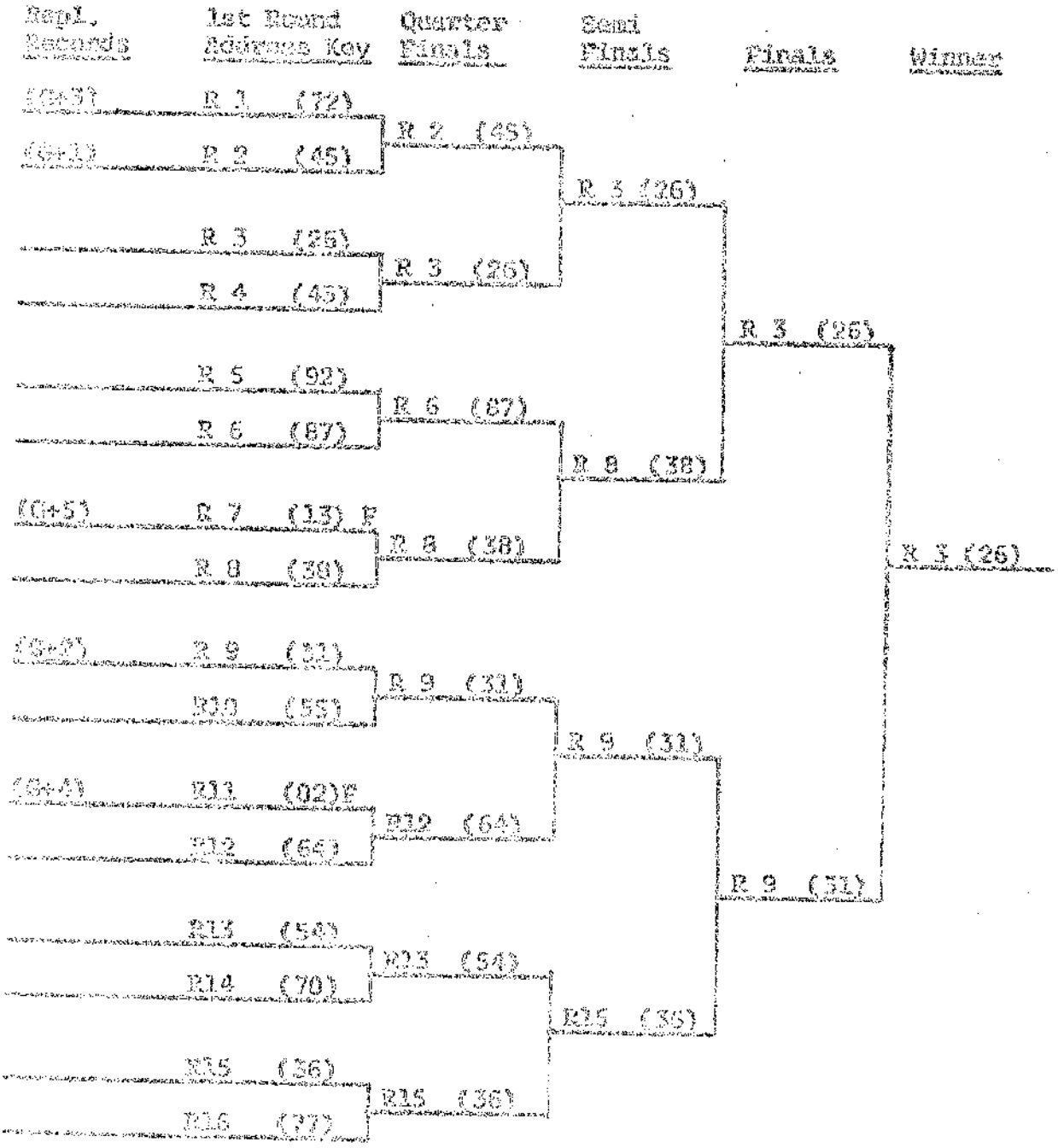
Example #153
REPLACEMENT - SELECTION, Tournament Sort
Initialization of Tournament



Example #130
REPLACEMENT - SELECTION, Tournament Sort
Play to Determine 2nd winner - New record is 45 in Record 2



Example #130
REPLACEMENT - SELECTION, Tournament Seed
Status of Tournament at Input of (6+5) Record



F. Regenerative Technique

This is not a basic technique as the others but can be used in conjunction with the insertion and selection techniques to develop strings longer than G. It does not apply to the exchange, Radix or merge techniques.

(1) Regenerative

The regenerative technique is not a sorting method in itself but a method of expanding several other methods to develop a string of records longer than the "G" size.

Method:

G records are sorted into sequence. The next record is placed into the sequence making room for it by placing the low record in output. When an input record causes a sequence break it is placed in the first position of G and the size of G is decreased by 1. The next sequence break record is properly positioned and again G is decreased by 1. The string is terminated when G is equal to 0.

Use:

This technique is used to develop strings longer than G. The insertion, binary search, replacement selection or quadratic selection techniques can utilize this method. In a reverse sequence the length of the string is G + 1 but dealing with random files the string length averages about 2G.

Example #14:

The small G = 4 was selected to show a complete cycle. The first four records were placed in G by the insertion method. G is 01, 19, 26, 43 when the regenerative procedure starts. The 92 is placed in G while the 01 is placed in output. When the first 21 is the input record it is found that it is less than the first record but higher than the last output so it is placed in output immediately with G unchanged. The 11 key causes the first sequence break. G is now equal to 3. At the second 21 key the G size becomes 2. The addition of the 70 key causes G to be depleted and the string ends. G reverts back to 4 and the process continues.

