1966

Magnetic tape and disc file organizations for retrieval

Robert M. Curtice
Lehigh University

MAGNETIC TAPE AND DISC FILE ORGANIZATIONS FOR RETRIEVAL

by

Robert Morris Curtice

A Thesis

Presented to the Graduate Faculty

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Information Science

Lehigh University

1966

Center for the Information Sciences
Lehigh University

EXPERIMENTAL RETRIEVAL SYSTEMS STUDIES

Report No. 1

Magnetic Tape and Disc File Organizations for Retrieval

by

Robert M. Curtice

The work reported here
was supported by the
National Science Foundation,
Grant No. GE-2569.

July 1966

EXPERIMENTAL RETRIEVAL SYSTEMS STUDIES

The Center for the Information Sciences has developed and maintains an experimental system for the literature of the information sciences. At present the collection contains about 2,000 documents and is used for instruction, reference, research and experimentation.

Documents are indexed manually and a coordinate index system is used with a controlled thesaurus. Posting, up-dating, author listings, and both associative and non-associative searches are performed on the GE-225 computer.

In addition, a growing collection of natural language text on tape is maintained for automatic indexing and abstracting studies.

This series of studies will report experimentation and research on this operating system.

This Thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

May 18, 1966
(date)

John M Carroll
Professor in Charge

Richard S. Taylor
Head of the Department

This work was supported by the National Science Foundation under Grant GE-2569

ABSTRACT

General considerations involved in the design of file organizations for information retrieval systems are discussed. The question of inverted file and serial file searching on magnetic tape is reviewed. Based on formulas concerning index term usage, it is shown for a specific collection that searching the serial file is more efficient than searching the inverted file. This result is then generalized to large collections. Two existing file organizations serve as illustrations. Disc storage devices are discussed, together with disc organization of specific files. It is concluded that the use of an inverted file scheme is most applicable on a disc.

TABLE OF CONTENTS

LIST OF FIGURES

MAGNETIC TAPE AND DISC FILE ORGANIZATIONS FOR RETRIEVAL

by

Robert Morris Curtice

ABSTRACT

General considerations involved in the design of file organizations for information retrieval systems are discussed. The question of inverted file and serial file searching on magnetic tape is reviewed. Based on formulas concerning index term usage, it is shown for a specific collection that searching the serial file is more efficient than searching the inverted file. This result is then generalized to large collections. Two existing file organizations serve as illustrations. Disc storage devices are discussed, together with disc organization of specific files. It is concluded that the use of an inverted file scheme is most applicable on a disc.

Magnetic Tape File Organizations

Introduction

The decision to mechanize an information retrieval system on magnetic tape imposes some severe restrictions on file design. Many systems, however, that are in operation do use magnetic tape for storage. Each system has its own file design. There have been many types of file organizations suggested, each having its distinct advantages and disadvantages. To evaluate these different designs we must develop criteria for evaluation.

For the purposes of this study we shall not be concerned with the evaluation of the total system, which depends on the search strategy, vocabulary, etc., as well as file design. Whatever the searching procedures are, however, we can measure and evaluate the efficiency of the file organization. We shall consider the total processing time it takes to implement the systems programs as measuring the efficiency of the file organization. Our general method shall be to compare the number of records needed to be read for the average search based on certain file organizations. Since internal manipulations for search programs based on different file organizations are similar, and because internal manipulation time is much faster than tape searching time, the determining factor in total searching time is the number of records that must be read before the search can end.

The main program of any retrieval system is the search program. We shall require this program to include some bibliographic output concerning those items retrieved. We must also consider the difficulty of updating the files imposed by a particular design.

Record Length

One of the most important questions to be answered before designing a file organization concerns record length. Fixed length records must be distinguished from variable length records. Fixed length records shall be considered as those which will never change in length, as for example a dictionary entry. Variable length records will change, as, for example, in an inverted file, the number of item tokens under a specific index term.

