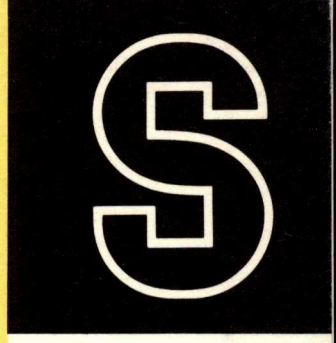


IBM

General Information Manual

Sorting Methods for IBM Data Processing Systems

SORTING METHODS

FOR IBM DATA PROCESSING SYSTEMS

Address comments regarding this publication to
Programming Systems Publications, IBM Corporation, P. O. Box 390, Poughkeepsie, N. Y.

TABLE OF CONTENTS

INTRODUCTION

Sorting applications involve a wide range of machines, problems, and special data considerations. Though sorting is based on simple concepts, efficient sorting is a complex problem because of the variety that exists in applications and because of the interaction of data, machine, and technique. The interaction of these factors is so involved that it is difficult to determine precisely the best possible sort for a given job.

This manual describes some basic methods of sorting and points out general considerations and rules that will make it easier to evaluate different approaches to the problem of sorting. This material is presented in non-technical fashion; the only formulas and tables included are for the purpose of making estimates of time. A list of terms and abbreviations, with some explanation, appears as an appendix to this manual. The terms listed are used in the generalized sort programs and the manuals which describe them. They should not be considered definitions but can be more accurately described as the conventions that are used by those who have developed the sorting programs. The reader should consult this list freely for a better understanding of the material presented.

METHODS OF TAPE SORTING

The use of computers and magnetic tape has not altered the sorting procedure as much as it has emphasized the need for speed and the ability to handle a very large number of records in one sort. The basic methods of sorting can be demonstrated by reverting to an example that has been used many times, the sorting of playing cards.

If a person is given one complete deck of cards and asked to put them in order, the procedure is a simple one. Most people will make an initial distribution by suit, creating four "files" of equal size. After that each "file" can be sorted by holding the 13 cards of a suit in one hand, while the cards are shifted about and placed in order. As soon as each of the four suits has been ordered, the four are stacked together and the job is completed. In sort terminology this was accomplished by the following basic sorting methods: a distribution, an internal sort, and then a final merge of four sorted files.

A more realistic picture of most sorting problems is created if the above problem is complicated slightly. Assume that 52 cards are taken from a stack of cards which contains four decks. The procedure outlined above might work for this second case, but there are good reasons to doubt that it will. It is extremely unlikely that the initial distribution by suit will produce four groups of equal size. In fact, there might not even be four such groups or "files" and the cards selected may contain two to four equal cards. If we also add the stipulation that fifteen is the maximum number of cards that can be held at one time, the solution for the second case is considerably more difficult. It can still be solved by a combination of methods, but either the sequence of operations will have to be altered or the initial distribution modified.

The following sections show in some detail how a few basic methods, the merge, the distribution, and the internal sort, can be used and modified to fit the various problems in sorting.

SORTING BY MERGING

The merge has been used as the basis for many specific sorting programs and most of the generalized sorting routines. This brief description outlines only the principles. A detailed discussion of the coding and logic of the merging operation within the older sorting programs is given in the manuals on The 705 Sorting Programs, SORT 53A (Form 328-6968) and SORT 57 (Form 328-7880). An improved technique will be found in the sorts for the 705III and 709.

The two-way tape sort provides a simple example of merge sorting. If a given

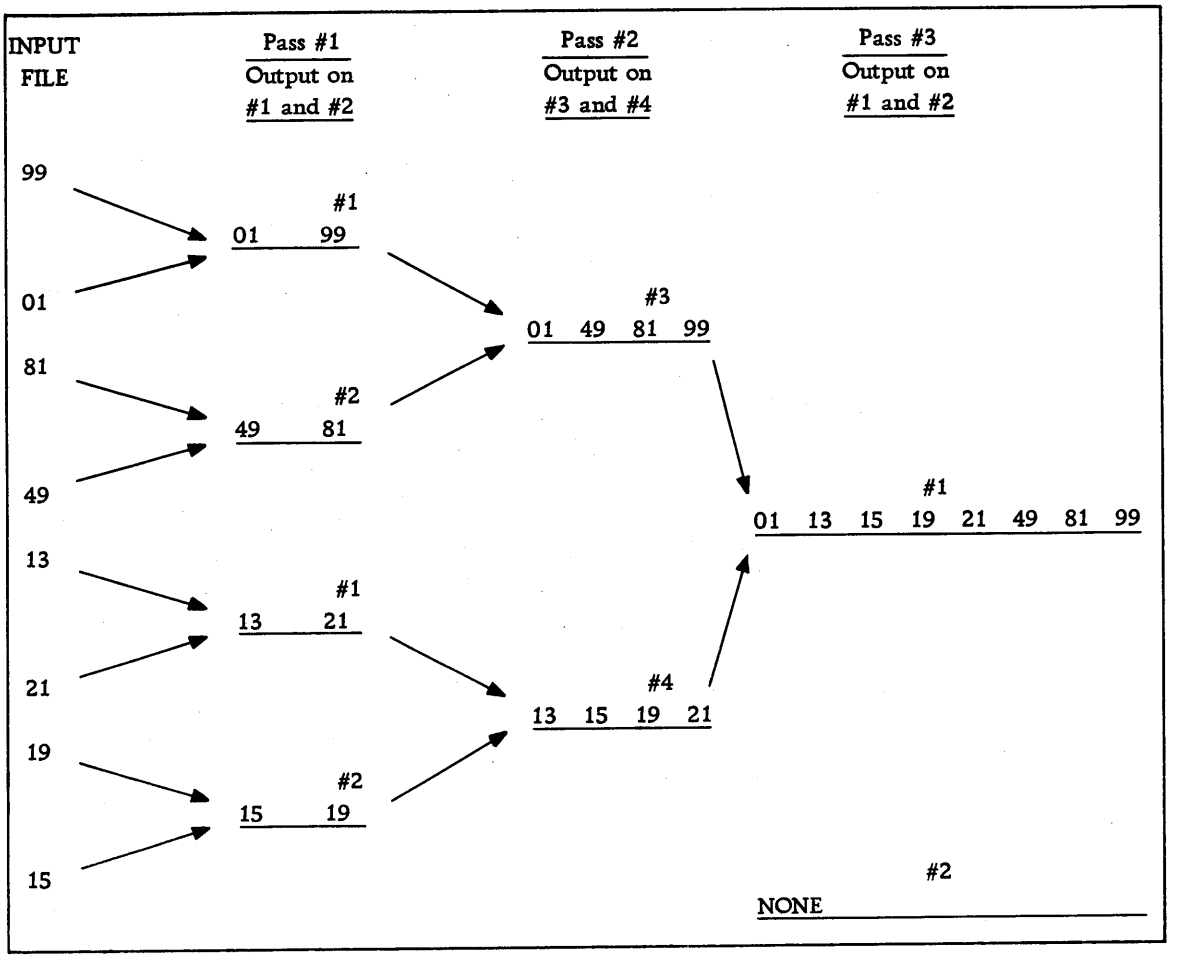
computer has the ability to read and write tapes and select the smaller of the items brought in, a series of random numbers can be sorted as follows:

> The first pass of the sort merges two single items to create sequences of two items.
>
> The second pass, using the output of the first pass as input, merges a pair of these two-item sequences, one from each of the two input tapes, and writes four-record strings on the output tapes.
>
> Continuing such a merge by repeated merging passes will eventually place all records in the file into one sequence.

Figure 1 shows how a set of numbers is gradually put into sequence by use of a simple two-way merge. In the first pass, the records are written out on tapes #1 and #2, each of which will then contain two-record sequences at the end of the pass. The tapes are rewound and the output of the first pass becomes the input for the second merge pass. The first two record strings from tapes #1 and #2 combine to make a four-record sequence that goes onto tape #3. Then the second pair of two-record sequences is merged and stored on tape #4. The merging process for the third pass is like that for the second pass, except that the sequences read in and written out are twice as long as they were in the preceding pass; input comes now from tapes #3 and #4 and output is written onto tapes #1 and #2. Each new pass doubles the length of the input sequences. The file is sorted when the number of sequences is reduced to one. The final output will all be on a single tape. In the example, the sorted records are stored on tape #1 after three passes.

In a simple merge such as illustrated in Figure 1, the number of merging passes is determined by the number of records to be sorted and the order of the merge. A two-way merge without an internal sort will develop a sequence of 2^n records in n merging passes, i.e., 8 items in three passes, 16 in four, 32 in five and 1,024 in ten passes. As the number of records to be sorted increases to a larger number, such as one might realistically expect in data processing applications, the number of passes will become quite large.

Therefore, in actual practice, a sorting program usually includes some modifications that will reduce the number of merging passes. The most common method is to precede the merge by one of the internal sorts that are described in a separate section.



SORTING BY A TWO-WAY MERGE

Figure 1

The merging principle can be used by sorting programs which employ various numbers of tapes. The combinations most frequently used are 2 X 2, 3 X 3 or 4 X 4, i.e., input from a given number of tape files is merged and written onto the same number of output tapes. Whenever the number of input files equals the number of output tape files, a merge is called "symmetrical." It is not necessary, however, that the merge be symmetrical; a 4-3-4-3 type of merge can be used if the equipment available makes it desirable. An asymmetrical merge has not been written as part of any generalized program because such a program reduces the maximum file size to that which can be sorted on the side with the smaller number of tapes, (See "Maximum File Size," page), and also requires more programming. Actual experience has shown that most commercial installations have more tape units available than the minimum for an efficient symmetrical sort.

All of the above merging methods move the records back and forth between tapes so as to require no tape changing. A few merge sorts have been written which merge several files and write all the output onto one tape, the objective being to achieve a maximum order of merge for a given number of tapes. As soon as this output tape is filled, it is replaced and saved so that it can serve as input for the next merge pass. Every tape written as output must be removed and then mounted again on the read side before the next pass can be started.

There are several disadvantages of this last merging technique which must be weighed against the higher order merge and greater record capacity that it does offer. The time and work required to make the reel changes are not trivial. It may also be difficult to overlap reading and writing because most equipment now uses separate channels, busses or buffers for input and output. Whenever this is the case, it is difficult to connect the desired large number of tapes on the one input side without losing the facility of simultaneous reading and writing. If the files to be sorted are not extremely large, the symmetrical merges as used in the generalized sorting programs will be much more adaptable to machine schedules and the usual organization of tapes, and will not sacrifice much, if any, running time.

DIGIT SORT

Sorting by digit in its basic form requires a large number of tapes. If a computer had a tape unit to correspond to the hopper and each of the 13 pockets of the card sorter, it could be made to sort in exactly the same way in which the card sorter works. The problem of sorting 52 playing cards selected at random from four decks is easily solved if 13 pockets (tapes) are provided for the output. Such a solution is generally unrealistic for tape sorting.

For those applications in which the sort is made on a short control field consisting only of numerical information, the digit sort may be an appropriate choice. Alphanumerical information could conceivably be sorted by an extension of this method, but it would require a large number of output units or multiple sorting passes and is therefore out of the question. Alphanumerical and/or long numerical control words as well as files which contain some sequencing should probably always be sorted by the merge type sort.

The most simple digit sort reads the records from one tape and distributes them among ten output tapes according to the value of a single character in the control field. The first distribution is made by the units position of the control field, the second by the tens position, etc. The number of tape passes is determined by the length of the control field. The total number of passes will be the length of the control word plus one (CW + 1) if there is no limitation on the number of tapes, i.e., not less than 20 tapes. The additional pass is required to create

one single final output tape. Such a digit sort has been written for sorting on a short and entirely numerical control word; this program uses a 705 with two tape control units and 20 tapes. This arrangement will permit simultaneous reading and writing and will therefore operate in the shortest time possible for a digit sort. The programming is relatively simple and the process time correspondingly low.

In actual practice, 20 tape units are not usually available, so some compromise must be made which reduces the required number of tape units and increases the number of tape passes. One program of this kind uses 10 tape units and requires 2 passes per digit in the control word. Read and write time can always be overlapped, but two tape passes per character are required for a total of 2CW + 1 passes. Any digit position is sorted by distributing the file during the first pass among five tapes, two digit values being assigned to each tape.

