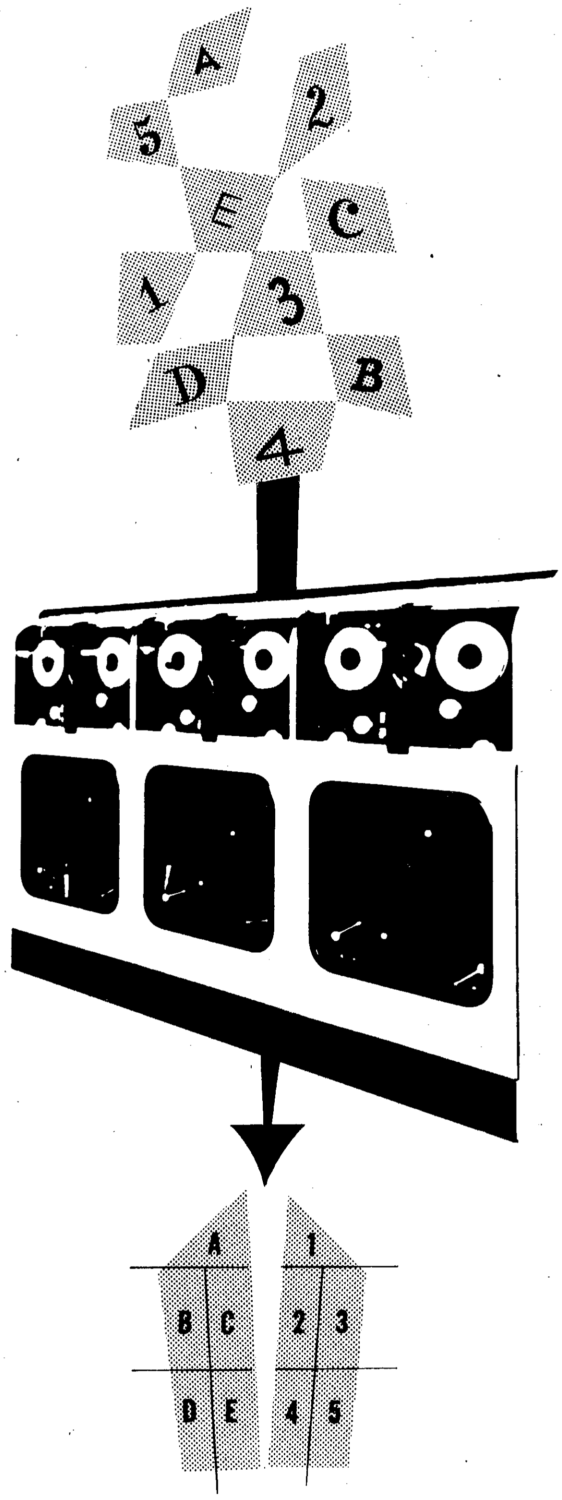


sorting methods for univac system

Copyright 1954
Remington Rand INC.

INTRODUCTION

A characteristic of most phases of commercial or business data-processing is the necessity of assembling, with a Master Record File, one or more files containing additional information about the items before their processing can begin. Each item of both the Master File and the subsidiary files is usually identified by means of a key contained within the item. For example, payroll record items are identified by means of a badge number, inventory records by a stock number. The assembly process then consists of selecting an item from the Master File and then selecting, from the subsidiary files, items with a matching key.

The central problem in such matching operations is the cost at which the information for a given item can be selected from the subsidiary files. The speed at which this selection can be performed will affect this cost. The longer the selection takes, the more equipment and operative personnel are necessary to accomplish it within the time available.

In investigating the speed of selecting an item, let us define the term *access time* as that time necessary to select a given item when that item is one of many items in a file. For our purposes, the given item may be chosen in two ways, with each manner of choice giving a different access time. *Random access time* is the time required to select an item designated in random fashion. *Sequential access time* is the time required to select item k + 1 after having already selected item k.

Fig. 1 is a table giving the average random and sequential access times for the internal memories of typical computers now in common use. The item size influences to a large degree the access times. An item of 120 digits was chosen as a common size. This is equivalent to 400 binary digits or bits.

MEMORY TYPE	TYPICAL COMPUTER	COST PER BIT STORED	ACCESS TIME IN MILLISECONDS	
			RANDOM	SEQUENTIAL
RELAY	**HARVARD: MARK II**	**$15.**	**240.**	**240.**
MAGNETIC DRUM	**ERA 1101**	**.15**	**17.**	**17.**
DELAY LINE	**UNIVAC**	**1.75**	**1.**	**1.**
CATHODE-RAY TUBE	**ERA 1103**	**4.00**	**.08**	**.08**

Fig. 1 Common Types of computer storage

The random and sequential access times shown in Fig. 1 are essentially the same, and with the possible exception of the relay-memories are extremely fast. Unfortunately the files of most commercial applications are of very large size. The Master Employee Record File for a 10,000 man payroll might contain 12,000,000 bits of information and this is a relatively small file compared to the million accounts in the master file of a public utility company. Yet the largest fast internal memory system built stores only 2,750,000 bits at present. Again, the cost of such storage is excessive.

In contrast to the above memory devices, the cost of storing large files on magnetic tape is well within the realm of economic feasibility. Considering UNIVAC tape, which has a re-use factor of about 1000, the cost of storing 10,000,000 bits is only $20 per year, assuming daily reference to every bit. The random access time of tape files depends upon the number of items stored, but even for 12,000 ten-word (120 digits) items (one tape reel) this time required to select one item is 1-3/4 minutes. The sequential access time for ten-word items recorded on magnetic tape is only 20 milliseconds.

In commercial applications involving assembly of information from several files, the initial order of one or more of these files is not controllable by the processor. An example of this is the receipt, by mail, of payments to be credited to an appropriate account--the checks never arrive in account number sequence.

To take full advantage of the large and inexpensive storage of magnetic tape, the items recorded on it must be arranged in a definite sequential pattern in order to use the short sequential access time in selecting an item from the tape. The usual sequential ordering pattern is an ascending sequence. That is, the item with key k_1 is recorded on tape first, then items with keys k_2, k_3, ... in that order, where $k_1 \leq k_2 \leq k_3 \ldots$.

The term sorting as used in this chapter refers specifically to that process which takes a file of items randomly arranged (according to a key within each item) and rearranges them in an ascending sequence according to this same item key. Several methods of sorting on UNIVAC are briefly discussed in the following sections.

DIGITAL SORTING

Digital sorting is the common means of producing ascending item series with punched card machines. It can be modified for UNIVAC in the following manner.

Assume that we have a random sequence of items recorded on tape. Each item has a two digit numerical key, by which the items are to be arranged in an ascending sequence. In Fig. 2a, a random set of items is shown on Uniservo 1. For simplicity in drawing, only