Example #24
Interactive

Input		S = 4					Output String
19	(Insertion Method)	19					
01		01	19				
26		01	19	26			
43		01	19	26	43		
92		19	26	43	92		01
87		26	43	87	92		19
21		26	43	87	92		21
38		38	43	87	92		26
11		11	43	87	92	(2.5 C)	38
55		11	55	87	92		43
21		11	21	87	92		55
64		11	21	87	92		64
07		11	21	56	92		87
70		11	21	56	70		92
11		21	38	56	70		11
77		36	56	70	77		21
		(At least 4 more to fill this sequence)					36
							56
							70
							77

Section III

Merging

Merging is used in the third phase of a generalized sort. Strings developed in the preceding phase are brought together forming longer strings with each pass. Input/output devices are implied as opposed to section II where all manipulations were within memory. In all examples numbers refer to strings and are not keys.

A. Balanced Merge

As the name implies, balanced merge uses an equal number of input/output devices on either side of the merge network. A two channel computer, complicates this. On a pass, reading is performed on one channel, writing on the other. The desire on this arrangement is to obtain complete overlap of reading and writing. Balanced merging can be accomplished in any order (n) depending upon the number of available I/O devices.

To determine the proper order of merge, several considerations are important:

(1) block size, (2) memory size, (3) relative input/output speed to calculation speed, (4) size of control field, (5) size of record.

The combination of these factors will determine the most efficient order of merge.

(1) Balanced four-way Merge

Only one example of balanced merging is shown due to complete similarity between merges of different orders. It is reasonably safe to assume that the higher the order of merge the fewer passes it will require to complete the merge phase. This statement can be misleading when dealing with specific generalized sorts. Using a 3-way merge as opposed to a 2-way merge in the preceding phase will have two output files requirements added. This could make a significant difference in the starting lengths developed. In many situations it is possible for a 2-way merge to beat a 3-way merge. Over the entire spectrum of sorting volumes and record lengths, the 3-way is superior, but in certain cases the 2-way is significantly faster.

(1) Balanced 3-way Merge (continued)

Method:

The preceding phase places a group of strings on each of the devices. During a merge pass one string each from all input devices comes together to form one string on the output device. (See example #16A). The next set of strings (second string on each device) is merged into a single string on the next output device. After the output string is placed on the last output device, the next is placed on the first. A pass is completed when all input strings have been placed in output. The number of strings placed on the output is calculated:

$$\frac{\text{Input strings}}{\text{order of merge (n)}} = \text{output strings}$$

The second pass starts reading the strings prepared by the preceding pass. Passes continue until s = 1.

The final merge pass is often designated as a separate phase to facilitate late sort modifications. The last pass occurs when there is one string or less strings on each of the input devices (Phase 3 of example 15B).

Example 15 A:

In example 15 A the transition is made from keys to strings. An unrealistic string length of 4 is used to have strings small enough to show the effect on a single page.

Strings 1, 2, and 3 are brought together comparing the 01, 21 and 11. The 01 is low and placed in output, the 01 is replaced by the 19 and the low of 11 outputed next. Example 15 A shows computer forming string 4 on output from S-1, S-2, and S-3 on input.

Example 15 B:

Example 15 B shows how strings 1, 2, and 3 on input form string A on output. Strings 4, 5, and 6 form string B, etc. When the final output (tape six) is used, the next string "D" is placed on the first output (tape four).

(1) Balanced 3-way Merge (continued)

Example 15 C:

Example 15 C shows the merging of 30 strings from the preceding phase down to one string in four passes. Notice that at the end of the first pass there are $\frac{S\,(30)}{n\,(3)} = 10$ strings left.

Pass two shows how the number of remaining strings does not remain equal when "n" does not divide evenly. After the merge of the ten strings, four remain. The 5 string was in effect copied. The tendril/dummy shows more dramatically in the final pass where the strings I, II, and VII are merged with string IV. Strings I, II, and III represent 27 original input strings where the string I or IV represents only 3. The last three strings in effect cause two extra passes.

If the order had been 6-way, the results would be

Pass 1	-	30 to 8
Pass 2	-	8 to 2
Final	-	2 to 1

If a 5-way merge were used

Pass 1	-	30 to 6
Pass 2	-	6 to 2
Final		2 to 1

If a 6-way merge were used

Pass 1	-	30 to 5 .
Final	-	5 to 1

Probably the most effective, assuming the same string lengths, would be the 6-way in two passes. This discussion will be continued in the variable order of merge, section III C (2).