The example in Figure 2 shows how a digit sort can be accomplished by this method. During pass #1, the input file is written onto the odd numbered tapes as shown. Then in the second pass, the records are read back in and, while still sorting on the units digit, written out on the even numbered tapes. Tapes are read in the order 1-3-5-7-9 to insure that the sequences 0-1, 2-3, etc. can be set up while writing the even tapes. One odd and one even pass are required for each digit in the control word. After the second pass has been made for the most significant digit, one additional tape pass is used to combine and write the entire file onto one tape. The example shows the movement of ten two-digit items from the time they are read into the final writing of the entire file on one tape. The tape time for this will be 2CW + 1.

It is possible to reduce the number of tapes used to less than 10 by grouping more than two digits per tape. Digital values assigned to each tape unit must allow ordering the digits into the order 0-9 in two passes. This method depends on the ability to distribute the digital values assigned to each tape in the first pass, to the other tapes, not more than one digit to a tape; thus no less than seven tapes are necessary. The problem of encountering an unequal distribution of records among the various tapes is increased when using less than 10 tape units, but the number of tape passes is still 2CW + 1.

TAPE UNITS		0	2	4	6	8		1	3	5	7	9
Digits assigned to each tape		(0-1)	(2-3)	(4-5)	(6-7)	(8-9)		(0-2)	(4-6)	(8-1)	(3-5)	(7-9)
		INITIAL INPUT										
Pass #1	Read	98,89	76,67	54,45	32,23	10,01	→Write	32 (7)*	76 (3)	98 (1)	45 (6)	89 (2)
								10 (9)	54 (5)	01 (10)	23 (8)	67 (4)
								↓	↓	↓	↓	↓
Pass #2	Write	10 (2)	32 (1)	54 (4)	76 (3)	98 (5)	←Read	32 (1)	76 (3)	98 (5)	45 (7)	89 (9)
		01 (6)	23 (8)	45 (7)	67 (10)	89 (9)		10 (2)	54 (4)	01 (6)	23 (8)	67 (10)
		↓	↓	↓	↓	↓						
Pass #3	Read	10 (1)	32 (3)	54 (5)	76 (7)	98 (9)	→Write	01 (2)	45 (6)	10 (1)	32 (3)	76 (7)
		01 (2)	23 (4)	45 (6)	67 (8)	89 (10)		23 (4)	67 (8)	89 (10)	54 (5)	98 (9)
								↓	↓	↓	↓	↓
Pass #4	Write	01 (1)	23 (2)	45 (3)	67 (4)	89 (6)	←Read	01 (1)	45 (3)	10 (5)	32 (7)	76 (9)
		10 (5)	32 (7)	54 (8)	76 (9)	98 (10)		23 (2)	67 (4)	89 (6)	54 (8)	98 (10)
		↓	↓	↓	↓	↓						
Pass #5	Read	01 (1)	23 (3)	45 (5)	67 (7)	89 (9)		FINAL OUTPUT				
		10 (2)	32 (4)	54 (6)	76 (8)	98 (10)	→Write	01,10	23,32	45,54	67,76	89,98

*The number ( ) indicates the sequence in read/write cycle when this item is used.

DIGIT SORTING

Figure 2

DISTRIBUTION SORTING

The two card sorting examples used earlier demonstrated both that a distribution, or block sorting, is a very good way to simplify the sorting of a group of N items, and also that this system should only be used when there is some advance information about the file to be sorted. A program of this type must be specifically designed for one certain application because an unforeseen distribution can disrupt such a sort. The distribution sort can handle alphanumerical information more readily than a digit sort, and some versions do operate in less time than a merge sort would require.

A distribution can be considered the reverse of a merge. The objective is to divide the file into several small groups which can each be sorted with an internal sort. On the first distribution pass, the file is divided into several blocks of items according to the most significant control information. On the second pass, each block is further subdivided according to the next most significant information. If this type of separation is repeated a sufficient number of times, the file will be completely ordered as the blocks are reduced to individual items. The more practical method uses a distribution to subdivide the blocks and sections until

each subsection can be sorted by an internal sort.

The number of records that the 705 and 709 can internally sort (G) will probably range from 30 to 800, depending on the record length and the amount of memory available. The average number of records (S) which remain in one section or block under optimal conditions after P distribution pass will be N/T^P

$$S = N/T^P$$

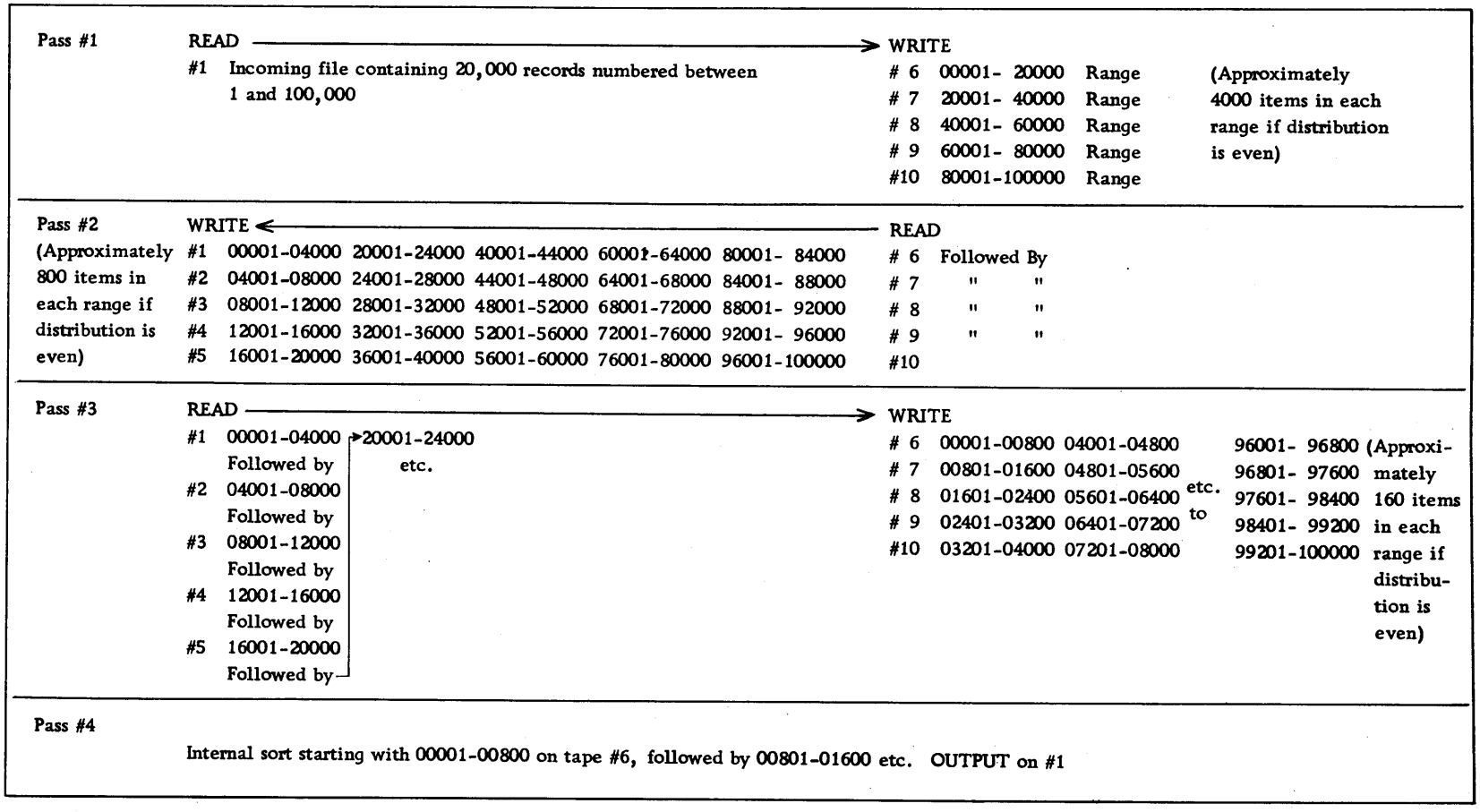
where T represents the number of output tapes during each pass and N represents the number of records. It is apparent that the maximum value of S can be reduced to a number less than G without going through many passes when $T \geq 5$. The numbers of tapes and passes are usually selected so that S is reduced to a number like $\frac{2G}{3}$ or $\frac{G}{2}$ to provide for the variations in S which will be found between blocks within a given file and for different files.

Figure 3 shows the example of a file of 20,000 items subdivided so as to form groups of approximately 160 items each. If we assume fixed length records and that the distribution creates equal blocks, the block size at the end of three passes will be $\frac{20,000}{5^3}$ or 160 record blocks if they are all of equal size. If the program in this example has provisions to internally sort groups less than or equal to 300 items, the fourth pass becomes an internal sort. This technique presupposes that the actual number of records within a group does not exceed the maximum of 300 records. In the example, the reading for the last pass is done on a sequential basis: all records in the first block (00-800) are read in from tape #6 and sorted. Then the 800-1600 block from tape #7 is brought in and the procedure continued until the last block on tape #10 has been finished.

To summarize:

1. As already indicated, the problem in writing a distribution sort consists of ascertaining the characteristics of the file to be sorted within the total possible range of the control word. If the records of the file are irregularly dispersed within that range, it may cause one output tape to overflow and may also add to the number of passes which must precede the internal sort.

2. The number of tape units on each channel of the tape control limits the subdivision factor to be achieved in any one pass. If the input for any pass does not require all tape units on that channel, it is theoretically possible to achieve a higher factor of subdivision by using the remaining tapes on the input channel as output tapes. In this case, however, the overlapping of reading and writing operations will have been, at least partially, sacrificed.



DISTRIBUTION SORT WITH AN INTERNAL SORT AS FINAL PASS

Figure 3

3. The distribution sort may prove to be a good sorting method for two reasons:

   First, process time is low since only a small part of the control information is tested during each pass.

   Secondly, less tape time will be required if the number of distribution passes is smaller than the number of merging passes which would be used in the merging sort method.

In the example in Figure 3, the number of passes would be the same for both methods when memory space permits sorting 167 records internally. If, however, G is less than 167, the distribution sort method will require one pass less than the merge sort. If the characteristics of a file are either unknown or changing between jobs, some method other than the distribution method should be selected.

INTERNAL SORTING

It frequently is necessary to sort or order a group of records that are located in the storage of a computer. The number of records involved may vary from very few to a large number, possibly as many as 1000 or more.

Internal sorting is usually done in conjunction with some other processing. Routines for internal sorting may be found in programs which range from compilers and assembly programs to generalized and specific sorting programs. By including an internal sort as the first phase of a merge type sort, four, five or even six or more merge passes can be eliminated. The savings for a given file depend on the number of items internally sorted and the order of merge which it precedes. This section will deal primarily with internal sorting methods that have been developed for the generalized sorting programs.

Any internal sort that is selected or developed for a specific application should be the result of evaluating six considerations:

(1) characteristics of the machine

(2) input and output

(3) record length

(4) size of control word

(5) natural sequences in data

(6) the associated program

There obviously is not going to be one way which is best for all types of computers. Therefore, the following considerations should be evaluated for each program and a solution selected which incorporates them in the best possible way:

(1) Sort as many items at one time as space will permit.

(2) Reduce the process time per record to a minimum.

(3) Mode of operation must be compatible with input-output operations and should result in a maximum overlapping of read, write and process time.

(4) The merge type sorting program should maintain and utilize sequences which exist in the input file.

(5) The routine should be compact and easily modified.

(6) The internal sort for a generalized program must be able to accept and sort variable length records with any size of control word.

(7) The routine should also be able to sort any number of records that can be packed into the record storage area.

Records can be sorted either (1) by physically moving them around until they are in order, or (2) by forming tables of machine addresses, "tags," which refer to the records in storage. In the second method, only the tags are moved during the sort. The final ordering of the tags indicates the proper sequence of the records. As a general rule, the early internal sort routines shifted entire records around in memory, and the later versions move only the tag references. There was an intermediate period when the control word and record tag were combined and both shifted in the same way that tags are now moved (SORT 53A). The same principles that are used for tape merging may also be used for an internal sort. There are also several other systems which lend themselves to sorting in high speed storage. One of these is called the insertion method or sifting; since it has frequently been used it is described here in some detail.