First, many compiler languages write physically fixed length records. For example, if the compiler writes records of 100 words long, it would be a great waste of space to make records 110 words long, because in this case the compiler would actually write two records, 100 words long each, when given a command to write a (logical) record of 110 words.

Second, many compilers, such as Fortran II, save two computer words for each subscripted variable, even in fixed point arrays. This is a waste of storage space and can become acute, especially if core storage is at a premium. These two words are then written on tape for each item in an array, one of them being completely blank. Thus, in writing an array 100 items long on tape, the compiler actually writes 200 words on tape with every other word blank. In our example, only arrays 50 items or less will be written as a single physical record.

When dealing with compiler languages these two considerations should dictate the record length. When the latter consideration is in effect, records should be multiples of one-half the record length written by the compiler. When only the former consideration is applicable,

records should be whole multiples of the length written by the compiler.

Any other solution to this waste of storage space or dictation of record length involves languages of lower level. Besides being able to choose the length of records which may contain data in every word, we gain the ability to block records. This technique is used in consideration of the fact that it takes less time to read one record, than to read two records each half the original record's length. This is because the start-stop time of the tape handler is generally time consuming and extra time is used in reading over the inter-record gap which separates the records. Thus we can block our records into longer ones and read, for example, 10 logical records at a time.

When considering variable length records it is unwise to choose a fixed length greater than the largest record needed. Experience has shown that a few of the terms are used to index many documents while the majority of terms index only a few documents (1). Thus with a moderate record length most terms will use only one record. Chaining provisions for multiple records are usually easily accounted for in programs when a term must use two or more records. A special sentinel in the record would indicate that the next record on the tape is a continuation of the previous one, and this technique can be used an indefinite number of times.

One last technique may be useful when dealing with variable length fields. For example, a record which contained the bibliographic data concerning a document might be 100 words long. However, some documents would have longer titles and shorter author names, while the opposite

may be true of other documents. If fixed fields are imposed there is no flexibility to account for this possibility. The use of a record image allows us to distribute the 100 words for each record by indicating in the first word where the author name begins. For example, suppose the record for a certain document contained the title in words 2-63 and the author in words 64-100. Then, by putting the number 64 in word one, we know where the second field begins.

Inverted and Serial File Organizations

"Some orderly arrangement of files is required for their efficient use" (2). Since the majority of processing time in an information retrieval system is spent on input-output functions the file design could very well decide the efficiency of the total system.

There exist many general rules for efficient file organization. Some of the more applicable ones are discussed above. However, almost all file designs for information systems have been variations of two main organizations. The first is generally called the inverted file while the second is called the direct or serial file. The organization of these files can be explained with the help of figure 1.

		Documents					
		1	2	3	4	5	6
Terms	A	X	X	X	X		
	B		X		X		X
	C	X		X		X	
	D		X		X	X	
	E			X			X
	F	X	X			X	

Figure 1. The Term-Document Matrix

In figure 1, the letters represent index terms, the numbers represent documents. This representation is usually referred to as the term-document matrix.

The 'X' in the left uppermost position tells us that term A indexes document number 1. Put another way, it tells us that document number 1 is indexed by term A. The fact that the matrix is very sparse, with only two values, makes two other representations more efficient. Figure 2 shows the serial file arrangement and figure 3 shows the inverted file arrangement for the data given in figure 1.

1	A	C	F	
2	A	B	D	F
3	A	C	E	
4	A	B	D	
5	C	D	F	
6	B	E		

Figure 2. The Serial File

A	1	2	3	4
B	2	4	6	
C	1	3	5	
D	2	4	5	
E	3	6		
F	1	2	5	

Figure 3. The Inverted File

A typical record for a serial file might contain the accession number of a document, the codes of the terms which index the document, the author and title of the document and certain bibliographic data. Thus, all the data concerning the document are located in a single record. An inverted file record might contain the term in alphabetic notation, its code number, a scope note, and all the documents which it indexes. In an inverted file, all data concerning an index term are located in a single record.