Example 15 A

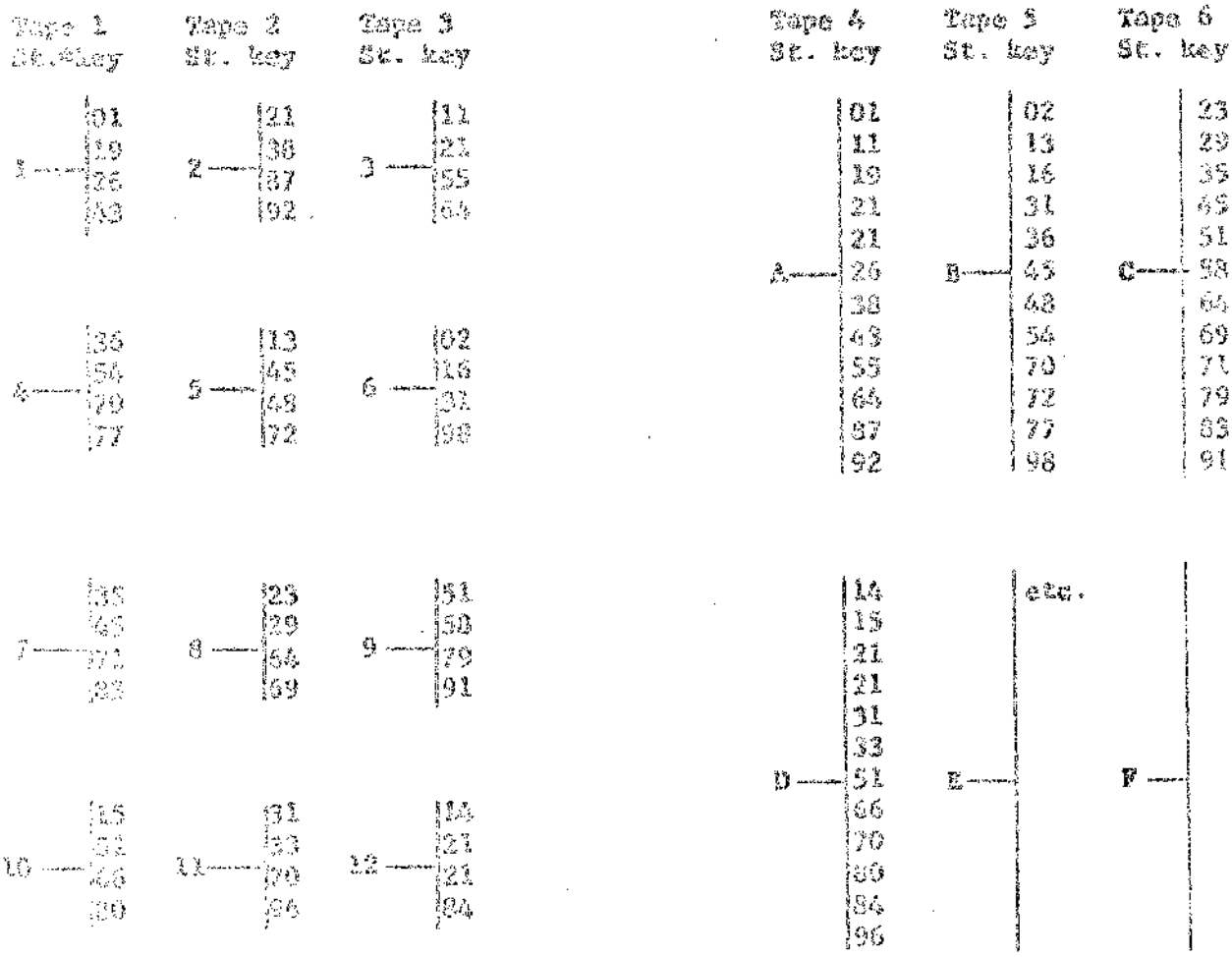
CONTENTS OF SELECTED MEMORY

Input		Compares			Output
key from string		S1	S2	S3	key from string
01	1	01	--	--	
21	S2	01	21	--	
11	S3	01	21	11	
Comparing Starts:		-01-	21	11	01 - S1
19 -	S1	19	21	-11-	11 - S3
21 -	S3	-19-	21	21	19 - S1
26 -	S1	26	-21-	21	21 - S2
33 -	S2	26	33	-21-	21 - S3
26 -	S3	-26-	33	26	26 - S1
38 -	S1	43	-33-	26	33 - S2
43 -	S2	-43-	43	33	43 - S1
55 -	S1	--	43	-33-	55 - S3
64 -	S3	--	43	-64-	64 - S3
87 -	S2	--	-43-	--	87 - S2
93 -	S2	--	-93-	--	93 - S1
END -	S2				END OF STRING

Example 15 B

MERGING KEYS (Keys to Strings)

Input | **Output**



etc.

* St. = String

Example 15 c

BALANCED 3-WAY MERGE (Strings)

Output Pass 1 (strings)

T1	T2	T3	T4	T5	T6
1	2	3	0	0	0
4	5	6			
7	8	9			
10	11	12			
13	14	15			
16	17	18			
19	20	21			
22	23	24			
25	26	27			
28	29	30			

Phase 2 Pass 1

T4	T5	T6
(1, 2, 3)A	(4, 5, 6)B	(7, 8, 9)C
(10, 11, 12)D	(13, 14, 15)E	(16, 17, 18)F
(19, 20, 21)G	(22, 23, 24)H	(25, 26, 27)I
(28, 29, 30)J		

T1 T2 T3

Phase 2 Pass 2

(A, B,C)X (D,E,F)XX (G,H,I)XXX
(J)XV

Phase 2 Pass 3

T1/XV	T5	T6
I, XX, XXX	XV	0

T1 T2 T3

Final Pass (Phase 3)

A, XX, XXX 0 0
XV

b. Polyphase Merge (Unbalanced)

The polyphase merge is the name given to any unbalanced merge technique. The polyphase methods follow the Fibonacci series. This series can be represented as:

$$a_n = (a_{n-1}) + (a_{n-2})$$

or in simple symbols any term is the sum of the preceding two terms, i.e., 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610 etc.

Fibonacci developed this series in the early twelve hundreds, experimenting with the breeding of mice. Many have expressed doubt as to what the relationship between mice and sort is, but the series seems to work. The only place where the series is used unmodified is a three tape sort. Polyphase sorts use a modification of this series on higher order merges to determine the distribution of strings during the execution of the internal sort phase.

(1) Cascade Merge

The cascade merge was probably the first of the polyphase techniques to be implemented. The outstanding feature is the utilization of the read backwards feature to completely eliminate rewind time. The other effect is a higher merge order.

A 3-way balanced merge requires six tapes, a 3-way cascade requires four.

Method:

Strings are distributed in a predetermined manner to approximate the series. Reading backwards, a 3-way merge is executed for the number of strings on the smallest tape. When this tape is exhausted, a 2-way merge is executed with the output on the tape just completed. At the next exhaustion, a copy is executed to complete a pass.