SIFTING - THE INSERTION METHOD

The insertion method places each record in sequence as soon as it is encountered. This is similar to the way that an individual might sort a hand of playing cards. Starting at one end of a set of records, the first record is assumed to be in order and the second record placed either before or just after the first so that a sequence of two records is set up. The third record is then inserted in its proper relationship to the first and second. The number of comparisons involved per record is not excessively high. The time spent shifting the sorted records each time there is an insertion does get to be relatively high as N becomes large. When compared to a merge type sort the insertion method ceases to be efficient on the 705 as the number of items to be sorted becomes 30 or 40. The number of comparisons per record is usually considered to be $\frac{(G-1)}{2}$, but this can be reduced by extending the programming considerably so that it is approximately $\log_2 G$. Either a binary search to locate the position of a new item or a system of subdividing the record area into several "bases" is effective in minimizing the number of comparisons. The average number of record shifts to permit the insertion of a new item is $\frac{G}{4}$ items. Despite the limitations just mentioned, the insertion method is often used when the input is buffered and consists of single records. In the case of single records, each new record can usually be inserted into the previously ordered sequence while the following record is being read in. Sifting is a particularly effective method on the tape 650. The table look-up feature, the load buffer and the store buffer instructions facilitate the comparing and record shifting.

This type of internal sorting is simple to code and generally effective for sorting small numbers of items. It is quite useful for ordering a few names, part numbers, memory addresses, and other data of this type. A combination of sifting and merging may occasionally be desirable. For example, two or three sets of 25 to 30 items may be sifted and the results merged. The time per record is favorable, but memory space for two programs is required. If the input-output is not buffered or the number of items is > 50, a merge type of sorting should be considered. A merge or distribution sort is preferable to the sift for the 709 because the sift method requires a greater amount of record shifting.

INTERNAL SORTING BY MERGING

The same kinds of merging operations can be performed within the computer as with tapes. Internal sorting by merging uses a series of merges, or a merge network, as shown in Figure 4.

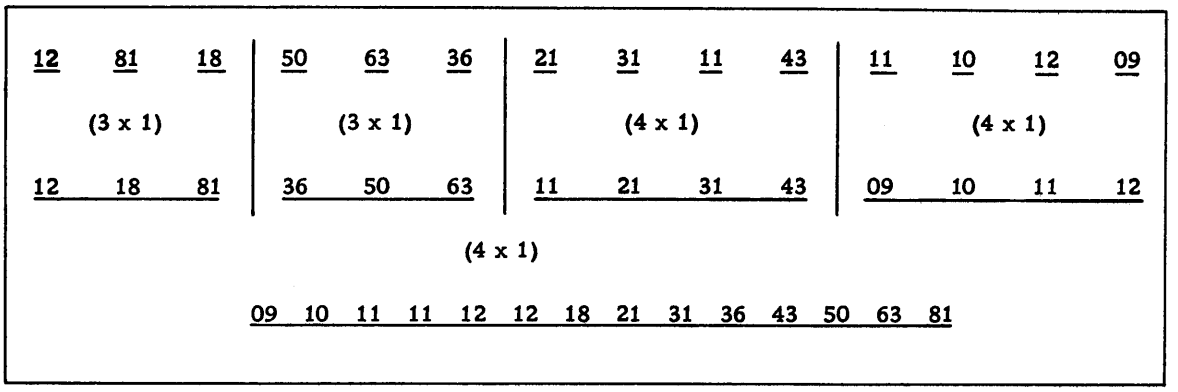


Figure 4

A merge network can be built up by combining different orders of merges in order to produce a final sequence of any given number of records.

For example, a string of 28 records could be sorted by combining two merges like those in Figure 4 and adding a two-way merge to consolidate the two 14-record strings. This type of internal sort is fast because it can be done without address modification or indexing. However, it does lack flexibility and generally will not produce sequences that are as long as those produced by the two-way merge.

The symmetrical merge (two-way merge) has been incorporated in generalized SORTS 53A, 54 and 57. The program used is a two-way merge which checks for step-downs between each item. The two-way merge was used because both the program and storage area requirements are smaller than those required for a three- or four-way merge. The increased number of records that can be sorted at one time using the two-way merge more than compensates for the additional number of internal merging passes that are required. The test for step-down was included because it makes possible (1) saving sort time by eliminating passes whenever sequencing is encountered and (2) handling any number of items (G) at one time. Sort programs which can function only when record groups are set up so that $G = 2^n$ (64, 128, 256 etc.) operate with less process time per record but are too restrictive for generalized usage. An alternative to checking sequence is a version of the merge which establishes and maintains a table that indicates the length and starting position of each sequence within the G records being sorted. This method is applicable to both the 705 and 709. The technique provides the same flexibility as the present version but reduces process time by eliminating one comparison per item per sort pass.

The merge requires more memory space than the sift does. In each method

there must be a record storage area (RSA) of G·L and also an area of G words or addresses where the "tags," representing each of the stored records, are assembled and ordered. The two-way sort also requires a second area for G tags.

Memory Requirements for Two Methods of Sorting

Sift	RSA	G TAGS		
Two-way Sort	RSA	G TAGS Sort 1	G TAGS Sort 2	
Two-way Sort Modifier	RSA	G TAGS Sort 1	G TAGS Sort 2	G ENTRIES Table of sequences

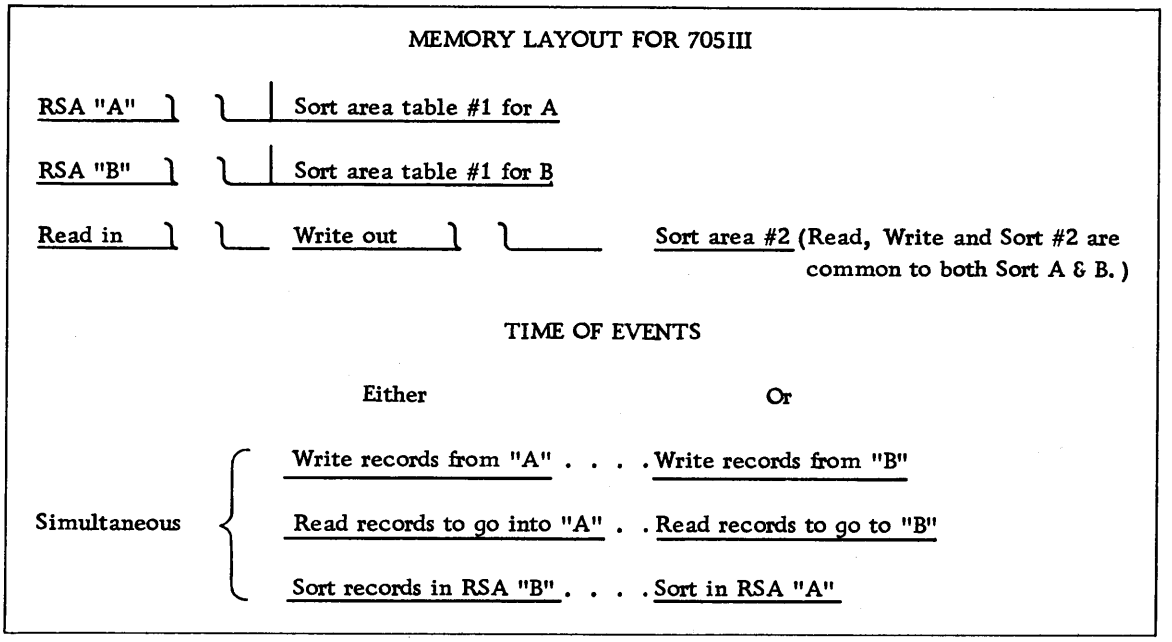
A detailed description of the two-way internal sort is included in the SORT 57 Manual. Two sort areas are used, rather than four as one might expect, because the program alternately sorts in descending and ascending order. This arrangement also insures that the tags will all be in Sort Area 1 as soon as the records are sorted.

The merge has a decided advantage over the sift on unbuffered equipment. The input time is the same for either method so there is no penalty attached to reading all G records into memory before the sort is started. The processing per record is definitely less using the merge sort except for cases where G is very small. It is implied that the sift offers some advantages when using buffered input-output. This would be true except that along with buffers for the 705 and the DS for the 705III and 709 there is also available a memory sufficiently large to make continued use of the merge desirable.

The next round of sorts will divide the additional memory into two sections as shown in Figure 5. Sorted records are being written from area A, input records are being stored in vacated spots of area A while a different set of G records stored in RSA - B is being sorted.

During the second half of the cycle, the reading and writing is into B and the sorting is in A. This arrangement assumes that records are transmitted to a write area and the locations thus vacated are refilled from the read-in area.

The sort program for the 709 will write directly from the read-in area. Therefore, to provide for simultaneous Read, Write, sort and error routines, a third or additional input area is established. The cycle then becomes, "read into C," "Sort B," and "Write from A."



PROPOSED INTERNAL SORT FOR 7051II WHICH OVERLAPS READ, WRITE, AND SORTING

Figure 5

650 INTERNAL SORT

It was mentioned earlier that the table look-up feature of the 650 worked very well with the insertion type of sorting. This machine can locate the place within a table of 48 items where a new CW should be placed and it can also shift the words within the table. Inasmuch as the look-up functions only where one word is concerned, problems are posed when the control word exceeds one word. Optimum conditions are one word CW and a G of 48.

The 650 Tape Sort II program has provisions for two groups of 48 records to be sorted internally and then merged as they are being written out, so that the effective internal sort becomes 96 records. Record length is the limiting factor on this version of the sort. When the length is 10 words or less, 96 records can be sorted. When it is > 10, a maximum of 48 records is sorted at one time. Actually, the two-way merge just mentioned is discontinued at this point. A description of this sort is given in the IBM 650 Tape Sorting Program Sort II Manual (Form 328-0415).

GENERALIZED AND SPECIFIC SORTING PROGRAMS

It becomes more apparent as various sorting problems are considered that the best possible sorting program for a particular file will seldom involve only a single sorting technique. Usually two or three techniques must be properly combined to achieve reasonable efficiency. The manner in which these elements are combined must be determined by the record characteristics, the size and order of the file, and the equipment to be used. Since the method used is based on a solution of equations pertaining to these factors there is good justification for letting the computer determine the sorting method for each file. The sorting programs for the 650, 704, 705 and 709 can be divided into two categories, i.e., generalized and specific.

The generalized sort program can be used to sort a large number of different files without intervention by the user. The program changes which must be made are all accomplished by changing the control card which contains the parameters for each sort application. The specific sort program on the other hand is written for a particular job. It could be used for different applications but hand-coded modifications almost invariably would be required. In the strictest sense the specific sort program is a rare item because different files seldom contain the identical records in exactly the same order and require sorting on the same control field. This is implied in the specific sort.

When a specific sort is written it should make use of all pertinent factors including file size, record arrangement, distribution and machine configuration. The objective is the best possible sorting program for a particular application using a particular machine. Most specific sorting programs which have been written use either the digit or distribution method. Unique tape configurations have accounted for some of the few specific merge sorts which have been written. However, since the saving in running time, if any, is negligible, writing specific merge sort programs for use with tape configurations on which existing generalized programs function would not be feasible. The developing of a specific sorting program by an installation which uses an unusual set of input devices is necessary because even the broadest specifications of a generalized program do not cover such unique configurations. It also might be practicable to write a specific program in cases where the volume of work is large and more equipment is available than is required and utilized by the generalized programs. Any such specific program should be designed to handle the main sorting applications using the equipment peculiar to the installation. Smaller sorting jobs can still be accomplished through use of the generalized programs.

Generalized sorting programs are employed because they can approach the running time of the best specific sorts which use the same number of tapes. The objective of each generalized sort program is a single program that can

be used by a large number of installations to sort a large variety of files. Several installations use fifteen to twenty-five different sorts per month. The cost of writing and testing such a large number of specific programs is very probably beyond any timesaving that could be achieved.

A generalized sort consists of two major parts; the assignment program and the running program. The assignment program does all of the work required to convert the running program into a specific sort for the object file. The parameters on the control card furnish all of the information required to assign tape units as input and output, make memory allocations, and compute machine addresses for the running program. Instructions which are not required in the running program for a particular application are eliminated during assignment time to save processing time during the execution of the main program. The memory space used by the assignment program is also used by the running program for data input, output, and other storage space. The sort consists of three phases. An internal sort is performed in phase 1, all merging but the final pass is accomplished during phase 2, and the last merging pass together with the internal control and checking completes the sorting operation in phase 3. In actual practice, each phase of the sort is accompanied by its corresponding section of the assignment program.