One may observe that, if data were required about a term, the inverted file would be more efficient to use, while the serial file should be consulted for data concerning a document. Thus the search program operating on an inverted file would be able to retrieve those document numbers which satisfy a search in the most efficient manner, since it requires data concerning a term (i.e., which documents it indexes). Note, however, that we require bibliographic information which the inverted file does not contain, necessitating the use of a bibliographic file. A bibliographic file contains the same bibliographic information concerning each document but does not contain which terms index the document. Thus, the entire bibliographic file contains exactly as many records as the serial file.

The basic question under consideration is then, which scheme requires fewer records to be read, on the average, the serial file or the inverted file with bibliographic look-up.

The proponents of the inverted file scheme argue that searching the serial file for all numbers of documents indexed by a term requires a complete pass of the file. While this is not necessarily the case

(as will be shown), it is obvious that the same data can be obtained from the inverted file with less searching, i.e., once we pass the record for the search term, we have the document numbers. However, the pass on the serial file produces the bibliographic data, while the inverted scheme must do more tape passing on the bibliographic file. Also, those who prefer the inverted scheme will claim that more comparisons (i.e., subtractions) must be made while passing the serial file. Thus,

> The first method (serial) was soon abandoned because of the projected size of the file and because too many comparisons had to be made. (3)

The inverted file scheme also requires more tapes, and Swid points out that tape mounting requires time. (4) For example, the description of the file organization for the Saint-Goban Company states:

> The entire (inverted) file took up about 750 feet of tape...Another tape was used to hold the dictionary of descriptors...In addition, a third tape was used to hold bibliographical identification for each document number. (5)

Although neither of these file organizations is optimum, the exact nature of the number of records passed for a search will be derived later.

When documents to be added to the file are indexed, the recording of index terms for a particular document forms a serial file record. Thus updating a serial file is quite easy. The process of inverting this data to update an inverted file is time-consuming and has been used as an argument against the inverted method. Therefore, in describing reasons for converting the NASA Scientific and Technical Information Facility files to a serial organization, Miller, Swid,

and Rosen state:

> Although the inverted concept produced satisfactory searching and statistical data, the benefits were offset by the unwieldiness and expense of frequent updating and changes to the multiple files. (6)

The inverting process is usually done by associating the document with each of its index terms separately, then sorting on the terms. This can be done on cards or by the update program itself. The data flow for this process is exemplifed in figure 4.