Example 16:

The strings are distributed TA-16, TB-11, TC-5, TD-0.
- 5 strings are merged to TD (3-way)
- 5 strings are merged to TC (2-way)
- 3 strings are merged to TB (1-way - copy)

(1) Cascade Merge (Continued)

At the start of the second pass TA-0, TB-3, TC-5, TD-6.
3 strings are merged to TA (3-way)
2 strings are merged to TB (2-way)
1 string is merged to TC (1-way - copy)

At the start of the third pass TA-3, TB-2, TC-1, TD-0.
1 string is merged to TD (3-way)
1 string is merged to TC (2-way)
1 string is merged to TB (1-way)

At the start of the fourth pass TA-0, TB-1, TC-1, TD-1.
The rule here is the same as with balanced merges.

The final pass starts when not more than one sequence remains on any of the input tapes. This final pass merges one string from each tape to the final string on TA.

Example 16

CASCADE

3-Way Merge on 4 Tapes

	TA 16 Rd	TB 11 Rd	TC 6 Rd	TD 0 Wr	
1st pass	8 Rd 3 Rd 0	5 Rd 8 Wr 3	0 Wr 5 3	8 6 6	3 way 2 way copy
2nd pass	Wr 2 3 2	Rd 0 Wr 2 2	Rd 2 Rd 0 Wr 1	Rd 3 Rd 1 Rd 0	3 way 2 way copy
3rd pass	Rd 2 Rd 1 Rd 0	Rd 1 Rd 0 Wr 1	Rd 0 Wr 1 1	Wr 1 1 1	3 way 2 way copy
4th pass	Wr 1	Rd 0	Rd 0	Rd 0	

(3) Unbalanced Merge (used in Sort 2, 1401) 3-way on four tapes

Method:

Internal sort (phase 1) is executed in the normal manner putting strings out on two output tapes.

An adjustment phase is executed next distributing the strings so that they can be merged down to the next lower number of strings entry in the polyphase table. At this point the polyphase sort starts.

In this method read backwards is not used and tapes are rewound. Three tapes are merged to one until the smallest of the input tapes is depleted. The output tape and the input tape are both rewound and 3-way merging continues. The table is constructed so that there will be one string on each of three tapes for the final pass.

Example 17:

All numbers in example how resulting strings:

1	Shows the output of internal sort. 55 strings T3-26, T4-27.
2-4	Copy 11 strings to T1; 6 from T4 and 5 from T3. Copy 5 strings from T4 to T2. Tapes are now set to merge down to the next lower number of strings in the table.
5	Eight strings are merged (2-way) T2 and T4 to T1.
6	Eight strings are merged (3-way) T1 and T3 and T4 to T2.
7	The 31 strings remaining are distributed according to table T1-11, T2-13, T3-7. This merge pass leaves the tape with 17 strings (6,6,7), the next lower entry in the table.
8-11	The merging process is continued, each step merging the number of strings on the smallest tape.

At first glance the complication in the adjustment phase and the complexity of keeping track of the unbalanced merge would seem to negate the saving. In the running of many samples with identical input, Sort 2 (four tape) polyphase compares favorably in running time with Sort 2, 3-way merge (six tape). The results of high-way merges using like programs would be interesting to study. The indication was that the polyphase, even without the read backwards, is superior to the balanced

Example 17

POLYPHASE MERGE

Tape Distribution

	T-1			T-2			T-3			T-4				
S	S	Op	Rw	S	Op	Rw	S	Op	Rw	S	Op	Rw		
62	17	R	X	0			28	W	X	27	W	X		

Original Pass
1 develop 55 strings

55	0			5	W		28			27	R			
55	6	W		5			28			16	R			
55	11	W		5			33	R		16				
44	19	W	X	5			25	R		5	R			
31	11	R		13	W	X	7	R		0	R	X		

Intermediate Passes
2 S (3), from T4 to T2
3 S (6) from T4 to T1
4 S (5) from T3 to T1
5 M (8) from T3, T4 to T1
6 M (8) from T1, T3, T4 to T2

17	4	R		6	R		6	R	X	1	W	X		
9	3	R	X	2	R		4	W	X	3	R			
5	2	W	X	0	R	X	2	R		1	R			
3	1	R		1	W	X	1	R		0	R	X		

Final Merge
7 M (7) from T1, T2, T3 to T4

1	1	R	X	1	R	X	1	R	X	1	W	X		

Multiphase Table used in Polyphase

Total Strings	Distribution on Tapes		
	A	B	C
3	1	1	1
5	1	2	2
9	2	3	4
17	6	6	7
31	7	11	13
57	13	20	24
105	24	37	44
193	44	68	81
355	81	125	149
653	149	230	274
1201	274	423	504
2209	504	778	927

S - Strings
R - Read
W - Write
Op - Operation
Rw - Rewind (R)
T - Tape
(n) - Number of Strings
C - Copy
M - Merge

(3) Read Backward Polyphase Tape Ventilating Polyphase

Method:

During the internal sort phase, records are written on output tape with strings in alternating sequence. The first string on one tape must be ascending (A), the next descending (D). Strings are then placed on tape according to table or formula oscillating between (A) and (D) sequence. The strings must still use the table sequence. If strings fail because table varies, pending strings are used.

Tapes are read backwards and written for odds (no rewind until after final merge). Strings are merged until the smallest tape is depleted. At this point this becomes the output and the just completed tape becomes part of the merge. The process is continued until the merge input have one string each.

It will be noted that with the read backwards all (A) input to merge is written out in (D) sequence and all (D) input is merged and written in (A) sequence. This takes full advantage of the records read without having to vary as the records did.