The question of when to generate the object sort is currently a matter of preference determined by operating procedure. The way present programs are written, the phase one running program and its assignment program go into memory at the start of the sort. The assignment is completed in a fraction of a second and the sort is then started. The phase two instructions and assignment program are called into memory and the assignment executed while the tapes are rewinding between phases one and two. The same is true for the third phase. An optional method is being offered where the assignment of all three sections is made at one time. The specific sort that is created in this way is written onto tape as three large records. The sections are called in one at a time when sorting just as the phases come in one at a time when there must be an assignment. The actual running time for the two is the same.

As refinements are added to generalized sorts there will be an advantage in continuing the present method of setting up the sort for each new file just before it is to be run because the program probably will change itself even when everything about a job is unchanged except the file size. A change of only a few records in the file size might permit the assignment program to compute and incorporate a program revision which will save several minutes or even one merge pass. The assignment programs can use the file size, blocking and record description so that the very fastest sort is programmed when the generalized instructions have been made into a specific sort. That is, the blocking will be set up for the sort so that it gives the fastest time with the input and output specified. This may

not be the largest blocking that is possible with the available memory and input/output units. The number of records internally sorted is first made compatible with the blocking and then reduced to a size that is best for the given number of records in the file. The best possible number of records to sort internally (G) is that number which will result in the least total time for the internal sort and the merge passes.

For example,

1. Assume that a file of 39,000 records is to be sorted and available tapes permit a four-way merge. The assignment program first determines that the record length is such that G maximum is 600. (600 records can be sorted internally.) The number of merge passes would be four since 39,000/600 produces 65 sequences. The assignment program would then reduce G from 600 to about 153 because this will reduce the sort time in phase one by about three minutes. The time used for the remainder of the sort will remain the same because 39,000/153 is still less than 256 and therefore in the range of four merge passes.

2. Now assume that a second file contains 39,500 items. If the program generated for "1" above is used again, i.e., without rerunning the assignment section, it will go through five merge passes instead of four (39,500/153 = 258 sequences). By rerunning the assignment program, G can be increased to 155 or more. The sort will remain a four pass sort and the savings achieved by sorting an amount less than G equal to 600 is still maintained.

Sophistications of this type are being incorporated in the newer IBM generalized sort programs. It is anticipated that future programs will be even more powerful because of refined techniques. As the generalized programs become more efficient, the number of specific sorts can be expected to decrease. The following list of specifications for SORT 57 indicates the flexibility that is now being built into sort programs.

(1) Either Model I or II 705 can be used.

(2) TRC tape system must be used. 9 to 12 tapes can be used, at least 4 tapes on each TRC.

(3) Records may be from 6 to 1019 characters in length.

(4) Input may be of single or blocked records.

(5) Output may be of single or blocked records.

(6) 1 to 5 control fields may be used and may be located any place within the record.

(7) Control word may be ≤ 63 characters.

(8) Several options available for use of tape labels.

(9) Restart and checkpoint type correction routines are included.

(10) Phase 2 can be interrupted and resumed at a later time.

(11) The user can add routines or instructions to both phases 1 and 3. The memory space used by the sort can be restricted to leave room for added instructions.

The checks, restart, and interrupt features which are included also require memory space and some process time. Some specific sorts eliminate these thereby saving some time, but this is a questionable procedure. Even the flexibility of programs like SORT 57 is not sufficient to meet every situation. Provisions do exist, however, for making changes to the program with very little effort in comparison to that required to write a specific sort program.

A specific sort is currently being used at a 705 installation which is an excellent example of what can be accomplished with such a program. The program is designed to sort a large (750,000 items) file of 30-character records on a 15-character control word. A 705 Model II with two Tape Record Coordinators using 13 tape units is employed. The sort operates as 2 three-way sorts until two groups of three sequences each are formed and then it performs a final six-way merge. No internal sort is included and no record rearrangement is needed.

The input records are sorted into 306-record sequences and blocked to 34 records during the operation which precedes the sort. The sort requires up to 6 three-way merge passes on each TRC plus the final six-way merge. The time per record as calculated from actual running is about 20% less than the generalized program would require.

Specific sorts frequently do not run much faster than generalized programs. When they do, however, the speed can usually be accounted for by the omission of checking and restart procedures. This is not the case with the specific sort being discussed. The use of hash totals is reduced but the improvement in time is otherwise due to the coding techniques that are used. Essentially these techniques eliminate every possible machine cycle that can be taken from the running program, e.g., the program (1) modifies only one character per instruction; (2) spaces routines and input areas 1000 positions apart; (3) predetermines read-in

addresses so that addresses within the records correspond to preselected characters; and (4) uses certain ASU's because of the special characters that result.

The time for a sorting run is about the same as if the same twelve tape units were used in a six-way sort with the TRC's operating as simple buffers. It is assumed that a three-way sort is process-limited. The specific sort being discussed is more flexible than a six-way sort would be because it runs with either one or two TRC's on the 705. The problem of program modification that is inherent in any specific program is magnified in this case because of the special memory assignments and coding techniques. This highly specialized program is justified by its heavy usage averaging 30 to 35 hours per week.

In summarizing, this program, as compared to current generalized sorting programs, sorts a larger file, utilizes more tape units when operating with both TRC's, reduces the sort time per record and provides for flexibility by being able to operate with only one TRC.

GENERAL CONSIDERATIONS IN SORTING ON THE 650, 705 AND 709

The interaction of the various factors in any sort program makes it difficult to evaluate separately any one single feature in a sort or a suggested change in the program. The items discussed in this section were chosen in the belief that a better understanding of these areas would provide a clearer picture of sorting in general. These topics are discussed only as they apply to sorting on three computers; the 650, 705 and 709. These items do not necessarily constitute a complete set of significant factors but do include topics which are of general interest. The material should be helpful both in using the present programs and in developing new ones.

BLOCK SIZE (BL) Records are blocked for sorting to reduce the tape start time per record and to increase the total number of items that can be written on one reel of tape. However, the efficiency of a sort is not measured by the relative size of the blocked records. There are several other factors which must be considered.

The 650 is limited to a block size of 60 words because of the size of Immediate Access Storage. Indexing does help handle blocked records, but since all reading and writing use the same core storage, blocking adds a significant amount of process time to a program that already involves much process time.

The use of the TRC on the 705 also limits blocking. It establishes a maximum BL of 1020 characters, which does not prove to be a restriction where the records are short and the CW is relatively long. Under this last condition sorting is process-limited, since the tapes must wait for the computer. No running time can be saved by shortening the start time per records, i.e., making larger blocks. The same process-limited condition may also develop on the 705 and 709 when using Data Synchronizers. Therefore, though there may not be a physical limit to the block size when using a DS, a block of 2500-3000 characters may prove to be large enough. When a job is process-limited, using a large block size only serves to increase the number of records per reel of tape.

Blocking on the unbuffered 705 has more ramifications than it does on either the 650 or 709. In an unbuffered operation, the total running time is directly affected by a reduction in tape time. The most significant reduction in tape time is achieved by completely overlapping reading and writing. This saving is so important that every effort must be made to achieve it. Once read and write have been overlapped, additional ways of saving time include reducing start time per record and eliminating merge passes. Therefore BL should be made as large as possible but only after assuring reading while writing.

The maximum block size will vary inversely with the order of merge. Therefore,

the highest possible order of merge does not always provide the optimum sort. It may be preferable to use the next lower order of merge and a larger block size. For example, assume that a 705 has 10 tape units available, therefore either a four- or five-way merge sort may be considered. Both the tape time per record and process time per record are greater for the five-way than for the four-way sort. Therefore, the higher order, or five-way sort, is only more advantageous in those instances where it saves a merge pass.

The following example (Figure 6) shows an interesting result where blocking is involved. When the input and output to an unbuffered sort is blocked, and both blockings are the same size, the sort is fastest when it retains the same input blocking (Bi) as the blocking throughout the sort. Bi and Bo are both given as 6. The maximum the sort can set up is $B = 36$. That is to say that the sort has the capability of reading in groups of ≤ 36, reblocking them for the sort at 36 and then preparing a final output that is ≤ 36. This is a special case and only holds true when:

a. The equipment uses the read while write mode.

b. The file is either small or moderate in size.

This example may not hold when five or six merge passes are required. It also does not apply to sorting techniques which use the Data Synchronizer, since that equipment uses independent (asynchronous) read/write.

Method B saves 4 minutes because read time and write time are completely overlapped throughout phases 1, 2 and 3. If the problem is changed so that Bo is 36 instead of 6, a sort blocking of 36 should be used and the total tape time would be 33.0 minutes. A large number of sorting applications either have large blocks in and out or single input and large blocks as output. In either of these situations the sort blocking (B) should be made as large as possible.

GIVEN:			
	N	File size	60,000
	L	Record length	80
	Bi	Input blocking	6
	Bo	Output blocking	6
	B	Maximum Sort blocking	36
	G	Maximum that can be sorted internally	450

Method A

Let B = 36
then G = 442
and $\log_3 \frac{N}{G}$ = 4 Passes

Tape time:

Phase 1 = $\frac{N}{Bi}(10 + .067\ BiL) + .067\ L\frac{N}{B}(B-Bi)$

= 11.5 min.

Phase 2
3 passes = $3\frac{N}{B}(10 + .067\ BL)$

= 16.2 min.

Phase 3 = Same as Phase 1

= 11.5 min.

Total = 39.2 Min.

Method B

Let B = Bi = 6
then G = 450
$\log_3 \frac{N}{G}$ = 4

Tape Time: (Phases 1, 2 and 3 are all the same.)
5 passes = $5\frac{N}{Bi}(10 + .067\ BL)$

= 35.0

Total = 35.0 Min.

Figure 6

BUFFERING

The core storage in the 650 and buffers such as the 777, or 760 on the 705 allow the user to overlap some processing time with tape time. The greatest benefit from a buffer is realized in those applications where process and tape time are about equal. Buffer transfer time is usually charged as unbuffered process time but there are some exceptions in the case of the TRC. The process time for an unbuffered 705 sort ranges from 50% to 100% of tape time, depending on such factors as record length, blocking and size of the control word. This ratio in itself indicates that buffers are generally useful for sorting.

Buffering such as the TRC provides is generally useful for sorting but it does not reduce the over-all time to the tape time of an unbuffered operation. There are three reasons for this: some extra processing is required because of the instruction necessary in a TRC program; the buffer transfer time must be accounted for; and the use of TRC's increases the tape start time per record. The start time is greater because BL is reduced to $\leq$ 1020 characters and the actual time required to accelerate the tapes is increased during most modes of operation. Therefore, on a per record basis both the individual tape time and process time are increased. In the more favorable cases where two TRC's are used as simple buffers, one in and the other out, sort time is reduced by about 30%.

A sort of long records can be slower on a 705 that is buffered than one which is not. At least five records containing 515 characters each could be blocked if using an unbuffered sort. The tape time would then be 36.5 ms/record. If we assume that process time is 3.5 ms/record the total time becomes 40.0 ms/record. When using the TRC, tape time will either be 47.0 or 49.4 ms depending on the mode of operation and all process time can be overlapped. The total running times are then 40.0 unbuffered, 54.0 one TRC and 47.0 using two TRC's, simply buffered. Notice in Figure 7 that reducing L to 510 characters makes the simply buffered system better than the unbuffered by .3 ms per record. As L decreases and/or process time per record increases the buffered system will become more favorable.

Multi-programming is shown as the fourth type of operation in Figure 7 because this is exactly the kind of operation where it best fits in. The process time is very low in respect to tape time. The effective time per record is 29.3 or 25.0 for this merge pass. Even when coupled with the relatively slow third phase that multi-programming requires it will prove to be the best sort if a total of 6 or more passes is required.