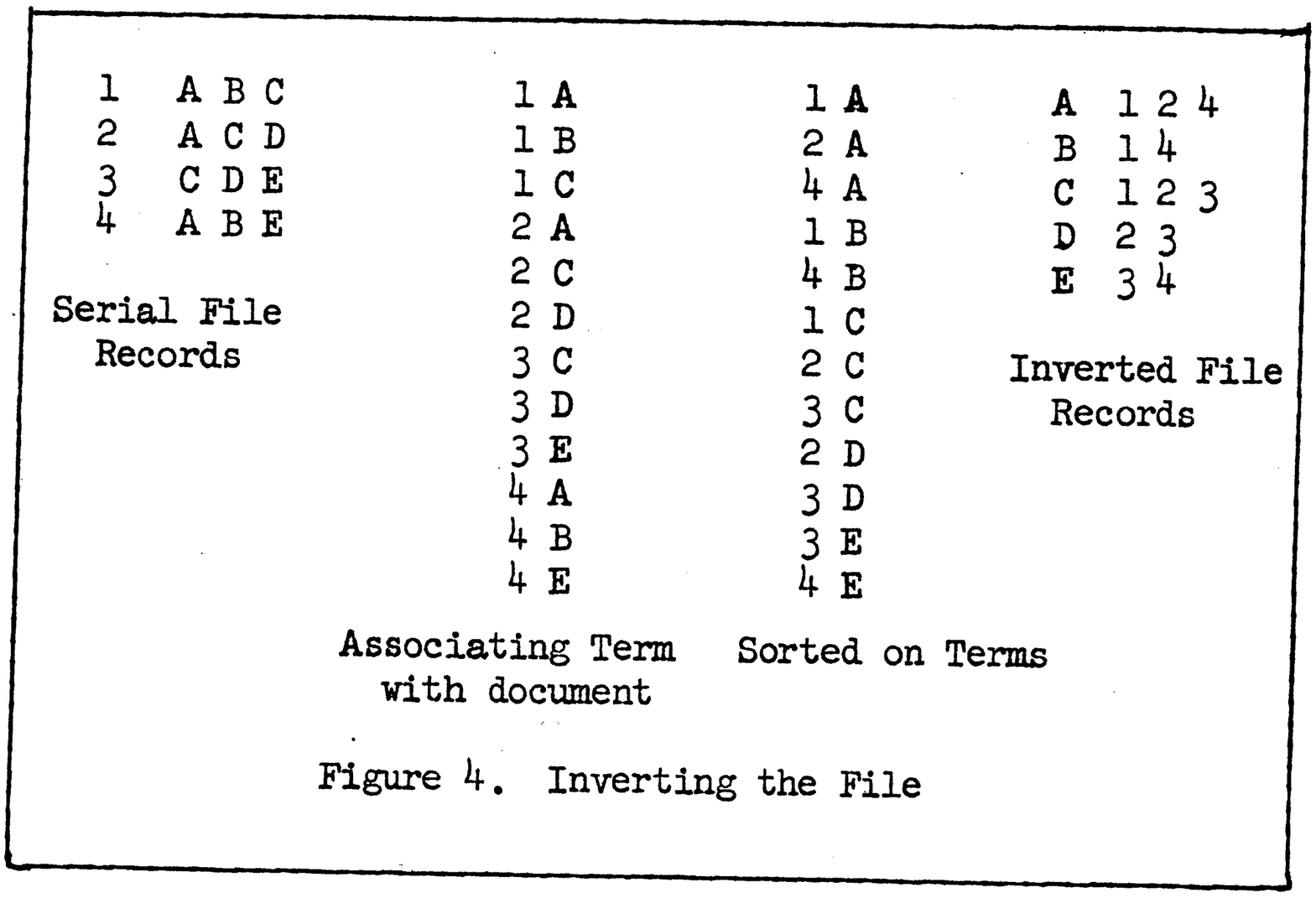


Figure 4. Inverting the File

Again Miller, Rosen, and Swid state:

> The linear system provided considerable improvement in speed in the printing of journal indexes and cumulations. For example, preparation of three indexes....required only 45 minutes to select the desired records from the entire (serial) file. The previous (inverted) system.... would have taken at least ten times as long. (7)

Many systems employ the use of Bound Terms, or indications of relationships between terms used to index a particular document.

For example, the following serial file entry indicates a relationship between terms A and B:

13715 C, F, A*B

The asterisk may indicate that term A modifies term B. Such a relationship would be difficult to indicate in an inverted file, as shown by Andrews (8). If one did intend to indicate such a relationship in an inverted file, it would be necessary to create a new term, such as the single term A*B. However, this method imposes restrictions on the vocabulary of the system.

There have been many methods proposed to compensate for any inefficiency introduced by a particular file design. Some of these proposals are standard data processing techniques of general application. For example, batch processing of searches, consolidation of files, and increases in hardware capabilities will improve the performance of any system. These methods will not be discussed in detail.