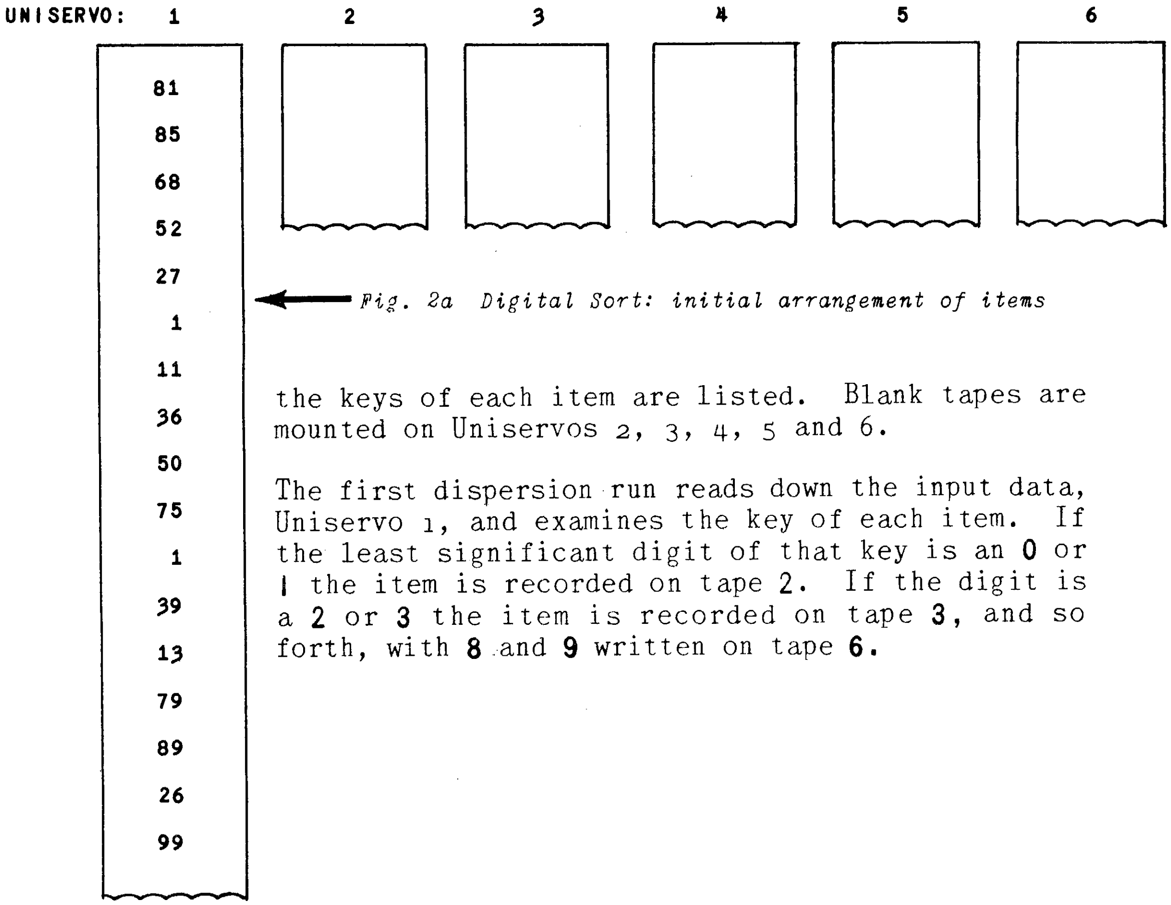


Fig. 2a Digital Sort: initial arrangement of items

the keys of each item are listed. Blank tapes are mounted on Uniservos 2, 3, 4, 5 and 6.

The first dispersion run reads down the input data, Uniservo 1, and examines the key of each item. If the least significant digit of that key is an **0** or **1** the item is recorded on tape **2**. If the digit is a **2** or **3** the item is recorded on tape **3**, and so forth, with **8** and **9** written on tape **6**.

The resulting item arrangement is shown in Fig. 2b. Tape **1** is now shown as blank, indicating that it is rewound and may be used to hold further information. This convention holds throughout the discussions in this section.

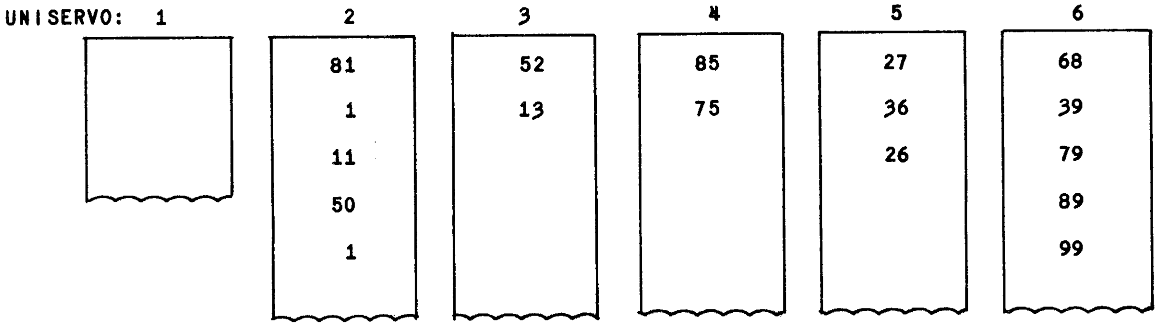


Fig. 2b First dispersion run, first column-digit

The next step is called the second dispersion run. Tape **6** is read backwards, and each item on that tape is examined again. If the least significant digit of the item's key is a **9** the item is recorded on tape **5**, following the **6** and **7** items already recorded. If the key is an **8** the item is recorded on tape **4**, following the **4** and **5** items. A line is shown between the two different sets of items in Fig. 2c, although the routine actually records a sentinel block as a marker. Fig. 2c shows the final item arrangement.

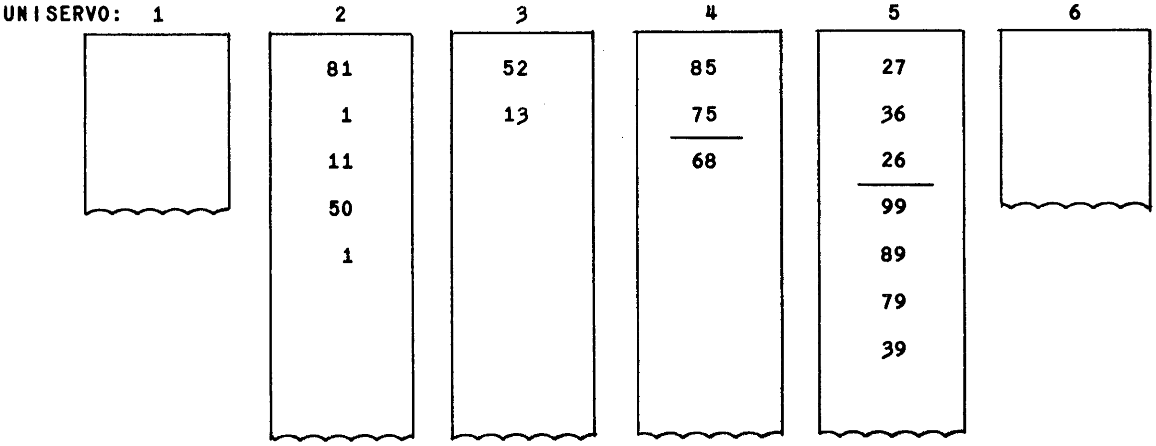


Fig. 2c Second dispersion run, first column-digit

The third step is the collection run. Tape **5** is read backwards. Each **9** item is picked up and recorded on tape **1**. The tape is not read beyond the first **9** item recorded during the second dispersion run. After the **9** items have been recorded on tape **1**, the **8** items are picked up from tape **4**, again reading backwards, and written on tape **1**. The item arrangements after the collection run is depicted in Fig. 2d.