A physical mark point above tape control is shot on each pass, except the last, only part of the file is passed. In fact in the three passes shown in Example #1, the large file is completely passed only once in the whole three passes. The method seems remarkably good regardless of how it arrives. The only difficulty is how to obtain a good manual pattern on a multi-drive tape computer. This problem seems minor. There is however, a difficulty in sensitiveness to phase 2. Phase 1 and 3 oscillations would be no more difficult than in a balanced case.

Example #1:

Strings 7-07, 6-12 and 6-02 enter the merge. Four strings are merged from T1, T2, T3 and T4, leaving 3-01, 2-05, 6-03, 6-01. The strings are merge from T1, T2 and T3 to T2, leaving 1-01, 0-02; 2-04, 1-06. One string is merged from T1, T2, T4 to T1, leaving 0-01, 1-02, 1-03, 1-04 (the sequence for the first pass).

* J. L. Gilstad, Minneapolis-Honeywell, presented at the Association of Computing Machinery Symposium, November 29 and November 30, 1952.

Example 12

READ BACKWARDS POLYPHASE

OF

OSCILLATING [illegible]

Output of Phase 1

T1	T2	T3	T4
1-A	2-D	3-D	0
4-D	5-A	6-A	
7-A	8-D	9-D	
(7)10-D	(6)11-A	(4)12-A	
13-A	14-D		
15-D	16-A		
17-A			

Phase 2, pass 1

T1	T2	T3	T4
1-A	2-D	0	E(17,16,12) (D)
(3) 4-D	(2) 5-A		(4) F(15,14,9) (A)
7-A			G(13,11,6) (D)
			H(10,8,3) (A)

Phase 2, pass 2

T1	T2	T3	T4
(1) 1-A	0	I(7,5,H) (D)	E(17,16,12) (D)
		(2)J(4,2,G) (A)	(2) F(15,14,9) (A)

Phase 2, pass 3

T1	T2	T3	T4
0	(1) K(1,J,F) (D)	(1)I(7,5,H) (D)	(1) E(17,16,12) (D)

Phase 2, pass 4

(K,I,E) (A)

C. Random Access Techniques

With the advent of large high speed random access devices, some new aspects have been opened up in sorting. Some of these affect internal sorting almost as much as merging, but since the basic techniques used in internal sorting aren't affected, random access techniques are included under merging.

Random access devices eliminate the need for some sorting in cases where files are permanently stored in them. But in this discussion, it is assumed that the file is used as a scratch pad for files external to the disk that must be sequenced.

Tag sort remains an extremely important consideration. Seek time is in the same category as rewind time on tape as non-productive time unless overlap can be obtained. In one example of improvement of seek time, the run time of a sort was decreased by 1/3 with the addition of an arm on the disk unit.

There remains a need in this area for more study to find better techniques to show the full potential of sorting on disk units.

(1) Tag Sorting

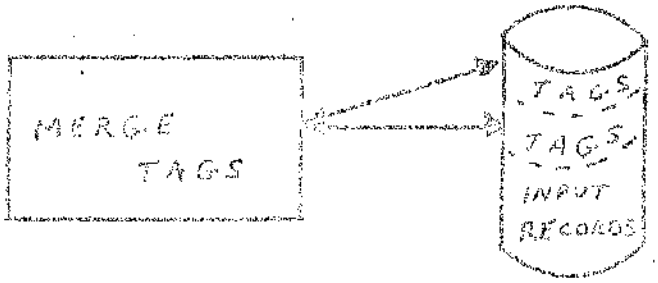
Tag sorting has been widely used in internal methods instead of moving the record in memory (this is strictly applicable to disk). The tag is formed by taking the control field and adding to it the disk address.

Phase 1:

1. Reads records from tape or file.
2. Places records on disk.
3. Takes control field from record, adds disk address and forms a small control record.
4. Builds a "G" from the tag records.
5. Writes "G" on disk file in maximum of five files.

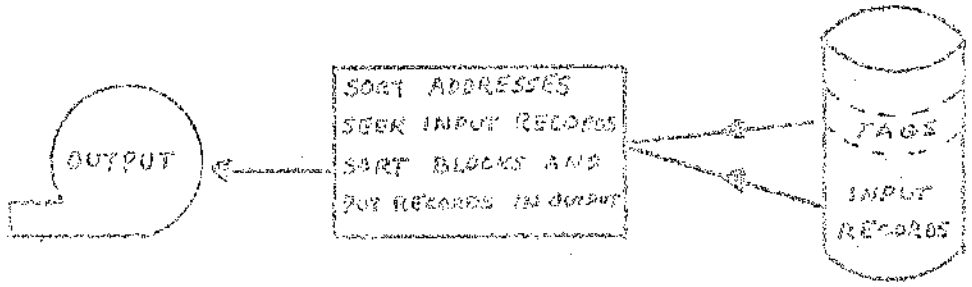
(1) Tag Sorting (continued)

Phase 2



1. Merges tag files from disk as if they were from tape.
2. Final phase 2 pass will strip control fields from tag records leaving the disk addresses only (at the user's option sort 3 can end at this point).

Phase 3 (optional)



1. Disk addresses are brought into memory in some multiple of the output blocking factor.
2. Addresses are then sorted into ascending sequence to minimize seek time.
3. Input records are then read from the disk and placed in the proper sequence, then written on tape or in disk memory.

The seek time in this method seems to almost negate the effectiveness of the technique. However, the implementation is on one of the slower disk units. With the announcement of new equipment, this method may still be feasible.