One of the approaches to make the direct file scheme more acceptable has been file-splitting. Essentially this method proposes the serial file be split into parts on the basis of whether or not the documents fall into a certain category, or into which of many categories the documents fall. If the category is not one which causes the parts of the file to be mutually exclusive, such as a subject category, then duplicate records of documents are needed. If a category such as the date of publication is used then no duplicates are necessary. The method presupposes that some statement about the category can be made for each request, thus allowing only part of the file to be searched. For example, a request for all documents on subject X, published

before 1960, would require a search on only that part of the file which contained documents published before 1960. However, if the date was not important, the entire file must be searched. Thus Whaley states:

> Who knows but what the portions of the file eliminated more or less arbitrarily by the inquirer or retriever in trying to guess source or date limits may contain an important answer to the inquiry? (9)

Usually partioning according to a subject category works better than by date, although a request using terms which are more generic than the criteria for file-splitting would again require a search on the entire file. General rules for file-splitting are discussed by Perry and Kent. (10).

A proposal to make the inverted file more efficient suggests taking advantage of the fact that records located at the beginning of the tape are found the quickest. Thus, records of terms which are used the most often (i.e., the most popular search terms) can be placed where they are most readily accessable. To accomplish this goal, the system must be in operation for a sufficient time so that 1) vocabulary changes will be at a minimum, and 2) records can be kept to determine which terms are the most popular in searching. A similar proposal has been made for the most popular documents by Goffman and Badger. (11). This scheme would be helpful only if the search could end when a certain number of documents has been retrieved, otherwise the entire file would have to be searched. Compare Swid:

> There is no particular file sequence (i.e., alphabetic, frequency of use, etc.) that will substantially reduce the tape passing time. (12)

We have seen from this discussion that there are many considerations in determining an efficient file design. Claims and counter claims for the preference of one design to another are numerous. Thus, Vickery states:

> To sum up, it is not established that either term (inverted) entry or item (serial) entry has a clear advantage over the other. Comparisons of particular systems using these two principles have been undertaken, but the results cannot be generalized. (13).

Analytical Comparison of the Serial and Inverted Files

In this section we shall attempt to show, for a specific example, the relative efficiencies of the serial file and the inverted file with bibliographic look-up schemes. Many variables affect the search time for a particular file organization, and while they do not all concern us here, some of the important factors can be listed as follows:

Computer Characteristics:

- Bit transfer rate of tape units
- Tape start-stop time
- Command execution times

File Characteristics:

- Number of documents
- Number of terms
- Average number of terms per document
- Average number of documents per term

File Organization:

- Length of serial file record
- Length of inverted file record

Length of bibliographic file record

Frequencies of use characteristics

Chaining techniques

Blocking factors

Our methodology shall use the relationships among file characteristics to estimate the number of records needed to be read with each file organization for a specific collection. We will then generalize on the results to show that the serial file is, for the majority of cases, more efficient than the inverted scheme. An improvement on the basic serial file structure is then discussed.

Although good formulas exist for the determination of index term usage, vocabulary size, etc., we have to assume one specific number as a parameter for the collection. In this case we assume the average number of index terms per document to be ten.

The collection can then be said to have the following characteristics:

1) n = the number of documents in the collection.

2) 10 = the average number of terms posted per document.

3) $18n^{\frac{1}{2}}$ = the total number of terms in the collection, as given by reference (14).

4) $\frac{n^{\frac{1}{2}}}{1.8}$ = the average use of terms (i.e., the average number of documents posted per term).

5) $10n$ = the total number of postings (i.e., index entries).

We shall make repeated use of the formula

$$A = \frac{t(x+1)}{t+1} \qquad (1.0)$$

In this formula, $\underline{x}$ is a number of serial records randomly arranged, $\underline{t}$ is a number of previously specified records, and $\underline{A}$ is the number of records needed to be read before locating the $\underline{t}$ specified ones, on the average. This formula is derived in the appendix.

A one term search on the serial file would require exactly $\underline{n}$ records passed. We assume each document in the serial file requires one record. The fact that the difference between reading times for records of unequal length is extremely small, and that serial records contain bibliographic data which makes their length comparable to those in the inverted file, allows us to assume that equal time is spent in reading a record from any file.