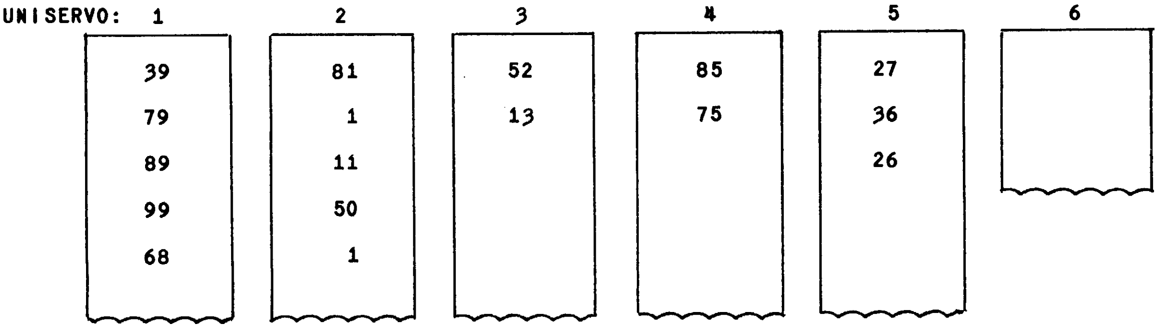


Fig. 2d Collection of **8** *and* **9** *items*

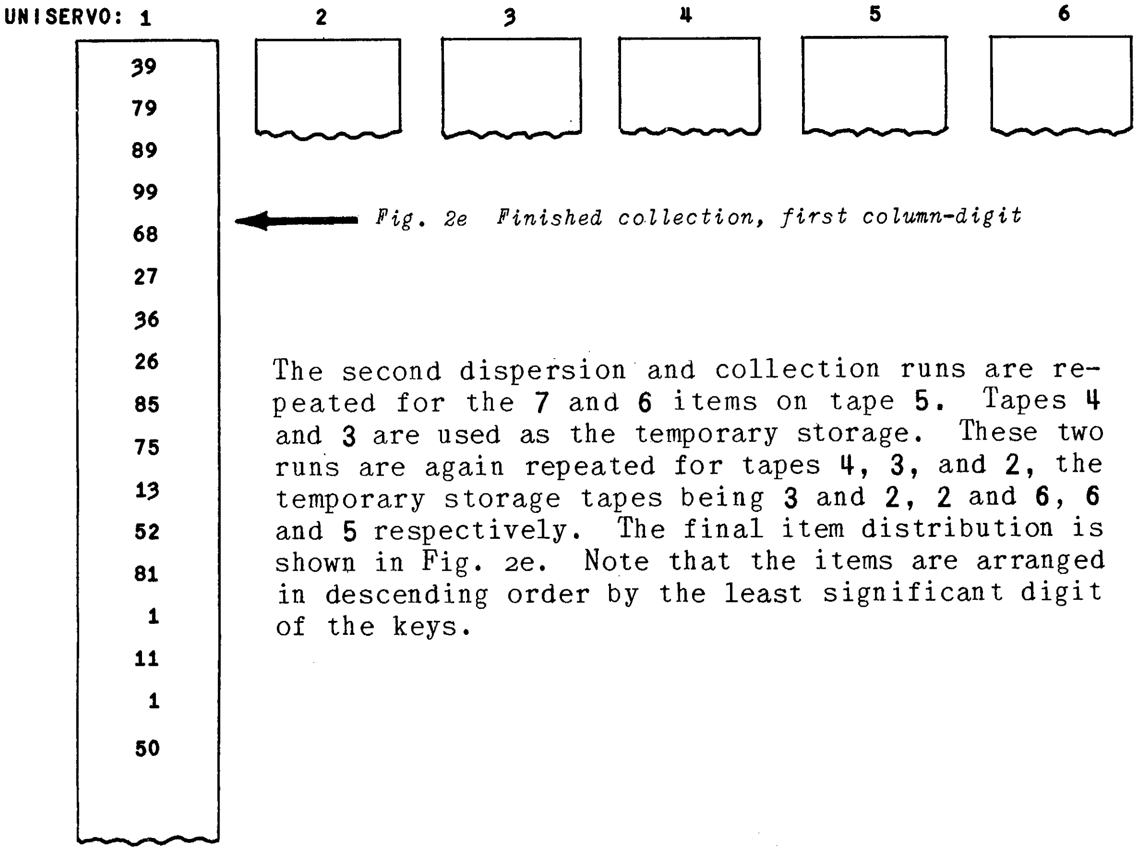


Fig. 2e Finished collection, first column-digit

The second dispersion and collection runs are repeated for the 7 and 6 items on tape 5. Tapes 4 and 3 are used as the temporary storage. These two runs are again repeated for tapes 4, 3, and 2, the temporary storage tapes being 3 and 2, 2 and 6, 6 and 5 respectively. The final item distribution is shown in Fig. 2e. Note that the items are arranged in descending order by the least significant digit of the keys.

Tape I is now read backwards repeating the original first dispersion run; this time, however, the digit in the second column of the key is examined. The item arrangement resulting is shown in Fig. 2f.

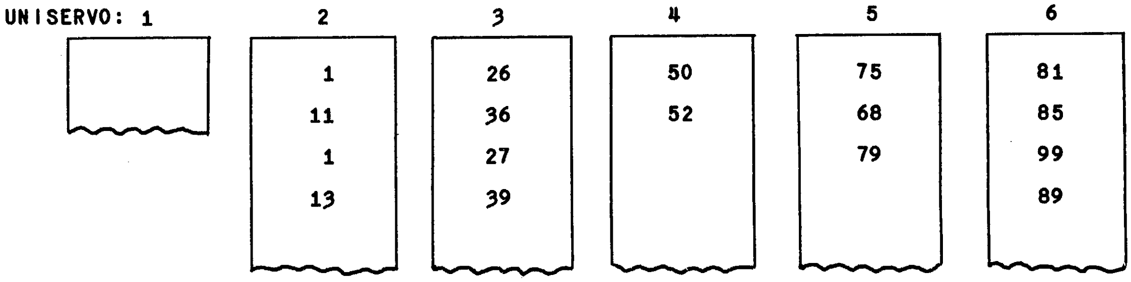


Fig. 2f First dispersion, second column-digit

Next, the second dispersion and collection run is repeated, starting this time with the 0 and I items on tape 2. Fig. 2g shows the items after this collection run. The second dispersion and col-

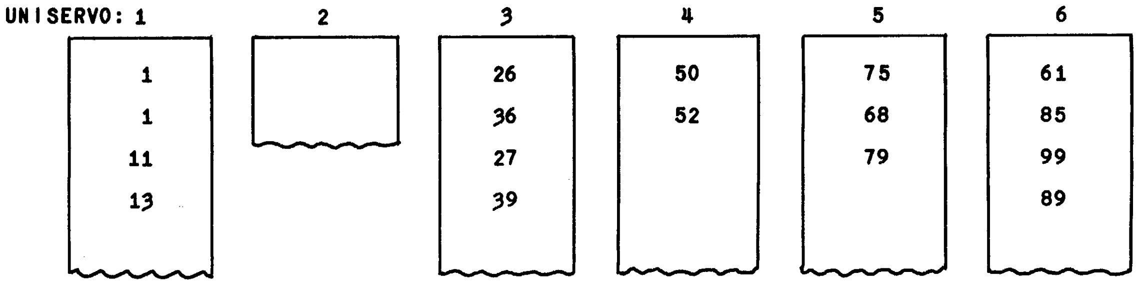


Fig. 2g Collection of 0 *and* 1 *items*

lections runs are repeated for tapes 3, 4, 5, and 6. The final disposition is shown in Fig. 2h. The initially random arrangement has now been transformed into the required ascending series of items.

The digital sort proceeds in exactly this fashion for each additional digit position tobe sorted, with the following exception: if the number of digits in the key is even (2,4,6...) the first collection run starts with the 9 items, the first collection on second column with the 0's, the first collection on the third column of the key starts with the 9's, and so forth; while if the key comprises an odd number of columns, the first collection is the 0's, the second column starts with the 9's and so forth. This ensures that the last collection run produces an ascending series rather than a descending one.