(2) Variable Range Under Disk Sort *

Example:

1410-1301 Sort is a partial example, however, order is determined by 1410 core size. The standard merge for this sort is a 12-way merge. This must be as close to this technique on any implemented one. In many volume ranges, it compares favorably with the 7070 Sort 90.

Considerations:

1. Core size (ability to handle input blocks).
2. Seek time and transmitter time.
3. Conversions block size as specified by file considerations.

Phase 1:

1. Could sort G's determined by record length, file and memory consideration as large as convenient.
2. Strings would be written as if they were being placed on one tape.
3. All string locations placed in a table as to position in file and high and low value.

Phase 2:

Using the information gained during phase 1, a formula, built around hardware considerations, could be solved for the most efficient order of merge. It is not necessarily a balanced merge; a polyphase could be very effective.

1. Table of sequences and address should be scanned and sorted to place all natural strings together.
2. When most efficient merge is determined, table should be broken to define merge files.
3. Formula should be evaluated prior to each pass to have proper order.

Phase 3:

Would execute as a standard sort.

* Delaney, J.R., Midwestern Napkin (publication unknown), August 1961, "A Sort Designed for the IBM 1410-1301 Disk Storage System (Five)"

Section IV

Conclusion

The sort of the future will probably look no more like our sorts today than a computer sort resembles a mechanical sort. One element that some future sorts will have is the use of multiple techniques in one sort package, i.e., instead of just picking the order of merge. The package will evaluate the specifications or parameters and decide what type of sort will produce the best results. Sort will be an operation (high level macro or Cobol sort verb). The programmer will use the sort macro, and a specific tailored sort will be generated as a part of his processing job.

Sorts and sort techniques can be very difficult to evaluate. The random files that are used to test the effectiveness of sorts just don't exist in installations. It is recommended that Sort Census be studied to show the pitfalls that arise when comparing sorts. Much of the analysis of sort techniques centers around compares or other criteria. I feel that the only true evaluation (including the computer and package) is the number of records sorted per computer dollar.

Sorting by itself is a non-productive though highly important aspect of data processing. The purpose sort is the world is no sort. This is the work that the System Engineer figures out how to avoid. No reports to management are produced while sorting. But many reports are highly dependent upon sorts. With the acceptance of management by exception, the cause of sorting will receive another blow. Sorting will decrease in the amount of time spent in the data processing installation, but probably never be eliminated.

* Jean Sammet, Datamation's National Bulletin #7, May 15, 1961.

Section V

A GLOSSARY OF SORTING AND SORTING TERMS

ASSEMBLING PROGRAM - The set of instructions by which a generalized program modifies and completes the set of instructions which will be utilized in performance of one specific sort run.

ASCENDING SORT - A sort in which the final sequence of records is such that consecutive keys compare, "greater than or equal to".

B - Internal sort blocking, number of records between inter-record gaps on tape. BI input blocking, BO output blocking.

BACKWARD READ - A feature available on some magnetic tape system whereby the magnetic tape units can transfer data to computer storage while moving in a reverse direction. Normally used, if available, during the external sort phase to reduce rewind time.

BALANCED SORTING - See: (T/2)-way merging.

BIASED DATA - A distribution of records in a file which is non-random with respect to the sequencing criteria. Biased data may affect sorting time, depending on the technique used during the first pass of the data. Normally, it is desirable to take advantage of existing sequence bias (whether ascending or descending) during the first pass.

CASCADE MERGING - A technique used in a sort program to merge strings of sequenced data. Given T work tapes, merging is performed on T-1 on part of the data, T-2 on parts of the data, and so on. Strings of sequenced data are distributed in a Fibonacci Series on the work tapes preceding each merge. The effective power of the merge varies between T-1 and T/2, but is all cases is less than the power of the polyphase merge. See: effective power of the merge.

CHECKPOINT - Also: restart point. The point at which a restart (or rerun) can be initiated. Memory, registers, and the position of tapes are recorded at this point. See: rerun point.

CORE SIZE, "SORT" - An optional card in COBOL 61.

COLLATING - Sequencing a group of records by comparing the key of one record with another record until equality, greater than, or less than is determined.

GLOSSARY (continued)

COLLATING SEQUENCE - The sorting sequence; a description of the sort key for a file of records.

COLLATING SORTING - A sort which uses a technique of continuous merging of data until one sequence is developed.

CONTROL CARDS - Sort program control cards.

CONTROL CARD - A card which is used to specify the parameters for a sort.

CONTROL FIELD, CONTROL AREA FIELD - A continuous group of characters within a record which form all or part of the control word K_1, K_2, K_n.

CONTROL WORD - A word in the key.

CONTROLS - Sort Sequencing controls. Sort key.

CONVENTIONAL SORT - A sort in which the final sequence of records is such that the successive keys compare "less than" or "equal to".

DIGITAL SORTING - A sort which uses a technique similar to sorting on calculator machines (e.g. IBM Sorter). The elapsed time is directly proportional to the number of characters in the sequencing key and the volume of data. Also "radix sort".

DISK SORTING - A sort program that utilizes disc memory for auxiliary storage during sorting.

DRUM SORTING - A sort program that utilizes magnetic drums for auxiliary storage during sorting.

EFFECTIVE PASSES OF THE SORT - Equal to $k^{\frac{1}{n}}$, where k is the number of input records and n is the average number of times each element of data is read.

FIBONACCI SORTING - A series where the current number is equal to the sum of the two preceding numbers: i.e., 1, 2, 3, 5, 8 and so on. Some sort programs distribute strings of data onto work tapes so that the number of strings on successive tapes form a Fibonacci series.

FIRST PASS OBJ CODING - Computer instructions created by the programmer, in assembly or absolute form, which are executed by a sort during the first pass of the file after input program has been loaded but prior to execution of first pass sequencing instructions.