No buffering, simultaneous R/W and blocked records.	B = 5 L = 515	B = 5 L = 510
Tape 10.0 5 x 515 x .067	36.5 per rec.	36.2 per rec.
Process 3.5	3.5 " "	3.5 " "
Total Time	40.0 " "	39.7 " "
One TRC Read/Write Mode		
Tape 17.0 515 x .063	49.4 ms/rec.	40.6 ms/rec.
Buffer TR 4.6 4.6	4.6 "	4.6 "
Computer Process 3.5		
Total	54.0 "	45.2 "
Two TRC's as simple buffers		
Tape (Read Side) 14.6 515 x .063	47.0 ms/rec.	39.4 ms/rec.
Tape (Write) 10.0 515 x .063	-------	-------
Buffer 4.6 4.6	-------	-------
Process 3.5		
Total	47.0 ms/rec.	39.4 ms/rec.
Two TRC's Multi-programmed		
Tape #1 17.0 515 x .063	49.4	40.6
Tape #2 17.0		
Buffer 4.6 4.6 4.6 4.6	9.2	9.4
Process 13.5 13.5		
Total	58.6	50.0
	2	2
	29.3	25.0

COMPARISON OF TAPE TIME ON THE 705

Figure 7

CHECKING AND RESTART

Many methods of checking results are used, but a count of records, a hash total from each record, and a sequence check on the final file seem to be sufficient. There have been instances when the checking built into the generalized sorts has detected intermittent machine trouble and several cases where the sort found that the data was not in proper form. The checking also serves as insurance against program errors. Two sort programs had been used hundreds of times successfully at several installations before the particular circumstances occurred to produce an error in the program's logic which was detected by this checking.

A checkpoint type of restart should be built into a sort program because sorting operations may easily run 20-30 minutes. IBM generalized sort programs take checkpoints at every end of file during phases 1 and 3, and at every end of pass during phase 2. This insures that no reel changing is ever required in order to execute a restart.

DRUMS

The auxiliary drum storage available on the 705 has been used in several specific sorting programs. In selected cases it has made sorting faster but the job has to fit the drum. Usually this means that the file can be divided into sections which fit onto the drum. The drum has not been used in generalized programs because it could only serve as a device to lengthen the sequences created by the internal sort. The time required for access and for read and write is greater than the probable savings. For sorting purposes, additional tape units or buffering is a more desirable addition to a system.

DATA SYNCHRONIZERS

The Data Synchronizer (DS) increases the sorting efficiency of both the 705III and the 709. The DS reduces the time used for buffer transfer, provides for complete independence of read and write and eliminates restrictions on the size of record blocks. Programs operate faster and are easier to write.

Memory requirements are generally larger when using DS's than on either buffered or unbuffered equipment. Part of the additional area is used to compensate for the temporary storage previously provided by buffers and part is needed in case of errors occurring during read or write. The 709 makes use of "scatter write," therefore three input areas per input tape should be used. The first area is used for writing, the second for processing, and the third for input. This method provides both the time and space needed to handle errors which may occur during writing. The 705III requires two input areas per input tape and alternate write areas. During phase one on either machine, memory should be divided so that

read, write and internal sort can all be accomplished simultaneously. When record length is about 120 characters or 20 words, memories of about 8,000 words (709) or 40,000 characters (705) will be required to enable phase 1 to sort the equivalent of four (4) three-way merge passes (G = 81) and overlap as indicated above.

ESTIMATING SEQUENCES

If the internal sort checks for step-downs in a file, or to put it the other way, utilizes the sequences in a file, phase 1 time is dependent on both the number of items and the number of sequences.

When the file is in random order, the average length of a sequence is two or the number of sequences in $\frac{N}{2}$. The number of internal sort passes required is then one less than it would have been if the file were in reverse order. Sequences longer than $\frac{N}{2}$ will tend to reduce the processing time in phase 1. The number of merging passes required by the generalized sorts is dependent on the sequences in the file at the end of phase 1. If a file was initially in random order or worse, the number of passes $P = \log_{mG}\frac{N}{G}$ (m = order of Merge). P will be reduced by at least one if several pairs of G records prove to be in sequence on the output tapes of phase 1. This would be the case if G = 10, and the smallest item in one sequence of 10 records is larger than the largest record in the proceeding sequence of 10. Unexpected sequencing such as this is not due to random occurrences but to some previous ordering of the file. It is easy to save one merge pass by detecting sequencing but very difficult to save two passes unless the file is nearly in order.

INTERRUPT ROUTINE

Some provision should be made whereby the sort can be interrupted before its conclusion and later resumed from the point of interruption. This becomes important whenever one hour or more is required to complete a sort.

An interruption can easily be made at the end of any merge pass but should not be attempted during phases 1 or 3. It can also be done in the last two cases but would probably involve some tape changing. If the interruption is made during merging, a restoration of all of memory and all tapes is the only requirement. The SORT 57 Manual describes one way to accomplish the interrupt routine.

LOCATION OF CONTROL FIELDS

Some sorts have required that the control information be located at the first of each record. Such a limitation is undesirable and should be avoided. Control fields are frequently scattered throughout a record and should be assembled

into one block by the sort. It is faster to form one control word for sorting and then restore the record to its original form at the conclusion of the sort than it is to look at each section of control word in every record, each time one record is compared to another. Whenever the control word consists of one field the latest programs leave the record format unchanged. If the control field is composed of complete but non-consecutive words, it may not be worthwhile to assemble a control word. Even in this case, the required indexing or address modification may become so involved that it would be better to put the words in order during the first pass of the sort. The procedure whereby the control word is assembled should also include a method of shifting the other information within the record into the vacated locations so there is no increase in over-all record length.

The sort for 650 indicates that it may be inadvisable to pack the record where control fields are concerned. Whenever possible use full words for the control word and leave the least significant portion of the last part blank if necessary. The time required by the first and last pass to shift the characters around will be greater than the tape time used in reading the added blank characters during the sort passes. If the file is to be processed daily for any reason, and only sorted quarterly or annually, packing would be advantageous.

MAXIMUM FILE SIZE

The maximum number of records that can be handled by a sort is directly related to the number of tapes used and the method of writing the records. The computer being used also directly affects this maximum. Larger blocks of records can be written with some equipment than with other. This increase will raise the limit on file size correspondingly. More records can be written on one reel if they are blocked to 2900 characters than when BL = 1020. The maximum file size can also be increased by manual intervention, such as changing reels during each pass of the merge, but this also exposes the user and program to a greater probability of human error.

IBM generalized sort programs handle as many records as can be written on those tapes that are used for merging. It develops that this file size is one reel less than the order of the merge. The four-way sort can accommodate three full reels of records, two reels for a three-way sort and one reel for a two-way sort.

The IBM generalized sort programs assume a reel of tape to be 2300 feet long. Blocking used to calculate maximum N is that which is used during the sort. Maximum file size (N) is dependent on:

M - the order of Merge

L - record length

B - blocking

and character density

At 200 characters per inch $N = \frac{5.52\ (M-1) \times 10^6 \text{ records}}{L + 150/B}$

At 534 characters per inch $N = \frac{14.63\ (M-1) \times 10^6}{L + 400/B}$

The programs could be changed to write a few more records but it would require that the present method of utilizing sequences in the file be abandoned. If users were permitted to exceed the sort limits that have been established, the sorts would work except in a few selected instances. The example in Figure 8 illustrates the trap that might develop and result in an unending sort.

Given a three-way sort

> 11 sequences are divided between tape units 0201, 0203 and 0205 at end of pass "P". At the end of the next pass "Q" these have been reduced to the four sequences on 0200, 0202 and 0204 as shown in Figure 8. Two sequences result when these four are merged during pass R but the first one is so long that #2 falls behind it on one tape. They cannot now be merged. Repeating the merge will simply created three more tapes like those now on 0201, 0203 and 0205. The program as written cannot merge sequence #2 with #1 and thereby conclude the sort. The situation can be resolved by moving the last part of sequence #1 from tape 0205 and merging #1 and #2 but this necessitates a separate routine which would not be used otherwise and is therefore wasteful of storage.

Sequences written out during each of four merge passes

P					Q			R			S		
Tape					Tape			Tape			Tape		
0201	#1	#4	#7	#10	0202	#1-3	#10-11	0201	#1-9	EOF	0202	#1-9	EOF
0203	#2	#5	#8	#11	0204	#4-6		0203	#1-9 (cont'd)	EOF	0204	#1-9 (cont'd)	EOF
0205	#3	#6	#9		0206	#7-9		0205	#1-9 (cont'd)	#10-11	0206	#1-9 (cont'd)	#10-11

EXAMPLE OF AN UNENDING SORT THAT CAN RESULT WHEN THE MAXIMUM FILE SIZE IS EXCEEDED

Figure 8

A modification of the present sort technique will be added to the sorting system for the 705III. This greatly increases the capacity of the programs and therefore may save merge time that has heretofore been necessary because a file has been too large for a sort to handle on the given number of tapes. Actually this technique will only save time when there is more than one control field. Briefly the user has the option of forcing the merging phase (phase 2) to go through one extra pass. The file is completely sorted but the records are left in the rearranged form set up for sorting. The user is able to process the maximum N records until they are ordered, set them aside after the extra phase 2 pass and these bring another set of N records up to the same stage of completion. Both ordered files are then fed into the final (phase 3) stage of the sort. The final merge, record rearrangement and everything else is the same in phase three as though the file were N records instead of 2N. This method increases the limits for the sort on the 705III to:

2 reels for the 2 x 2 sort

6 reels for the 3 x 3 sort

12 reels for the 4 x 4 sort

20 reels for the 5 x 5 sort

MEMORY SIZE

Whenever a computer has core storage greater than the minumum required to perform the merge phase, the space can be used in one of several ways:

1. Sorting with some editing or other additional operations added to phases 1 and 3.

2. Sorting a larger number of records with the internal sort.

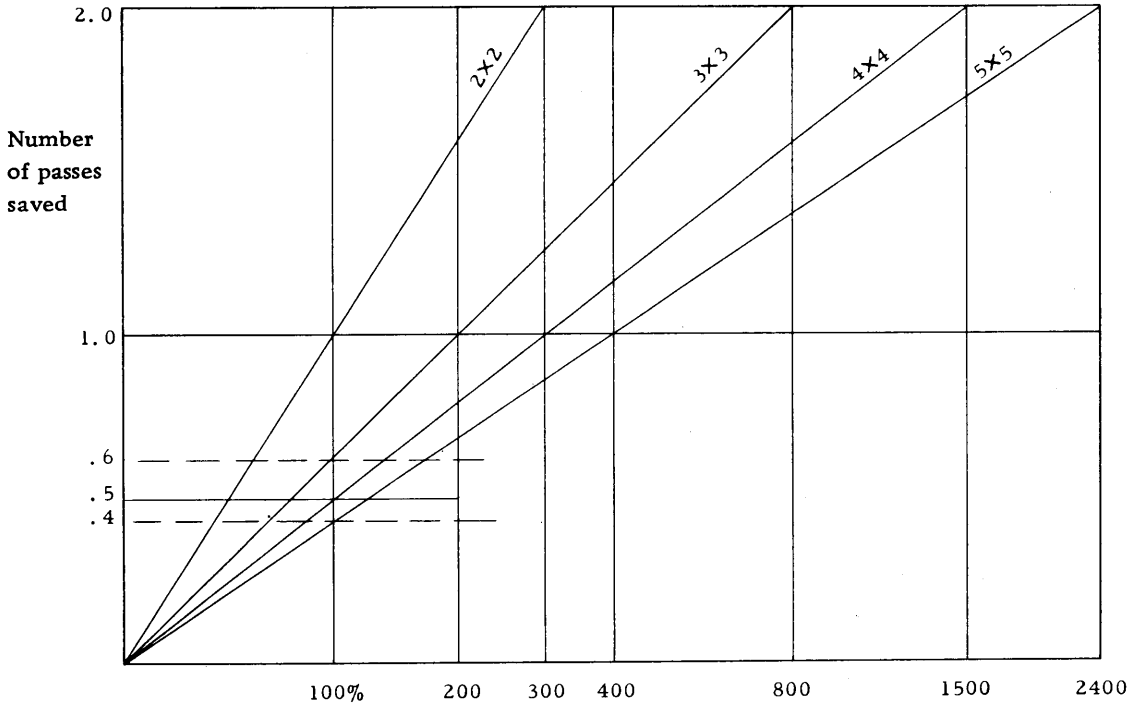
3. Reading and writing the records in larger blocks, assuming the input and output devices will permit this.

4. The order of merge might be increased but this is usually determined by other factors such as number of tapes available and record characteristics.

The first of the above items will usually be the most rewarding. Many times work can be done either as the records first come in or after they have been sorted, which would otherwise require a separate run.

Increasing the number of items internally sorted will only be worthwhile in a

few cases. When the first phase is backed up with a three- or four-way merge the number of items sorted in phase one must be increased 3 or 4 times to insure that one pass will be saved. The graph indicates that the probability of saving a pass by doubling the number sorted (100% increase) is . 65 and . 51 for the three-way and four-way sorts.