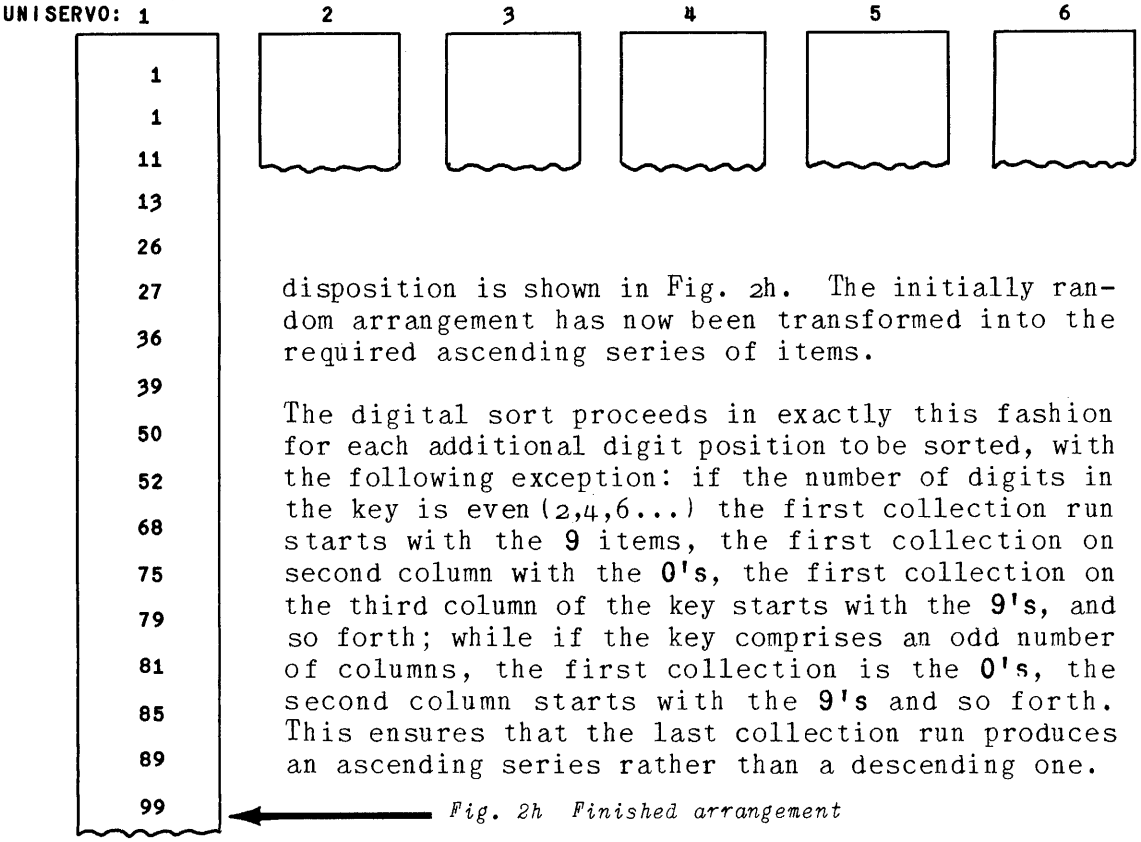


Fig. 2h Finished arrangement

The following table lists the time required per digit of the key, to sort a full tape of 2000 blocks, each containing sixty 12-digit words, using the digital sorting method just described. Three typical item sizes are listed.

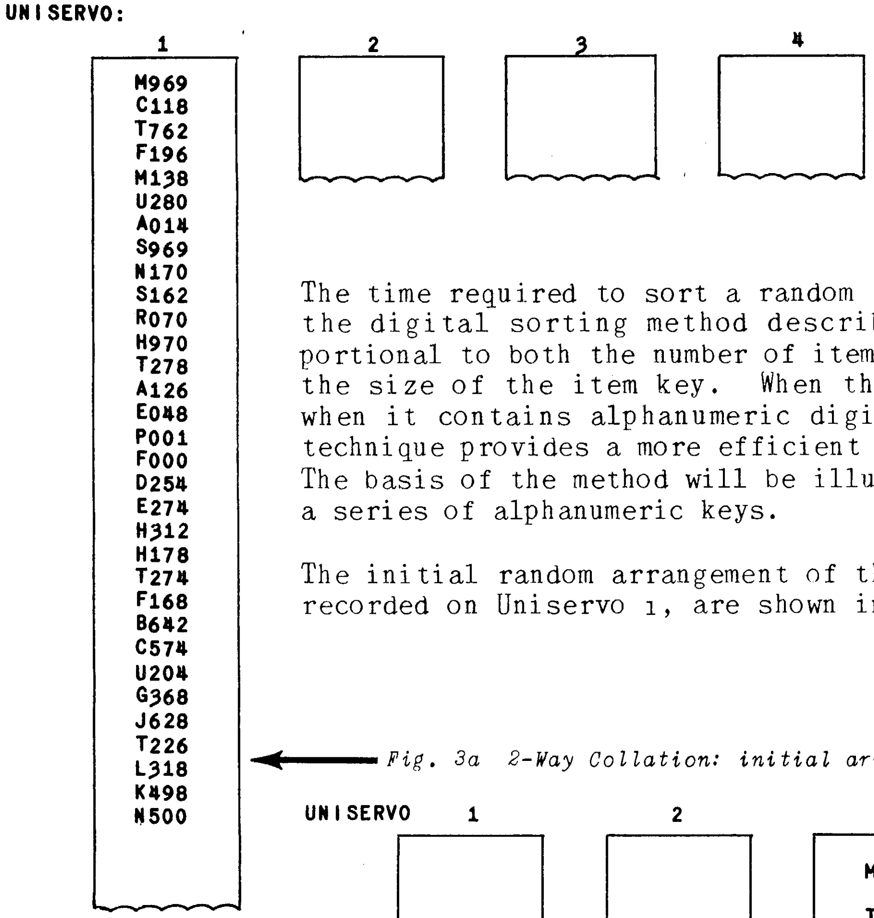
For example, to sort 12,000 ten-word items on a two digit key would require 21 minutes, plus 7 minutes for rewinding input and output tapes. The reader will note that for each digit of the sorting key, the items must pass through the computer three times, first dispersion, second dispersion, and collection.

ITEM SIZE	NUMBER OF ITEMS	MINUTES PER DIGIT OF THE KEY
2	60,000	26.4
4	30,000	13.2
10	12,000	10.5

COLLATION

Digital sorting, as described in the preceding section, is the act of arranging items of information according to rules dependent on a key contained in the items. *Collating*, the second method to be discussed here of rearranging information in a desired sequence, combines two or more similarly ordered sets of items to produce another ordered set composed of information from the original sets.

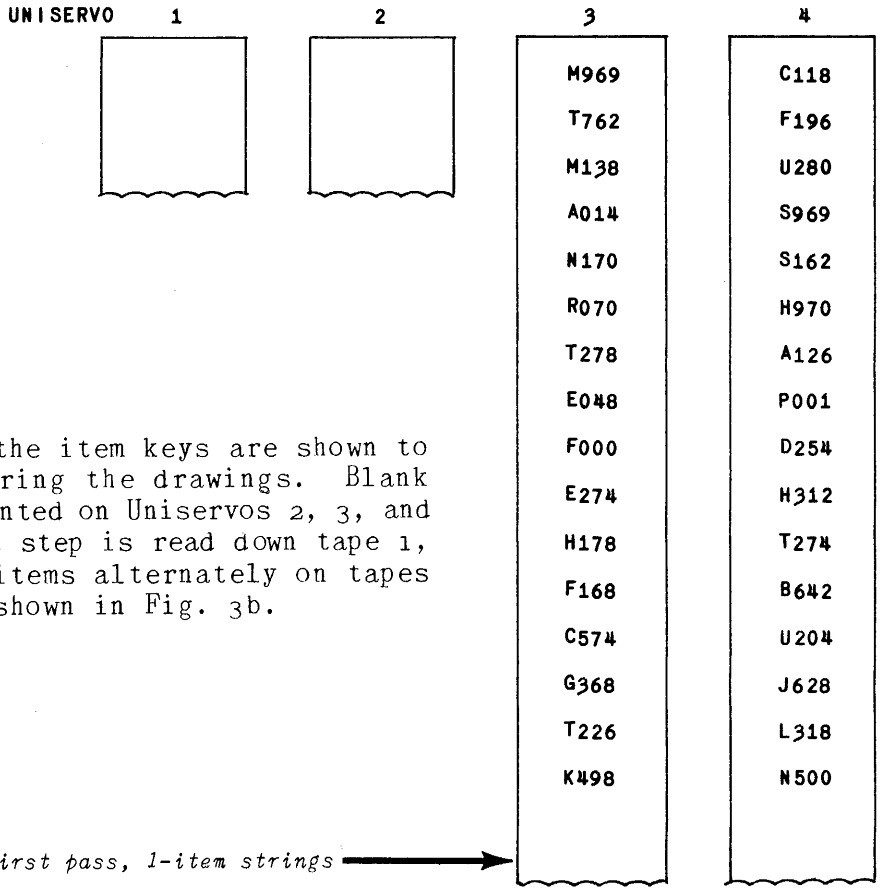
It goes without saying that the method of sorting to be selected will be largely governed by the form and order of the information, and by the item size and the key size. The efficiency of the operation is the prime consideration.