FIXED SIZE RECORDS - Serviceable file elements each of which has the same number of words, characters, bits, fields, etc. (vs. variable-length records.

G - Area in memory used to sort records into sequenced strings. In some cases 2 "G" areas are used. It is desirable to increase the string length to decrease the number of phase 2 passes (sometimes called MIS).

GENERALIZED SORT - A sort program which will accept the introduction of parameters at run time and which does not generate a program.

GENERATED SORT - A production program which was produced by a sort generator.

HASH TOTAL - A total taken on an arbitrary field of the record for control purposes only. The hash total taken during one pass may be compared to one taken during a previous pass to aid in detecting any mishandling of records.

I/O AREAS - Same as Buffers.

INPUT FILE(S) - Record(s) representing a file in arbitrary sequence to be introduced into a sort/merge program.

INTERMEDIATE RECORD - See: strings.

INTERMEDIATE PASS - Any phase of a merging operation which, because of the number of strings, or otherwise, does not reduce the file to a single sequenced string.

INTERMEDIATE PASS OBJ CODING - Computer instructions created by the programmer, in assembly or absolute form, which are executed by a sort during the intermediate passes of the file after the execution of instructions for comparisons of keys but prior to output of the selected records. May also be executed during the internal sort, but after the selection of records.

KEY - Same as record.

KEY - Also, sequencing key; criterion; sequencing criteria. The fields in a record which determine, or are used as a basis for determining, the sequence of records in a file.

LABEL - A record, usually located at the beginning of a file, which contains information related to identification, operation, and control of the file or [illegible] to a record which is part of the file itself.

LAST PASS EXIT [illegible] - Computer instructions written by the programmer, in assembly or absolute form, which are executed by a sort during the last pass of the file after the final merging instructions have been executed but prior to unloading the output record.

MAGNETIC TAPE SORTING - A sort program that utilizes magnetic tapes for auxiliary storage during a sort.

MAJOR KEY - The most significant key in a record.

MERGE - A program that performs merging. [illegible] order of merge.

MERGING - The forming of a single file of sequenced records from two or more files of sequenced records.

MULTIFILE SORTING - The automatic sequencing of more than one file, based upon separate parameters for each file, without operator intervention.

MULTIPASS SORT - A sort program which is designed to sort more data than can be contained within the internal memory of a central computer. Intermediate storage, such as disk, tape, drum, etc., is required.

MULTIREEL SORTING - The automatic sequencing of a file having more than one input reel, without operator intervention.

N - Number of records to be processed by the sort. N MAX = maximum number of a type of records. [illegible] is capable of processing.

OPTIMUM MERGING PATTERN - The determination of the sequence in which specific [illegible] input tapes in a file should be processed so as to minimize the total number of merge passes required to create a single file of sequenced records.

ORDER OF MERGE - The number of input files to a merge program. Also: power of the merge.

GLOSSARY (continued)

OUTPUT [illegible] - Tapes containing a file in specified sequence as a result of a specific sort/merge process.

OWN CODING - Special coding, provided by the programmer, which is integrated with sort/merge coding.

PADDING - One or more records, consisting only of arbitrary characters added to the file to fill out a block (HP - high padding, LP - low padding).

PARAMETERS - The response to the requirement for specification for a sort/merge generator. Parameters are used to fix input and output formats, computer configurations, location of keys, and so on.

PASS - The processing of each file record once for the purpose of reducing the number of strings of sequenced records and increasing the number of sequenced records per string.

PHASE - An arbitrary segmentation of a sort program. Many sorts are segmented into three phases: initialization phase, internal phase, and merge phase.

POLYPHASE MERGING - A technique used in a sort program to merge strings of sequenced data. Given T work tapes, merging is performed at the power of T-1. The effective power of the merge varies between T-1 and T/2, depending on the amount of input data and the number of strings.

POWER OF THE MERGE - Also [illegible] the number of inputs to a merge program. [illegible] effective power of the merge.

PROCESS LIMITED - Processor limited. Also computer limited. A sort program in which the execution time of the internal instructions determines the elapsed time required to sort. See tape limited.

RADIX SORT - See digital sorting.

RECORD - The basic element of a file such that the sorting of a file constitutes the reordering of file records; also referred to as "item".

REPLACEMENT SELECTION TECHNIQUE - A technique used in the internal portion of a sort program. The results of the comparisons between groups of records are stored for later use. A selected record is placed on the output and a new record replaces the selected record. Given P records, a string is obtained with [illegible] the average string length for random data is 2P records.

RESTART POINT - A point at which a computer program can be restarted in case of computer failure or other reason for inability to continue, without loss of processing accomplished prior to the interruption. See: checkpoint.

RESTART - The return to a previous point in the program where processing may begin. The previous point may be the beginning of the program or it may be a checkpoint. See: checkpoint; return point.

RESTART POINT - See: checkpoint.

RUNNING TIME - Elapsed time consumed by a sort/merge program for restoring intermediate and final tape files to original position.

SCRATCH TAPES - See: work tapes.

SEQUENCE - Any number of records (or tags) in the desired sort order.

SEQUENCE CHECK, SIDE TEST - That point in a file between the end of one string and start of another.

SEQUENCING CRITERIA - See: key.

SHUFFLE - A method of internal sorting where records are moved to permit the insertion of records; also called insertion method.

SORT - The copying of a file of records into a corresponding file in a specified sequence.

SORT, EXTERNAL - The second phase of a multipass sort program, where strings of data are continually merged until one string of sequenced data is formed.

SORT, INTERNAL - The sequencing of one or more records within the central computer memory; the first phase of a multipass sort program.