CHANCE OF SAVING A MERGE PASS
AS THE NUMBER OF RECORDS SORTED INTERNALLY INCREASES

Figure 9

In a few selected situations a slight increase in G will save one pass. Doubling the size of memory will usually increase G by 100 - 150%. The problem then consists of comparing the product of merge plus rewind time/record, and the probability of saving one pass, with the increased time per record required to sort the additional records in phase 1. For example, consider a sort where N is unknown and a four-way sort is to be used. Merge time is given as 4. 54 ms/record and rewind estimated at . 10 ms/record for a total of 4. 64 ms/record/ merge pass.

Phase 1 can internally sort 560 items @ 28. 08 ms/record or 220 items @ 23. 22 ms/record which is an increase of 155% @ 4. 86 ms/record. There is a probability of . 68 that a merge pass will be saved. This saving is worth . 68 x 4. 64

or 3.06 ms/record, which is < 4.86 the cost of such an increase.

In a second case, merge and rewind time amount to 10.90 ms/record, and the four-way merge is still used. Here either 238 records @ 29.03 ms each or 98 records @ 26.07 ms each can be internally sorted. The increase of 143% costs 2.96 ms/record. The value of such an increase in G is .6 x 10.90 or 6.54 ms/record. In this case where record length is greater there is a clear cut advantage in sorting the larger number during phase 1.

It is generally desirable to sort as many items as possible with the internal sort when other factors are unknown. If file size (N) is known the best time results when the maximum number (G max) is reduced as much as possible without changing the number of merge passes that will follow. This last procedure necessitates that N be known with reasonable accuracy; probably + 2%.

ORDER OF MERGE

There is an understandable tendency to assume that increasing the order of the merge within a sort produces a faster sorting program. However, such an increase will not always save a merging pass. If two programs, such as a three-way and a four-way sort require the same number of passes, the higher order sort will be both slower and more expensive. The expense is due to the additional tape units in use and the extra computer time required.

The table in Figure 10 shows the number of items sequenced by merges which range from a two-way to an eight-way. Four passes are required to sort down 80 sequences using either a three-way or four-way sort. Whenever the number of sequences is > 81, the four-way will always require fewer merging passes.

Number of Passes	Order of Merge						
	2	3	4	5	6	7	8
2	4	9	16	25	36	49	64
3	8	27	64	125	216	343	512
4	16	81	256	625	1296	2401	4096
5	32	243	1024	3125	7776	16807	32768
6	64	729	4096	15625			
7	128	2187	16384				
8	256	6561					
9	512						
10	1024						

TOTAL NUMBER OF ITEMS SEQUENCED BY VARIOUS ORDERS OF MERGE

Figure 10

A sample case will demonstrate the relationship between merging passes and order of merge.

Given: $N = 38,000$ records, $G = 450$
$S = N/G$
$S = 85$ sequences produced by phase 1 of a sort.

A 2-way merge will require 7 passes to complete the sort.

A 3-way merge will require 5 passes to complete the sort.

A 4-way merge will require 4 passes to complete the sort.

A 5-way merge will require 3 passes to complete the sort.

A 6-way merge will require 3 passes to complete the sort.

A 7-way merge will require 3 passes to complete the sort.

An 8-way merge will require 3 passes to complete the sort.

Minimum total running time is the primary objective in a sort, therefore factors other than the number of merge passes should be considered. If the sort is unbuffered, increasing the order of merge and saving one pass will reduce the total sort time. Time can also be saved by using a higher order of merge if the sort is buffered but still a tape-limited job. However, when the records and control word are of such a length as to make the sort process-limited, the total running time will be about the same even in those cases where increasing the order of merge saves a pass. Each pass will take longer with the higher order sort, thus the total running time comes out about the same. The higher order merge does offer the advantage of sorting more records at one time.

The reduction in running time, which may be achieved by increasing the order of merge for tape-limited sorts, is directly related to the reduction in passes that is achieved. The table in Figure 11 shows the approximate relationship which exists between the different orders of merge. Since the number of passes (P_m) is the integer just larger than $\log_m S$, the relationship of P_m to P_{m+1} will not be constant throughout all values of S.

The ratio of the number of passes required when using different orders of merge is based on the following:

P_m = Merge passes required to sort a file using an m-way merge.

R = Ratio of merge passes for two orders of merge $\left(R = \frac{P_{m+1}}{P_m}\right)$

$P_m \approx \log_m S$ and $P_{m+1} \approx \log_{m+1} S$

$$R \approx \frac{\log_{m+1} S}{\log_m S} \approx \frac{\log_{10} m}{\log_{10} m+1}$$

Original Order of Merge	Increased Order of Merge			
	3	4	5	6
2	37%	50%	56%	61%
3		21%	31%	38%
4			14%	22%
5				10%

APPROXIMATE PERCENTAGES OF MERGE PASSES SAVED DURING PHASE 2 IF THE ORDER OF MERGE IS INCREASED

Figure 11

The running time for a sort is the sum of phase 1 and phase 2 (merging) times. The time required by phase 1 will remain the same regardless of the order of merge that follows. Phase 1 may actually require 20-40% of the total sort time, therefore the savings effected by increasing the order of merge will be less than is indicated in Figure 11.

To summarize, a sort which is definitely tape-limited, usually due to a long record length, will be improved by increasing the order of merge; while the sorts which are process-limited because of short records, or a large control word, will probably not show a significant improvement by such a change.

PADDING

Padding is a convenient device which eliminates the necessity of handling partial blocks of records or determining the number of records in each block every time the record is read during sorting. Since records are sorted as all other records, the location of padding in the sorted file is established by the collating sequence. Padding records for the 705 consist of either blanks (bb. . .b) or nines (99. . .9). The 709 sort uses blanks (bb. . .b), nines (99. . .9) or ones (11. . .1). Such records can be eliminated at the end of the sorting program when they are no longer required.

The IBM 705 generalized sorts (SORT 53A, 54 and 57) and the programs being written for the 705III and 709 give the user the option of placing the padding before or after the main file, if such padding is necessary. Nines padding has been recommended if the user has no particular preference. (See SORT 57 Manual, page 13.)

Instructions and process time required to check input length, write short records and merge short records are eliminated. The process time saved is greater than the extra tape time expended in reading padding routines.

A modification of nines padding has been used which has some merit. At the conclusion of the sort, the nines are changed to zeros or blanks as the final output is written. These characters give a sequence check when they are encountered during the primary process runs. This system retains the advantages of nines during the sort and eliminates the need to test for padding during processing.

The following example gives the tape time required to handle padding compared with the process time used to test the blocks if padding was not used. The times are based on 705II speeds.

Given: $N = 32,002$
$L = 100$
$B = 10$
$P = 4$ passes

Method A. Padding is used.

3201 blocks of 10 records each are processed. (Eight 100-character records of padding are added.)
The tape time added to the sort because of the padding records is computed as follows:

$$\frac{4(8 \times 100) \times .067}{1000} = .214 \text{ sec.}$$

Method B. No padding is used.

Instead of padding, a two-position record count is added to each block of ten records. The tape time for added characters is computed as follows:

$$\frac{4(2 \times 3201) \times .067}{1000} = 1.716 \text{ seconds}$$

The process time required to test the count as each record is processed is computed as follows:

$$4 \cdot N\left(10 + \frac{4}{B}\right)\frac{.017}{1000} = 21.7$$

Total 23.4 seconds

Method C. No padding is used.

In order to check the length of each block as it is read in, a minimum program might require the following:

$$\frac{4(19N/B) \times .017}{1000} = 4.13 \text{ seconds}$$

Method "A" is the best over-all, if the equipment does not overlap tape time and process time. With buffering, "A" is still the most favorable method, costing .214 seconds if the run is tape-limited, and nothing if the run is process-limited.

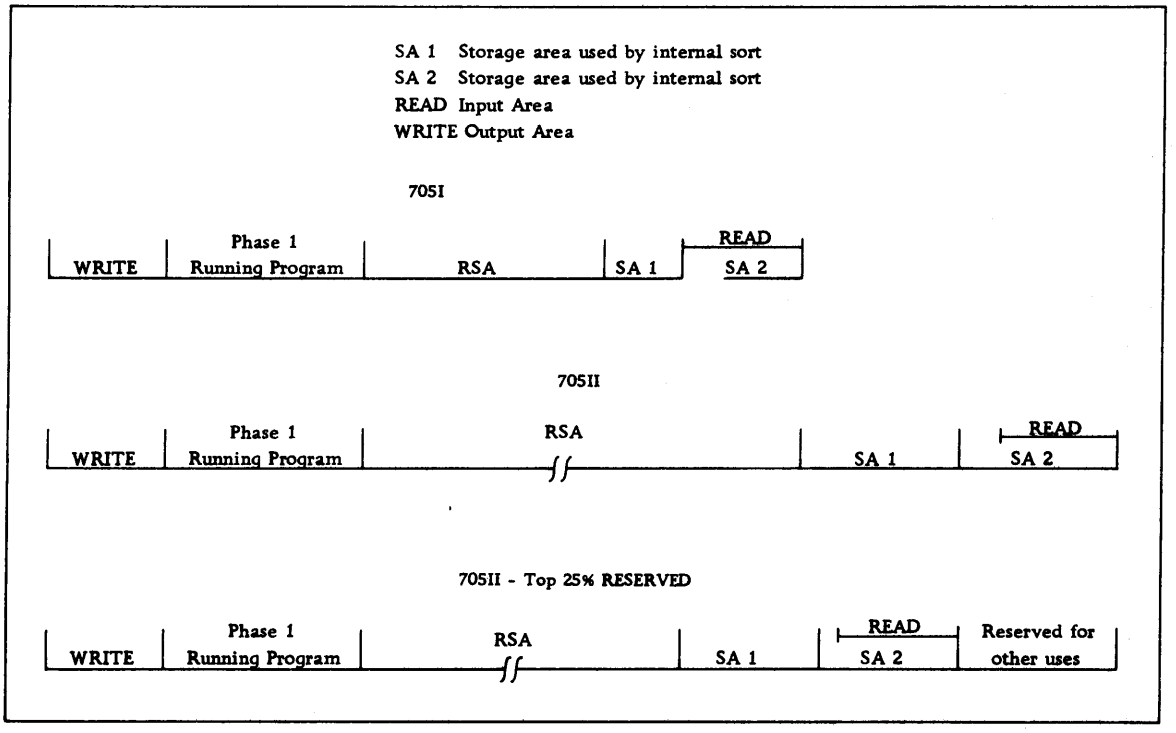
RESERVATION OF MEMORY AND EXIT POINTS

Generalized sorting programs should be written to operate in different amounts of memory. This flexibility is necessary because of the different capacities of different models of the same computer and because the user may wish to withhold part of memory for other programming in conjunction with the sort. Frequently the record length or format must be modified before the sort. The final output record is sometimes changed or the file split between several outputs. Various special tape labeling procedures require some flexibility of memory and its assignment. Flexibility in memory usage by the sort will permit all of these additions and modifications.

The first generalized sorting programs used a fixed amount of memory and were therefore difficult to use when any additional programming was attempted. The problem has been solved in recent sorts by allowing the user to specify effective

memory size as far as the sort is concerned. * Generalized SORTS 54, 57, MERGE 57 and new sorts for the 705III and 709 incorporate this flexibility. The control card for the sort allows the user to specify memory availability for both phases 1 and 3, but they need not be the same. A provision is also made for cutting down on phase 2 memory in SORT 54. The assignment program has the responsibility of insuring that record blocking and memory are compatible throughout the three phases. Usually records in phase 2 would not be changed as to length or number because such factors as blocking, hash totals, sequencing within blocks, and the timing for read and write would probably be disturbed.

The SORT 57 Manual gives a detailed explanation of memory portioning on the basis of usable space. Figure 12 shows how the six areas in phase 1 can be adjusted to use the available space. The areas used in phase 3 can be adjusted in a similar fashion.



ALLOCATION OF MEMORY FOR WORKING AREAS DURING PHASE ONE OF SORT 57

Figure 12

* 650 programs are an exception to this because of the addressing system.

Exit points should be built into a program as it is being written, since the original programmer should be most familiar with his program and therefore be better able to place exits at logical points. He also knows the contents of the various registers and accumulators at various stages of the program. If exits are built into a program, reassembly can be held to a minimum. The time used for such reassembly of a program is of little consequence, but unless great care is used in making changes considerable testing must be made before the reassembled program can be used.