The time required to sort a random series of items by the digital sorting method described above is proportional to both the number of items to be sorted and the size of the item key. When the key is large or when it contains alphanumeric digits, the collation technique provides a more efficient method of sorting. The basis of the method will be illustrated by sorting a series of alphanumeric keys.

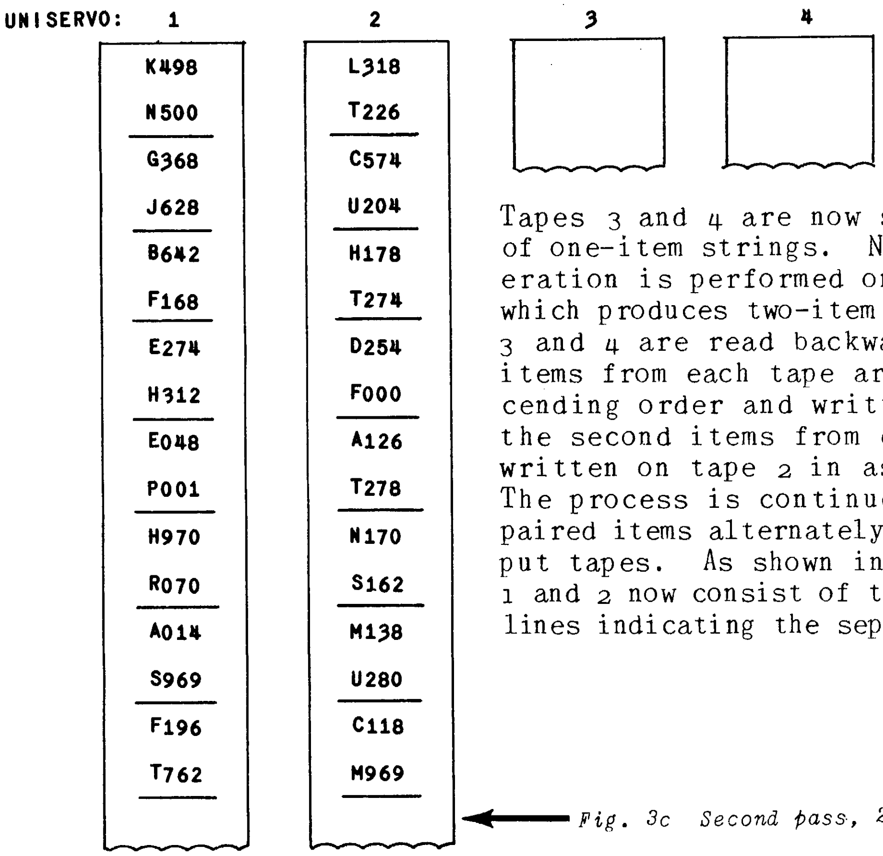
The initial random arrangement of the input items, as recorded on Uniservo 1, are shown in Fig. 3a.

Fig. 3a 2-Way Collation: initial arrangement of items



Again, only the item keys are shown to avoid cluttering the drawings. Blank tapes are mounted on Uniservos 2, 3, and 4. The first step is read down tape 1, writing the items alternately on tapes 3 and 4, as shown in Fig. 3b.

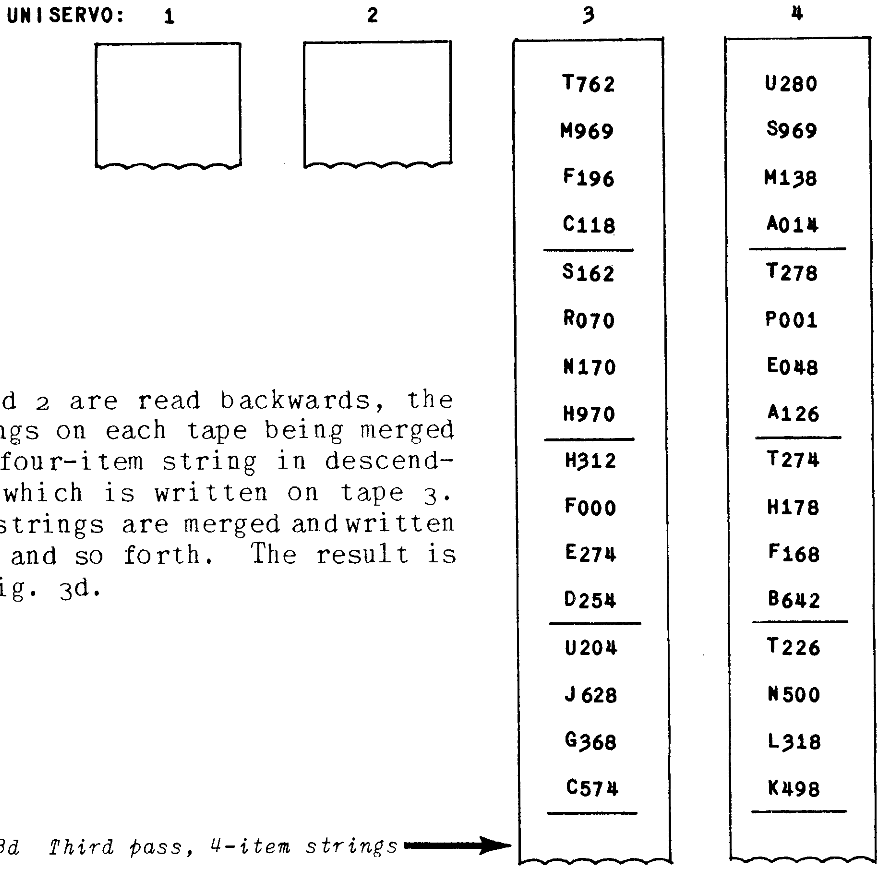
Fig. 3b First pass, 1-item strings



UNISERVO: 1	2	3	4
K498	L318		
N500	T226		
G368	C574		
J628	U204		
B642	H178		
F168	T274		
E274	D254		
H312	F000		
E048	A126		
P001	T278		
H970	N170		
R070	S162		
A014	M138		
S969	U280		
F196	C118		
T762	M969		

Tapes 3 and 4 are now said to consist of one-item strings. Next a merge operation is performed on these strings which produces two-item strings: tapes 3 and 4 are read backwards, the first items from each tape are placed in ascending order and written on tape 1,* the second items from each tape being written on tape 2 in ascending order. The process is continued, writing the paired items alternately on the two output tapes. As shown in Fig. 3c, tapes 1 and 2 now consist of two-item strings, lines indicating the separate strings.

Fig. 3c Second pass, 2-item strings



UNISERVO: 1	2	3	4
		T762	U280
		M969	S969
		F196	M138
		C118	A014
		S162	T278
		R070	P001
		N170	E048
		H970	A126
		H312	T274
		F000	H178
		E274	F168
		D254	B642
		U204	T226
		J628	N500
		G368	L318
		C574	K498

Tapes 1 and 2 are read backwards, the first strings on each tape being merged to form a four-item string in descending order which is written on tape 3. The second strings are merged and written on tape 4, and so forth. The result is shown in Fig. 3d.

Fig. 3d Third pass, 4-item strings

* Remember that the UNIVAC Tm instruction compares alphanumerically for "greater or or less than."