SORT GENERATOR - A program which produces a sort program for production running.

STRING - A group of sequenced records, normally stored in auxiliary computer storage; i.e., disc, tape, or drum.

(2-1)-WAY MERGING - Sometimes called (2). A technique used in a sort program to merge strings of sequenced data. For tape systems that permit backward reading, the effective power of the merge is equal to 2-1.

(2/2)-WAY MERGING - A technique used in a sort program to merge strings of sequenced data. The power of the merge is equal to 2/2.

ADD - The address of a record on disc, drum, or tape, or memory. Sometimes the control field is included as part of a key record.

BALL SORTED - Also: B/O Method. A sort program in which the effective transfer rate of tape units determines the elapsed time required to sort. Cf: process limited.

BINARY TAPE MERGING - See: replacement-selection technique.

BUBBLING SORTING - See: replacement-selection technique.

CHARACTER-COUNT RECORDS - Recordable file elements for which the number of words, characters, bits, fields, etc., is not constant.

CON SECTION TIME - See: (2/2)-way merging.

DEPTH OF THE MERGE - See: power of the merge.

DUMP TAPE(S) - Also: scratch tape. Tape(s) used to store intermediate runs when doing a sort program.

DYNAMIC SORT PROCESS - See: replacement-selection technique.

Section VI

BIBLIOGRAPHY

Cooke, W. C., A Tape File Merge Pattern Generator, Communications of the Association of Computing Machinery, Volume 6/Number 5/ May 1963 - pages 227-230.

Elvash, R. S., An Analysis of Unbalanced Tape Sort Techniques, IBM, publication unknown.

Falkin, J. and Savastano, S., Jr., Sorting with Large Volume, Communications of the Association of Computing Machinery, Volume 6/Number/5 May 1963 - pages 240-244.

Flores, I., Analysis of Internal Computer Sorting, Journal of the Association of Computing Machinery, Number 8/ January 1961 - pages 41-80.

Frank, R. M. and Lazarus, R. B., A High-Speed Sorting Procedure, Communications of the Association of Computing Machinery, Volume 3/Number 1/ January 1960 - pages 20-22.

French, R. C., Computer - Planned Collates, Communications of the Association of Computing Machinery, Volume 6/Number 5/ May 1963 - pages 225-227.

Friend, E. H., Sorting on Electronic Computer Systems, Journal of the Association of Computing Machinery, Volume 3/ July 1956 - pages 134-168.

Gilstad, R. L., Polyphase Merge Sorting - An Advanced Technique, Proceedings of the Eastern Joint Computer Conference/1960 - pages 143-148.

Gilstad, R. L., Read-Backward Polyphase Sorting, Communications of the Association of Computing Machinery, Volume 6/Number 5/ May 1963 - pages 220-223.

Goetz, M. A., Internal and Tape Sorting Using the Replacement-Selection Technique, Communications of the Association of Computing Machinery, Volume 6/Number 5/ May 1963 - pages 201-206.

Goetz, M. A., and Toth, G. S., A Comparison between the Polyphase and Oscillating Sort Techniques, Communications of the Association of Computing Machinery, Volume 6/Number 5/ May 1963 - pages 223-225.

Goetz, M. A., Organization and Structure of Data on Disk File Memory Systems for Efficient Sorting and Other Data Processing Programs, Communications of the Association of Computing Machinery, Volume 6/ Number 5/ May 1963 - pages 245-248.

Gotlieb, C. C., Sorting on Computers, Communications of the Association of Computing Machinery, Volume 6/Number 5/ May 1963-pages 194-201.

Hall, M. H., A Method of Comparing the Time Requirements of Sorting Methods, Communications of the Association of Computing Machinery, Volume 6/Number 5/May 1963 - pages 259-263.

Hibbard, T. N., An Empirical Study of Minimal Storage Sorting, Communications of the Association of Computing Machinery, Volume 6/ Number 5/ May 1963 - pages 206-213.

Hildebrandt, P. and Isbitz, H., Radix Exchange - An Internal Sorting Method for Digital Computers, Journal of the Association of Computing Machinery, Volume 6, April 1959 - pages 156-63. Rev. Communications of the Association of Computing Machinery, Volume 3/April 1960 - page 2[illegible].

Hubbard, G. U., Some Characteristics of Sorting in Computing Systems Using Random Access Storage Devices, Communications of the Association of Computing Machinery, Volume 6/Number 5/ May 1963 - pages 248-255.

Isaac, E. J. and Singleton, R. C., Sorting by Address Calculation, Journal of the Association of Computing Machinery, Volume 3/ July 1956 - pages 169-174.

Malcolm, W. D., Jr., String Distribution for the Polyphase Sort, Communications of the Association of Computing Machinery, Volume 6/ Number 5/ May 1963 - pages 217-220.

[illegible], Multiphase Sorting, Communications of the Association of Computing Machinery, Volume 6/Number 5/ May 1963 - pages 214-217.

McCracken, D. D., Weiss, H. and Lee, T. H., Sorting, Programming Business Computers, 1959 - pages [illegible].

Kelsey, J. L., A Sort Technique for the IBM 1410-1301 Disk Storage System (Rough), IBM Midwestern Region (publication unknown), August 1961.

Taylor, [illegible] Applications Bulletin, Journal of the Association of Computing Machinery, Number 2/ October 1959 - pages 459-468.

Patterson, J. L., The IBM 1401 Sort Test, Communications of the Association of Computing Machinery, Volume 6/Number 5/ May 1963 - pages 253-259.

Sorting Methods for IBM DP Systems, IBM General Information Manual, F28-8 03, 1959 - page 32.

Sort Power, IBM Mechanical Reference Bulletin 47, May 28, 1961.