TAPE RECORD COORDINATOR

The effect of the TRC on a 705 Sort is discussed under the topic of Buffers. The Manuals for SORT 57 and MERGE 57 describe the actual utilization of the equipment in generalized sort and merge programs. The TRC speeds up a sort as long as the record length is less than 511 characters.

TAPE TIME vs. PROCESS TIME

The activity percentage in sorting is always 100%, thus, processing and total running time tend to be greater than that for many other data processing jobs. The ratio of process time to tape time extends over a wide range. Most sorting programs tend to require more process-time than tape time. The special features offered by computers such as provision for large control words, blocked records, a high order of merge, and high performance tape all help to create this process-limited condition.

The topic of process time is quite broad. To avoid a general discussion, a set of items that can be considered individually is offered. The brief note with each subject explains the direct affect of that item on process time. In some cases this explanation alone will be sufficient. If more elaboration is required consult the appropriate section of this manual.

Blocking	This tends to make a buffered operation process-limited by reducing the tape time per record.
Buffer Time	Buffer transfer time adds to the process time. It may also add to tape time in the case of TRC using the read-while-write mode.
Checking	Complete checking and restart procedures can add as much as 20% to the process time.
Control Word	Process time is proportional to the size of the CW. This is true in both the sorts for variable word length and fixed word length machines.

Control Fields	The process time in phases 1 and 3 increases with the number of fields.
Data Synchronizer	The units require much less computer time during read or write than TRC buffer transfer time amounted to, but the required memory access cycles must be included in process time.
Internal Sort	A merge type of sort makes phase 1 process-limited. Sifting a small number of items can be accomplished during read time when the input is not blocked to more than two.
Order of Merge	The process time per record usually increases with the order of merge. The three- and four-way merge are very nearly equal thus the four-way over the three-way might be considered an exception.
Record Length	Merge type sorts of short records are process-limited. The record length at which the sort becomes process-limited depends on the size of CW, tape speed and the computer involved.
Start Time	The tape start time per record varies with blocking and the type of tape control used.
Tape Speed	The increased tape speed of the 729III tape unit tends to create a process-limited situation, therefore every effort made to reduce processing would be advantageous.

TESTING

Since the logic in a sort program becomes involved, a well-planned and systematic testing procedure must be followed to insure that the program will handle any and all conditions that may arise. The following list includes the major areas covered by the testing of SORTS 53A, 54 and 57. Tests should be made incorporating the features in this minimum list before any new sorting program is used. Where numbers are used a four-way sort is assumed.

1. Input Data

    a. Single and blocked records.

b. Record length minimum to maximum.

c. Record order: random, in one sequence, in reverse order, and random with long strings of sequenced items included.

d. Input from auxiliary (alternate) input or merge tapes.

e. Some equal control word in data.

f. Files of one record, one block and G records.

2. Control Word

a. Maximum and minimum size.

b. Every legitimate combination of field size, number and arrangement within the record.

c. Test merge section with equal control word in each input file.

d. Insure that control word is restored to the original location within the record by the phase 3 routine.

3. Step-Down Operation

a. Test merge with one, two, and three input files in step-down condition. Use all combinations of step-down and input files.

b. Test for proper merge with step-down occurring in different sequences. Such as file A, B, C, D, then A, C, B, D etc.

c. Intersperse step-down and end-file on the input testing every combination and sequence.

4. End-of-file

a. On the input during phase 1, after one or more files, from both auxiliary and merge tapes.

b. From input to merge phase. All possible sequences, i.e.,
1 - 2 - 3 - 4, 1 - 4 - 2 - 3, 1 - 3 - 4 - 2 etc.

c. On write side during phases 1, 2 and 3.
On checkpoint tape.
On unreadable record tape during phases 1, 2 and 3.

5. Labels

    a. Test all combinations.

    b. Restart with and without labels.

6. Restart and Interrupt

    a. Restart during assignment and running program of phases 1, 2 and 3. This should be both automatic and by a manual STOP, CLEAR MEMORY, and RESTART.

    b. During both first and subsequent reels of input phase 1.

    c. During both first and subsequent reels of output phase 3.

    d. After end-of-file on output phases 1 and 2.

    e. During both odd and even passes of phase 2.

    f. Test interrupt. Remove and restart entire program.

7. Padding: Test using all types and combinations.

8. Error Routines: Force errors to test occurrence while

    a. Reading Program

    b. Reading Data

    c. Writing Data

    d. Writing checkpoint

    e. End-of-file is taking place

9. Operate with various acceptable sizes of memory.

10. RECHECK THE SORT AFTER A PROGRAM CORRECTION OR CHANGE.

The testing can be done most systematically if a record generator is used. Such a program will produce files which contain the exact conditions to be tested and

write the records on designated tapes to insure that the tests are actually made. A small number of items will suffice for each test if the desired conditions are created. The generator must be subject to modification by one or two control cards. It should do the following:

1. Write records on a designated tape to a specific count.

2. Generate single or blocked records of any length.

3. Make records that are either in random order or in sequence. If in sequence, the increment and length of sequences are to be controlled.
   For example:
   1 2 3 ... 10; 7 8 9 ... 16; 13 14 15 ... 22
   or 1 3 5 ... 21; 5 7 9 ... 25; etc.

VARIABLE LENGTH RECORDS

Variable length records present a problem where generalized programs are concerned since blocking such records requires considerable computer time. The solution has been to handle variable length records individually during the merge part of a sort. This is not the ultimate solution but will serve until conventions are adopted and generally used which simplify the problem of ascertaining the length of each record.

One method which could be used would be to assume that variable length records carry the record length as the first field of each record. When variable length records are blocked the block of records would be preceded by one field which gives the number of records within the block. These two devices would help produce faster and more widely applicable programs.

SORT AND MERGE PROGRAMS AVAILABLE

650 Sort II

Two-way sort	- 4 tape units required
Fixed length records	- no blocking
Record length	- 1-60 words
Control word	- 1-5 words
Maximum file	- 1 reel

Phase 2 of Sort II can also be used as a merge program.

705 SORT 53A

Three-way sort	- 7-10 tapes can be used
Fixed length records	- single or blocked
Record length	- 6-500 characters
Control word	- 1-50 characters; 1-5 fields
Maximum file	- 2 reels while blocked for sorting

SORT 54

Three-way sort	- 7-10 tape units
Fixed length records	- single or blocked
Variable length records	- single
Record length	- see SORT 54 Manual
Control word	- 1-63 characters; 1-5 fields
Maximum file	- 2 reels while blocked

SORT 57

Four-way TRC sort	- 9-12 tape units; 2 TRC's
Fixed length records	- single or blocked
Record length	- 6-1019 characters
Control word	- 1-63 characters; 1-5 fields
Maximum file	- 3 reels while blocked for sorting

MERGE 52

Two- three- four- or five-way merge	- 5-10 tape units can be used
Fixed length records	- single or blocked
Variable length records	- single
Record length	- see MERGE 52 Manual
Control word	- 1-50 characters; 1-5 fields
Maximum file	- < 100 reels

MERGE 57

Two- three- or four-way TRC merge	- 6-8 tape units can be used; 1 TRC required
Fixed length records	- single or blocked
Variable length records	- single
Record length	- 10-1019 characters
Control word	- 1-63 characters; 1-5 fields
Maximum file	- < 100 reels each input file

Programs contained in the SHARE (704/709 users' association) library.

NSSRT 1 — National Security Agency
SHARE Distribution #129

704 block sort	- 3 tape units required
Fixed length records	- single or blocked
Record length	- 600 words
Control word	- must be at left end of record

NSMRG 1 — National Security Agency
SHARE Distribution #129

704 two-way merge	- four tape units required
Fixed length records	
Record length	- 600 words
Control word	- must be at left end of record

NSSRT 2 — National Security Agency
SHARE Distribution #427

704 Block Sort	- 4 tape units required
Fixed length records	- binary single or blocked
Record length	- maximum block size 8K 832 words 32K 8000 words
Control word	- must be at left of record. (May be as long as record.)

NSMRG 2 — National Security Agency
SHARE Distribution #427

704 three-way merge — 7 tape units required
This program is expected to be used following NSSRT 2.

GISG — General Electric Phoenix
SHARE Distribution #404

704 Sort Generator
Produces an internal sort and two-way merge:

Fixed length records	- binary or BCD single or blocked

ESTIMATING SORT TIME

The methods and formulas as used herein are those used for IBM generalized programs. The procedures can be used for specific sort programs but might not elicit precise results. Any approximations should indicate what can be expected from another sort.

Some sort estimates can be made directly from tables and graphs which have been published (See SORT 57 Manual) and others will require calculation. It is assumed that records are in random order; therefore, actual sorting time will be less than estimated as the sequences within the file are greater than two.

The following abbreviations are used throughout this section:

B	Sort blocking
B_i	Input blocking
B_o	Output blocking
CW	Length of control word
F	Number of control fields
G	Number of records sorted internally
G^1	Preliminary approximation of G, usually the maximum
L	Record length, adjusted if necessary
M	Order of merge
N	Number of records in the entire file
P	Number of merging passes

$= \text{LOG}_3 \frac{N}{G}$ for SORT 54

$= \text{LOG}_4 \frac{N}{G}$ for SORT 57

P1 Number of merge passes for one internal sort of G records

$= \log_2 \frac{G}{2}$ (actually this is $\log_2$ sequences in G records)

RIA Read in area

RSA Record storage area

T Tape time/record

Ti Internal sort time/record

T1 Total time/record in phase 1 as given in tables for SORT 57

T2 Total time/record in phase 2 as given in tables for SORT 57

T3 Total time/record in phase 3 as given in tables for SORT 57.

Constants used in tables and formulas:

.009 milliseconds character rate - buffer to memory with TRC

.063 milliseconds character rate with TRC

.067 milliseconds character rate with tape control unit

7.5 milliseconds tape start time 729III

10.0 milliseconds tape start time TCU and for write on TRC

14.6 milliseconds tape start time reading via TRC

2300 ft. Amount of usable tape per reel.

650

Tape Sort II

Sorting time can be estimated from the information given in 650 Tape Sort II, Section VII. The time per 1000 records for each phase can be read directly from the curves which are provided.

Notice that Sort II makes the use of phase 1 optional. The sorting in phase 1 is only by the 10 most significant characters of the control word.

Therefore the number of sequences in the G records internally sorted may be one or many more depending on the control word size. It is for this reason that the operating instructions state that phase 1 should be used when previous

experience shows that it is worthwhile.

Sample of estimate for Sort II

Given: N = 14000; L = 12 words (numeric) and CW = 3 words.

From tables

passes required	9-13
phase 1 time	1.53 min./1000 records.
phase 2 time	1.00 min./1000 records.
use A	.14

Use 11 as the probable number of merge passes.

Then:

$$\text{Phase 1} = \frac{14,000}{1000} \times 1.53 = 21.2 \text{ min.}$$

$$\text{Phase 2} = \frac{14,000}{1000} \times 11\ (1.00 + .14) = 176.0 \text{ min.}$$

$$\text{rewind} = 1.2\ (11 + 1) = 14.4 \text{ min.}$$

Total time = 211.6 min.

705

SORT 54

Since SORT 54 is an unbuffered sort, total time per record is the sum of process time and input-output times per record. The following formulas can be used as a guide in estimating time required for a sort. Total sorting time in minutes is equal to the sum of the following results multiplied by N/60,000.

Phase 1 - Time in Milliseconds/Record

$$\text{Process: } .017 \left[P1^{*} \left(80 + 4CW + \frac{194}{G}\right) + \frac{3L}{5} + 166 + \frac{41}{B} + \frac{340-L}{5B_i} \right.$$

$$\left. + \frac{93 + 2CW}{G} + \frac{60,000}{N} + (6CW + 10F + 31)^{**} \right]$$

*P1, the average number of passes per internal sort in phase 1, equals Log_2 G/2. This assumes that input records are in random order and that their average length of sequence is two.