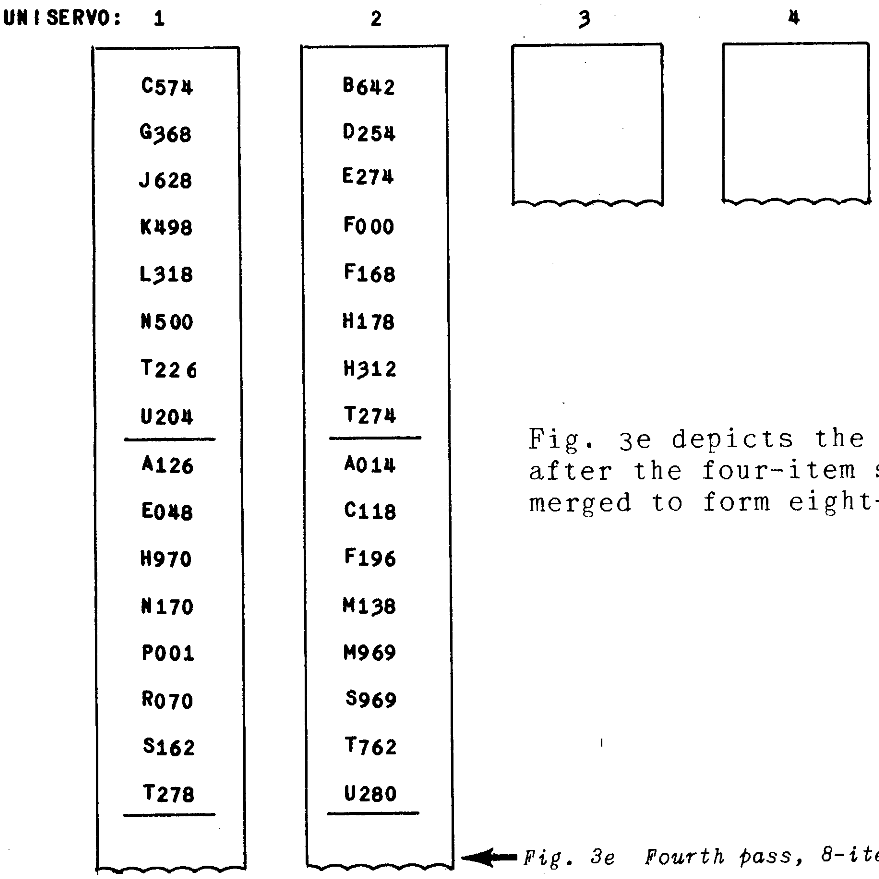


Fig. 3e depicts the item arrangement after the four-item strings have been merged to form eight-item strings.

Fig. 3e Fourth pass, 8-item strings

These eight-item strings are then merged to form sixteen-item strings as shown in Fig. 3f. Since we had only thirty-two items initially we have only two such strings.

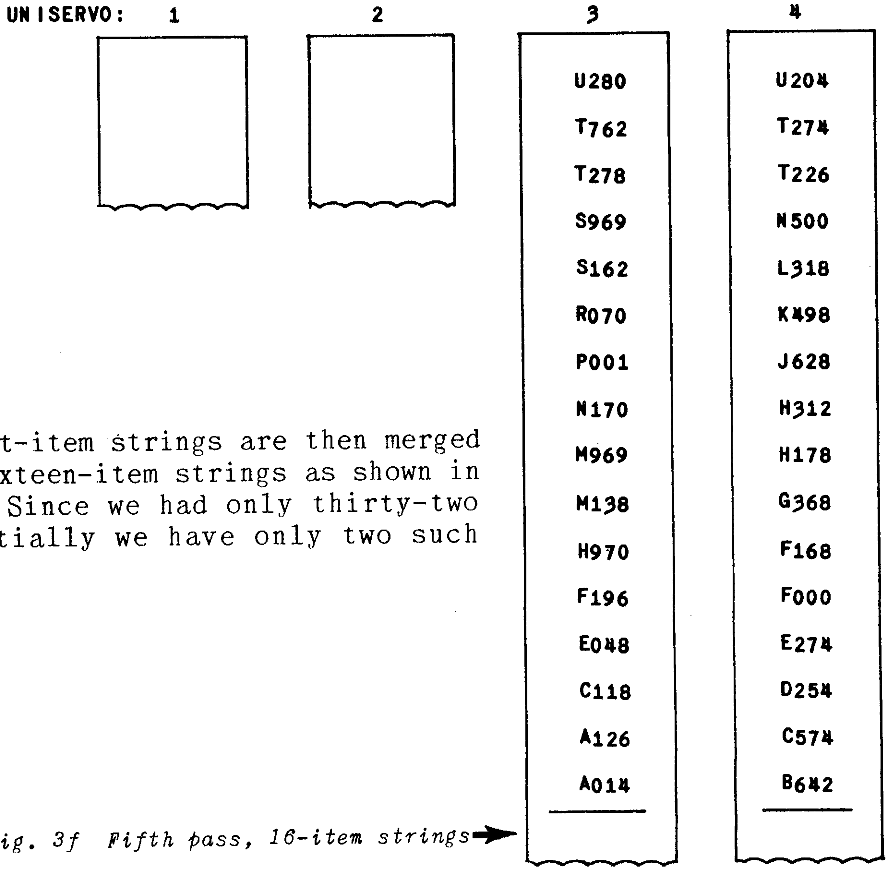


Fig. 3f Fifth pass, 16-item strings

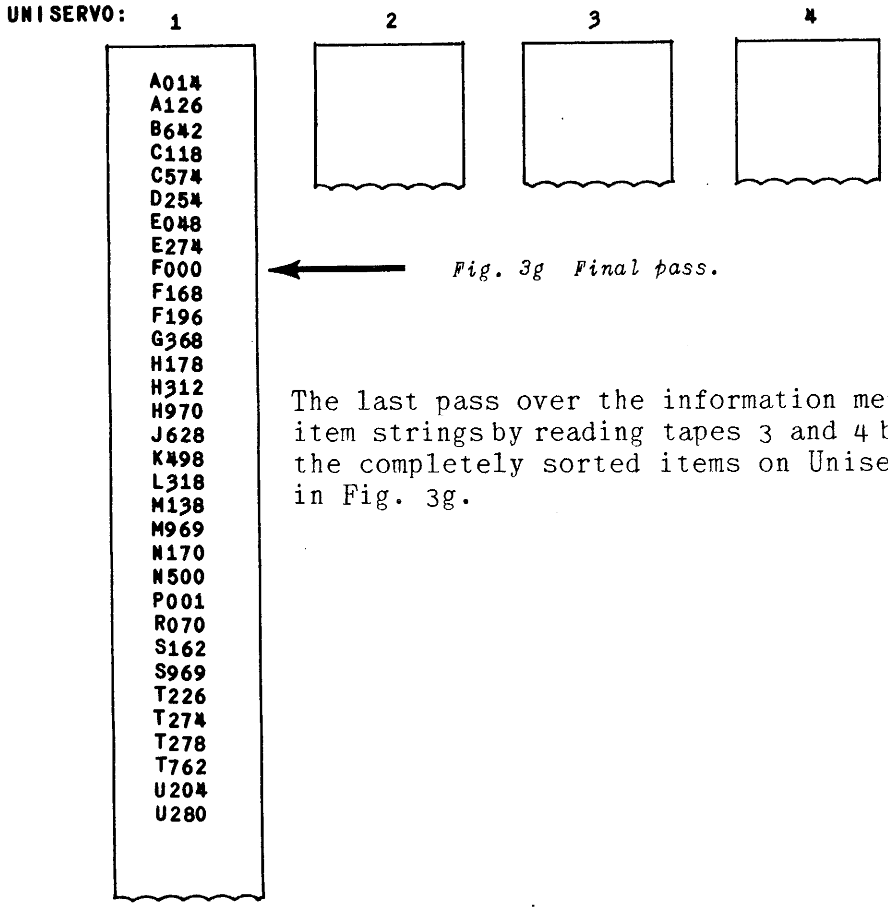


Fig. 3g Final pass.

The last pass over the information merges the sixteen-item strings by reading tapes 3 and 4 backward, writing the completely sorted items on Uniservo 1, as shown in Fig. 3g.

Collation, thus, consists of successive merging of item strings. After each merge, the number of ordered items within each string doubles and the number of such strings is halved. It is apparent that the same number of tape passes could have sorted these items no matter what size key was involved. This fact is the great power of this sorting method.

When items are actually collated on the UNIVAC a modification is made in the first tape pass. Each block of randomly ordered input items read into the computer is ordered before it is written on the output tape. Thus, if two-word items are to be sorted, the first pass puts out thirty-item strings (one-block strings).

The collation just described is called two-way collation. The minimum number of Uniservos required is four, and the number of tape passes for sorting a full tape of 2000 blocks is twelve. When six Uniservos are available, three-way collation is a more efficient method. This proceeds exactly as two-way collation, except that on the first pass the input data is divided among three output tapes. This permits a series of three-way merges. Thus, the number of blocks in the strings increases by powers of three rather than two as in a two-way collation. Eight tape passes are required to sort a 2000-block tape.

Since sorting is a basic operation for all commercial applications of a computer, considerable effort has been, and is still being, expended in further improving the efficiency of the collation method through simple modifications of the described technique and the use of minimum latency coding. The following representative sorting times for three-way collation have been obtained:

12,000	10-word items (full tape)	28 minutes
30,000	4-word items (full tape)	50 minutes
60,000	2-word items (full tape)	68 minutes

These figures are based upon a 12-digit alphanumeric key. Each additional twelve digits added to the key increases the sorting time by about five to ten percent.

FUNCTION TABLE SORT

While the collation method provides a powerful general sorting method, there are occasions when the nature of the items to be sorted permits specialized techniques that, for those cases only, are more efficient. One such technique is called the Function Table Sort.

This technique is feasible when the following conditions are met:

1. The sorting key is numeric and the number of different keys is small.
2. No two input items have the same key, unless they can be summarized into a single item of the same size.
3. The item size is small. 2 or 4 word

This method will be illustrated by the following example. Assume the input data consists of a series of two-word items, and that the item keys are sequentially numbered from 0 to 4999 though not all items bearing those keys need be present.

The first run reads the input items recorded on tape 1 and disperses them on tapes 2, 3, 4, 5 and 6. Those items with item keys 0 to 999 are written on tape 1, keys 1000 to 1999 on tape 2, and so forth as shown in Fig. 4. This can be called the first dispersion run.

The second dispersion run reads tape 2 backwards, distributing the items on that tape among tapes 3, 4, 5, and 6; the items being written following the first dispersion items on that tape. Sentinel blocks may be used to mark the separation between the two sets of items. Again, as shown in Fig. 4, items with codes 0-249 are written on tape 3, codes 250-499 on tape 4, etc.

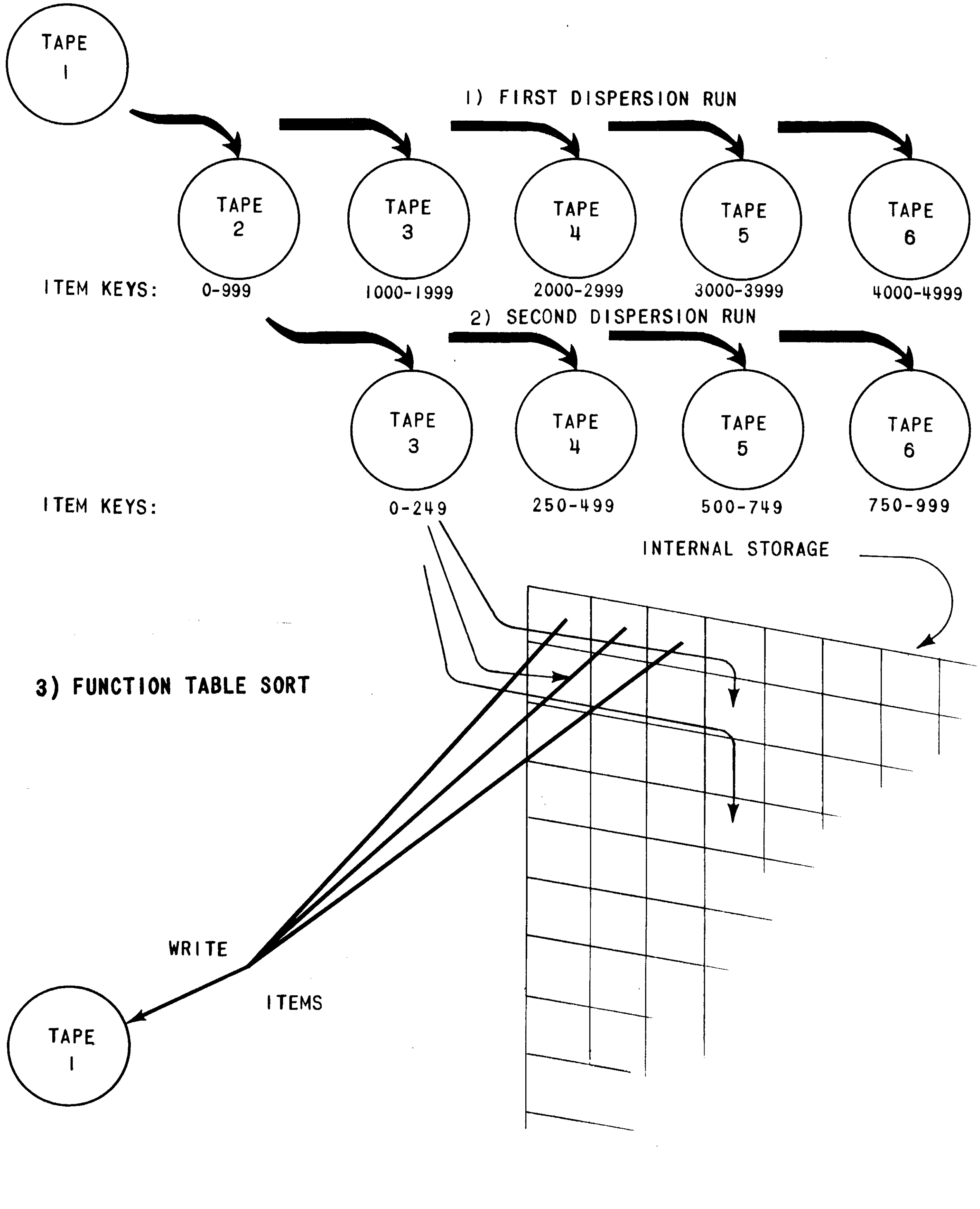


Fig. 4 Function Table Sort

The third step is the function table sort proper. Five hundred consecutive locations are reserved in the memory. They will store 250 items; they are cleared to zero initially. The items with codes 0-249 from tape 3 are read into the computer. The key of each item determines the memory cells in which the item is to be placed. Cells 500 and 501 are reserved for items with key 0, 502 and 503 for items with key 1, and so forth. The memory address is determined from the following formula:

$$\text{Memory address} \quad m = 500 + 2(k - 250n)$$

where k is the item key and $n = 0$ initially. Then the item is transferred with a V W instruction-pair to cell m and $m + 1$. If more than one item with a given key is possible, the quantity fields in the items are added to the quantity fields already stored in the paired memory cells.