**This factor is included only if F, the number of control fields, is greater than one.

Tape: $$\frac{10 + .067BL}{B} + \left(1 - \frac{B_i}{B}\right) \frac{10 + .067B_iL}{B_i}$$

Phase 2 - Time in Milliseconds/Record

Process: $$P2^{***} \left[.017 \left(\frac{L}{5} + 3CW + 100 + \frac{100 + CW}{B} + \frac{22,000}{N} \right) \right]$$

Tape: $$P2 \left(\frac{10 + .067BL}{B} + \frac{.2BL}{N} \right) + \frac{72,000\ (P2 + 1)^{****}}{N}$$

Phase 3 - Time in Milliseconds/Record

Process: $$.017 \left[\frac{2L}{5} + 6CW + 175 + \frac{60}{B_o} + \frac{10B_o + 45}{B} + \frac{31,000}{N} \right.$$

$$\left. + (4CW + 12F - 4)^{**} \right]$$

Tape: $$\frac{10 + .067BL}{B} + \left(1 - \frac{B_o}{B}\right) \frac{10 + .067B_oL}{B_o} + \frac{.2BL}{N}$$

SORT 57

The tables published in the SORT 57 Manual provide the best method of estimating running time for this program. If either B_i or B_o are < B the additional read or write time can be computed and this factor added into the total time given by the calculations from the tables.

Additional read time in Phase 1 when $B_i < B$

$$\left(1 - \frac{B_i}{B}\right) \left(\frac{14.6}{B_i} + .063L\right) \text{ milliseconds/record}$$

Additional read time in Phase 3 when $B_o < B$

The time added because of a small block factor is dependent on the ratio of tape time to process time. Therefore calculate the write time/record for B_o

** This factor is included only if F, the number of control fields, is greater than one.

*** P2, the number of merging passes in phase 2, equals $(\text{Log}_3\ N/G) - 1$.

**** If extra tape routine is used, substitute $\frac{72,000}{N}$.

$$\text{write time/record} = \frac{10 + .063LB_o}{B_o}$$

Use this new figure instead of T3 in those cases where it is greater than the T3 given in the tables.

The SORT 57 tables are based on the following:

$$B = \frac{1020}{L}$$

$$I_1 = \frac{14,000 - 2CW}{L + 10} \qquad I_2 = \frac{34,000 - 2CW}{L + 10} \qquad \begin{pmatrix} I_1 \text{ 705 Model I} \\ I_2 \text{ 705 Model II} \end{pmatrix}$$

$$R = 14,000 - 2CW - GL - 10G$$

$$RIA = 5G + R \quad \text{when } 5G < B_iL$$

$$P1 = \log_2 G/2$$

$$P = \log_4 N/G$$

$$T = \frac{14.6 + .067\ BL}{B} \qquad \text{*read time/record (milliseconds)}$$

$$t = \frac{14.6 + .067\ BL - .018BL}{B} \qquad \text{available process time}$$

$$T_{int} = \frac{.017}{G} \text{ P1 or P2 } G(4CW + 65) + 4CW + 175 + 50G \qquad \text{internal sort time}$$

$$T1 = .017\left(\frac{3L}{5} + 3CW + 150\right) \qquad \text{Phase 1 process}$$

$$T2 = \frac{.017}{B} B\left(\frac{L}{5} + 3CW + 130\right) + 7CW + 150 \qquad \text{Phase 2 process}$$

$$T3 = \frac{.017}{B} B\left(\frac{2L}{5} + 4CW + 120\right) + 5CW + 120 \qquad \text{Phase 3 process}$$

* The character cycle of .067 instead of .063 was used at the time these were formulated.

APPENDIX

List of Terms and Abbreviations

Adjusted Record Length (La)

The length of a record while it is being sorted as distinguished from its length prior to that time.

In most sorting jobs, the adjusted record length is the same as the original length. There are, however, several instances when records must have a different length while they are being sorted than they had when first read into the computer. This difference must be accounted for in the parameters supplied to a generalized sorting program, and may result from an edit performed on the records after reading, and prior to sorting, or it may be due to the characteristics of the computer. The 650, 704 and 709 must deal with full-word records, and the 705 must have records that are some multiple of five characters if it is to make efficient use of the data transmission feature. Therefore, occasionally records will be lengthened by the program, the increase being as much as is needed to complete a group of five characters on the 705 and one word on the 704.

Assemble Control Word

The machine operations performed by the generalized sorting programs which bring together the fields which comprise the control data. Also called "Pulling the Control Word."

Assignment Program

The set of instructions by which a generalized program modifies and completes its own set of instructions so that it can perform one specific sort.

Binary Coded Decimal (BCD)

A system of representing alphanumerical characters by a combination of six binary positions. A seventh position is used for checking purposes. Information is recorded on tape in one of two forms, i.e., pure binary or binary coded decimal. The 650 and 705 always use the BCD mode. The 704 and 709 can read and write in pure binary or BCD; therefore, the type of input and output must be specified when using the latter computers. When reading in the BCD mode on the 704 and 709, six characters of six bits each make up one word. The double digit representation of alphabetic characters in the 650 is a special way of handling BCD characters in a numerical machine.

Blocking (B)	The grouping of two or more records on tape to create one long record. Blocking increases the number of data records which can be written on a tape and also reduces the tape start time per data record by reducing the number of record gaps.
Block Length (BL)	The total number of words or characters contained in one block of records.
Character	A digit, letter or special symbol.
Checkpoint (CP)	A reference point at which error-free operation of the program has been verified and to which the program may return for restart in the event of subsequent failure. Checkpoint also refers to that routine in the program which writes the checkpoint record.
Collating Sequence	The relative order of precedence which a computer assigns to the numbers, letters, and special characters. The 702 and 705 are designed to maintain the sequence established by the IBM 89 Alphabetic Collator. The Read Alphabetic instruction on the 650 converts numerical, alphabetic and special characters into two-digit numbers whose relative values correspond to the same sequence. The 709 generalized sorting program will offer this same collating sequence at the programmer's option. This sequence will be maintained in the 709 sorting program by converting each BCD character in the control word to a new six-bit number which does have the correct position relative to all other characters. At the end of the sort it is restored to the original BCD representation.
Control Card	The card which contains the parameters required to set up a generalized program for one particular application. The contents of such a card depend on the type of generalized program. In a simple case, the control card might specify only the location of the input file. The control card for a generalized sort program must specify as a minimum the record length, control field size and location, and the number and addresses of reels of the input files. Usually more information is required; some is used to make the sort

program more efficient and other information is used to cross-check the control card itself.

Control Field — A continuous group of characters within a record which form part or all of a control word.

Control Word (CW) — That part of a record made up of selected characters or fields of characters upon which the record is sorted. It is also sometimes referred to as the "key." The abbreviation "CW" signifies the length of the control word when used in a sorting formula. (In the 705III and 709, the term "control word" also pertains to information used in the execution of reading and writing instructions with a Data Synchronizer.)

Convert Control Word — On the 709, to replace each six-bit binary coded decimal character in the control word by the six binary bit representation which indicates the relative position of that character in the collating sequence used by the IBM 89 Alphabetic Collator.

Digit — A number from zero to nine in the decimal system represented by four bits of a six-bit 704, 705 and 709 BCD character and by five or seven bits on the 650.

Fixed Length Records — Records comprising a file in which every record is the same length.

Generalized Program — A program which is designed to process a large range of specific jobs within a given type of application and which has the facility of computing instructions for itself so that it can perform one particular job.

The difference between a generalized program, as it is presently conceived, and a "generator" is not as clearly drawn as it once was. The present generalized programs have provisions to add and/or delete sections of instructions as they are required and move areas within memory so that the running program is both compact and efficient. The running program is therefore similar to that which would have resulted if the sorting program has been compiled by a generator.

Grouping (G)	The number of records ordered in memory by the internal sort. Also called Records Internally Sorted (RIS).
Hash Total	A total of data made for auditing or control purposes which would not ordinarily be added together, e.g., summing a list of parts numbers. In tape sorting applications, such hash totals are reconciled at the end of each pass with the hash total from the preceding pass. This provides a check for machine or program failures and may detect the presence of illegal characters in data.
Illegal Characters	Bit combinations which are not acceptable to the computer or to a given program. Most programs are designed to sort records which are comprised of those characters normally used as data. Thus, tape marks, group marks, and record marks within a record are to be avoided on the 705.
Index	Used in some 650 sorting material to mean "control fields." This usage is not recommended.
Indexing	Usually employed to refer to a method of address modification.
Initialization	Resetting counters, switches, and instruction addresses at specified times in a program. This process is not to be confused with the assignment program which is performed only once, and is always executed before the running program has been started.
Internal Sort	Process whereby several records are stored in memory while their proper order is established according to their control words.
Interrupt Feature	A provision built into a program which enables the operator to stop the program while it is running and remove it from the computer. The program can be put on the computer again at a later time to continue processing from that point at which the program was halted.
Key	A term synonymous with "control word." It is also sometimes used to describe a set of memory addresses which refer to the location of records.

Location of Control Field	The relative position of the right-hand character of the control field with reference to the first character of the record, which is considered as character "1."
Merge	The process of combining several sorted sequences so that they form one sequence. The regular merge programs combine several files which are each completely in order into one new file, also in sequence. The function of the merging phase within a sort program is to combine several ordered strings of records into one longer sequence. These ordered strings of records are obtained by bringing in one sequence from each input tape. The merged sequence is then written onto one of the output tapes.
Order of Merge	The number of files which can be combined into a consolidated file during a merging operation or during the merging phase of a sort.
Padding	One or more records, consisting only of arbitrary characters, added to a file to fill out a partial block of records which would otherwise contain fewer records than every other block. The addition of padding increases the number of records in the file to a multiple of the blocking factor.
Parameter	A specification presented as input to the assignment program of a generalized sort or merge in order to set up the generalized program for a specific job.
Pass	A complete cycle of reading, processing, and writing an entire tape file. A sort program usually requires one pass for the internal sort and several merging passes.
Phase	A logical division in a sorting program wherein a specific portion of the sorting operation is accomplished. This division corresponds to those largely independent parts of the sorting program and is made to save memory space and thereby develop a more efficient program.
Process-limited	The operating condition of a computer when processing time exceeds tape time. This term only applies to equipment which provides for overlapping, or simultaneous input/output operations and processing. The opposite condition is tape-limited.

Pulling Control Word	See "Assemble Control Word."
Record Count	The number of items or records in a file.
Records Internally Sorted (RIS also G)	See "Grouping."
Restart	The return to a previous point in the program to begin processing again. This previous point may be the beginning of the program or it may be a checkpoint. "Restart" also refers to that routine in the program which accomplishes the return.
Record Storage Area (RSA)	That part of memory wherein records are located during the internal sort. This area is used in those internal sort systems which move only references to records and not the actual records.
Restore Record	The process of restoring a record to the input format prior to writing it on tape during phase 3. Restoring the record is the reversal of the process involved in assembling the control word at the beginning of the sort. See "Assemble Control Word."
Running Program	A generalized program which has been set up for a particular job by the assignment program with its control card (parameters).
Sequence	A string or group of items either in ascending or descending order. Ascending order is implied unless descending order is specified. The length of each sequence can be one or more records. It is customary to allow breaks in sequence to occur only between blocks of records. One sequence may, however, include several blocks of tape records.
Sequence Break	Condition when a record has a lower-valued (in the collating sequence) control word than the previous record. The preceding sequence is concluded and a new one must be started. When sorting in reverse order, high to low, a sequence break is just the opposite from the condition described above. In this case, sequence break or "step-down" is the condition which arises when the control word of a new record has a higher value than the control word of the preceding record.

Sifting	A method of performing an internal sort; also called the insertion method.
Sort	To place a file of records in order according to some designated control word data.
Step-Down	See "Sequence Break."
String	See "Sequence." The term "sequence" is preferred.
Tag	This term has different connotations which vary with computer and context. In the 704 and 709, the tag is that part of the instruction which specifies the index register to be used in the execution of the instruction. In assemblies and compilers the term is used to denote the indicator of, or to make reference to, the location in the program of specific instructions, routines or data. As pertains to sorting, the tag is the address of specific records in the record storage area.
Tape Label	A record, usually at the beginning of the tape, ending of the tape, or both, designated for purposes of identification and control. Tape labels have a large number of functions, among which may be some or all of the following: The tape label identifies the input records as belonging to the desired file; confirms that the tape reels put on tape units for output may be used as output tapes; contains the purge data; checks, or enables to check, the switch settings and manual operations of the console operator, etc. Labels at the end of a tape usually contain record count, control totals and end-of-job notations.
Tape-limited	The operating condition of a computer when tape input and output time exceeds processing time. This term only applies to equipment which provides for overlapping, or simultaneous input/output operations and processing. The opposite condition is process-limited.
Variable Length Records	Records comprising a file in which the number of characters in each record varies.

Printed in U.S.A. F28-8001

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York