Thus, as each item is read into the computer a memory address is fabricated from its key and the item stored in that memory location and the one succeeding. If an item is already in that location the quantity field is added to the item stored. The process is akin to a post office pigeonhole set-up where letters selected at random are stuffed into the proper route box. After all the 0-249 items have been stored in their appropriate memory cells the computer then "looks" in locations 500, 501. If an item is present, it is written on the output tape and locations 502, 503 are examined. If no item is stored here locations 504, 505 are examined. In this manner the items which were stuffed into the proper pigeonholes in random fashion are now extracted and written on tape 1 in item-key sequence.

After all 250 paired memory locations have been examined, they are cleared to zero and the items from tape 4 with item keys 250-499 are read into the computer. These items are stored in the appropriate memory locations, the address being fabricated from the formula given above, with $n = 1$. Thus, the item with key 256 is stored in cells 512 and 513. Again, after all these items have been stored, the cells are emptied sequentially and their contents written on the output tape: Uniservo 1.

This step is repeated for each set of 250 item keys. Then the second dispersion and function table runs are repeated for each of the items on tapes 3, 4, 5, and 6. Tape 1 now contains the sorted items.

Only four passes over the information are required: one for first dispersion, one for second dispersion and two for the function table sort (one pass to store the items, one pass to pick them out of the cells and write them on tape 1). The time required for each pass is about the same as for the digital sort or collation method on like size items, since the same general operations are required. The time required in this example (60,000 two-word items with key of four digits) is approximately 40 minutes as compared with 68 minutes for three-way collation and 125 minutes by digital sort. Of course, the method is not of universal application, but on sufficiently restricted input data it is highly efficient. Thus, the nature of the input items to be sorted should be given careful study in order to determine if such special sorting techniques are applicable before recourse to the general method of collation is made.

The sorting methods described in the preceding sections are designed to sort one reel of data only. Where the amount of data to be sorted exceeds one full tape, a merge operation must be done to produce the multi-reel ordered items. For example, if three reels of data are to be sorted, the three reels are first individually sorted by that sorting method most efficient for this data, among those described. Then the three sorted reels are put through a three-way merge to form a single ordered file of three reels.

The merging time is not trivial for very large amounts of data. Generally speaking, the merging time required depends on the number of Uniservos available and the number of full reels to be merged. The objective in sorting and merging work is to keep the number of tape passes to a minimum. Because of this it is desirable to have the maximum number of input reels to merging runs as the Uniservos available will allow. Fig. 5a is a table giving the 3-way merging time for various item sizes and numbers of input reels (includes all rewind times).

ITEM NO.	3 REELS		9 REELS		27 REELS	
	ITEMS	TIME(MIN)	ITEMS	TIME(MIN)	ITEMS	TIME(MIN)
2-WORD	**180,000**	**30**	**540,000**	**162**	**1,620,000**	**960**
10-WORD	**36,000**	**14**	**108,000**	**77**	**324,000**	**329**
60-WORD	**6,000**	**14**	**18,000**	**77**	**54,000**	**329**

Fig. 5a Times required for 3-Way Merge

Care should be taken in the manner in which merging operations are performed. As an example, consider the merging of 12 reels with the number of Uniservos permitting a maximum of 3-way merging. The straightforward approach would do four 3-way merges producing four ordered piles of three reels each. Then three of these piles would be merged to form one ordered pile of nine reels which is then merged with the fourth, three-reel pile to form one pile of twelve reels, as shown in Fig. 5b. The number of tape passes is 33.

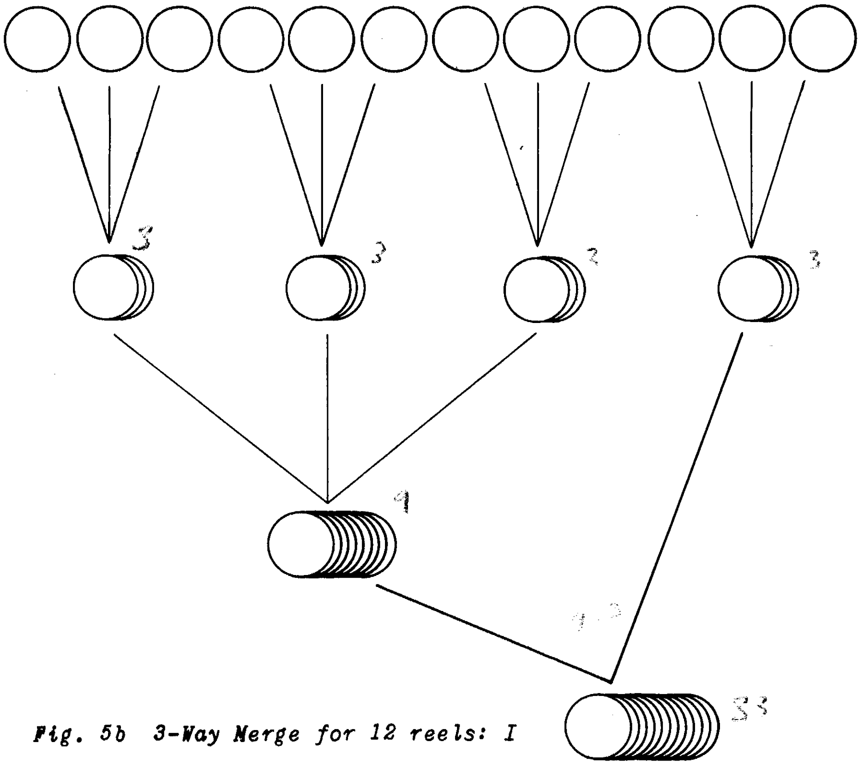


Fig. 5b 3-Way Merge for 12 reels: I

However, by merging in the manner shown in Fig. 5c only 29 tape passes are involved. Four tape passes for two-word items amount to over half an hour of computer time. An algorithm exists for expressing the exact manner in which the merging may be done for minimum tape passes.

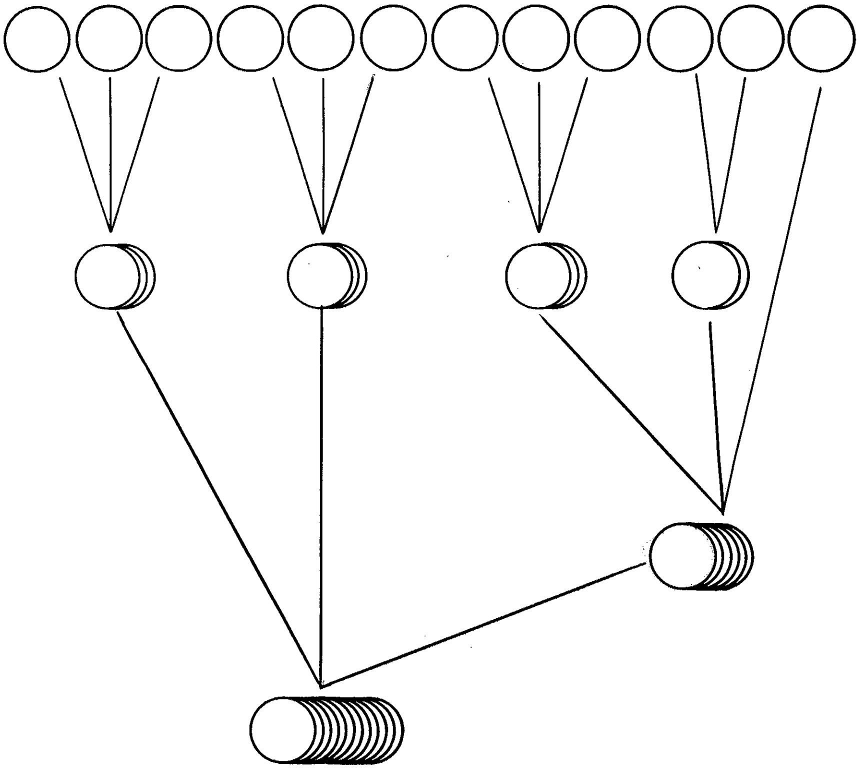


Fig. 5c 3-Way Merge for 12 reels: II