For the inverted file, we have $18n^{\frac{1}{2}}$ terms, so using (1.0) on the average we will have to read through

$$\frac{18n^{\frac{1}{2}} + 1}{2}$$

records to find a specified term. For the bibliographic look-up part of the search we need to estimate the number of documents resulting from the average one term search. This is given by the formula

$$A' = 1.24 + .013(10n)^{.774} \qquad (1.1)$$

suggested by Wall. (15), where 10n is the total number of postings. Notice that the total number of documents retrieved has very little effect on the processing time for the serial file, since every record has to be looked at. Thus, we need to look up $\underline{A'}$ things in the bibliographic file. Assume the bibliographic file is sequenced by document accession number or alphabetical order so the search through it is random. Then, substituting A' in formula (1.0) we will have

to pass through

$$\frac{(1.24 + .013(10n)^{.744})(n+1)}{1.24 + .013(10n)^{.744} + 1} \qquad (1.2)$$

records on the average to find the bibliographic references of the resulting documents.

For example, if we assume a collection of 10,000 documents then for the serial file we would require exactly 10,000 records passed for a one term search. For the inverted file with bibliographic look-up formula for A' tells us that the average search term is posted approximately 65 times. Thus, we would require

$$\frac{1800+1}{2} + \frac{65(10,000) + 65}{66} \qquad (1.3)$$

or a total of 10,749 records passed for a one term search.

Formula (1.1) given for the number of resulting documents is based on the usage of the average search term, which is claimed by Wall to be posted more frequently than the average index term (16). If N was the average number of documents posted per term, then the average index term would be that term which comes closest to indexing N documents. On the other hand, if every time a search was performed we listed the number of documents the search terms index, and found the average per search term, then the average search term would be the one which comes closest to indexing that number of documents. Thus the average index term would be posted only

$$\frac{(10,000)^{\frac{1}{2}}}{1.8}$$

or about 55 times. However, this figure results in a value of 10,740 records passed, still more than the number required with the serial file.

The above results show that on the average, the serial file enables the search to pass fewer records than the inverted file for one term searches. Since the serial file is easier to update we can safely say that the serial file organization is more efficient than the inverted file organization for a one term search.

In discussing multiple term searches, we should note that the bibilographic look-up time accounted for the major portion of the time in the inverted file scheme. Thus, the number of documents which satisfy the search is in direct proportion to the total processing time. Results given in the report Centralization and Documentation (17) indicate that whn n = 10,000 the average 3-term search will result in only one or two documents. It is only in a few specific cases that the inverted organization will require fewer records to be read than the serial file, and in these cases the difference is great. For example, a 3-term search on the inverted file will require 81 o/o of that file to be searched, from formula (1.0). However, bibliographic look-up time is much reduced since we are only looking for a few documents. For 1 document resulting, only 5000 bibliographic records need be read on the average. Two, three, and four documents resulting would require 6700, 7500, and 8000 records passed respectively. In the later case, the addition of all 1800 records of the inverted file still keeps the figure under the 10,000 record mark. Thus the total number of records passed for the inverted search depends almost totally on the number of resulting documents.

At this point it might seem evident that the serial file be recommended for searches which result in many documents and the inverted organization for searches which result in few documents. However, since batch processing is necessary to make a tape system economical, the number of resulting documents to be looked up in the bibliographic file will always be great enough to make the number of records needed to be read approach the entire length of the bibliographic file. These records plus the records read in the inverted file will again be greater than the total number of records in the serial file.

Figure 5 shows the case for the specific collection. The number of terms in the search can be for different searches which are batched or for terms in one search. Thus any search (or searches) which result in more than 9 documents will have to pass through more than the total number of records in the serial file, when an inverted scheme is used for searching. The limit is reached when many terms are searched for in the inverted file and many documents result. In this case the number of records passed approaches the sum of the inverted and bibliographic files. It should be noted that in larger collections the ratio between the number of terms and the number of documents will be larger, making the break even point (9 documents when n = 10,000) much higher. However, an increase in size batches in larger collections will increase the number of documents to be looked up in the bibliographic file to the point where the serial file remains more efficient.

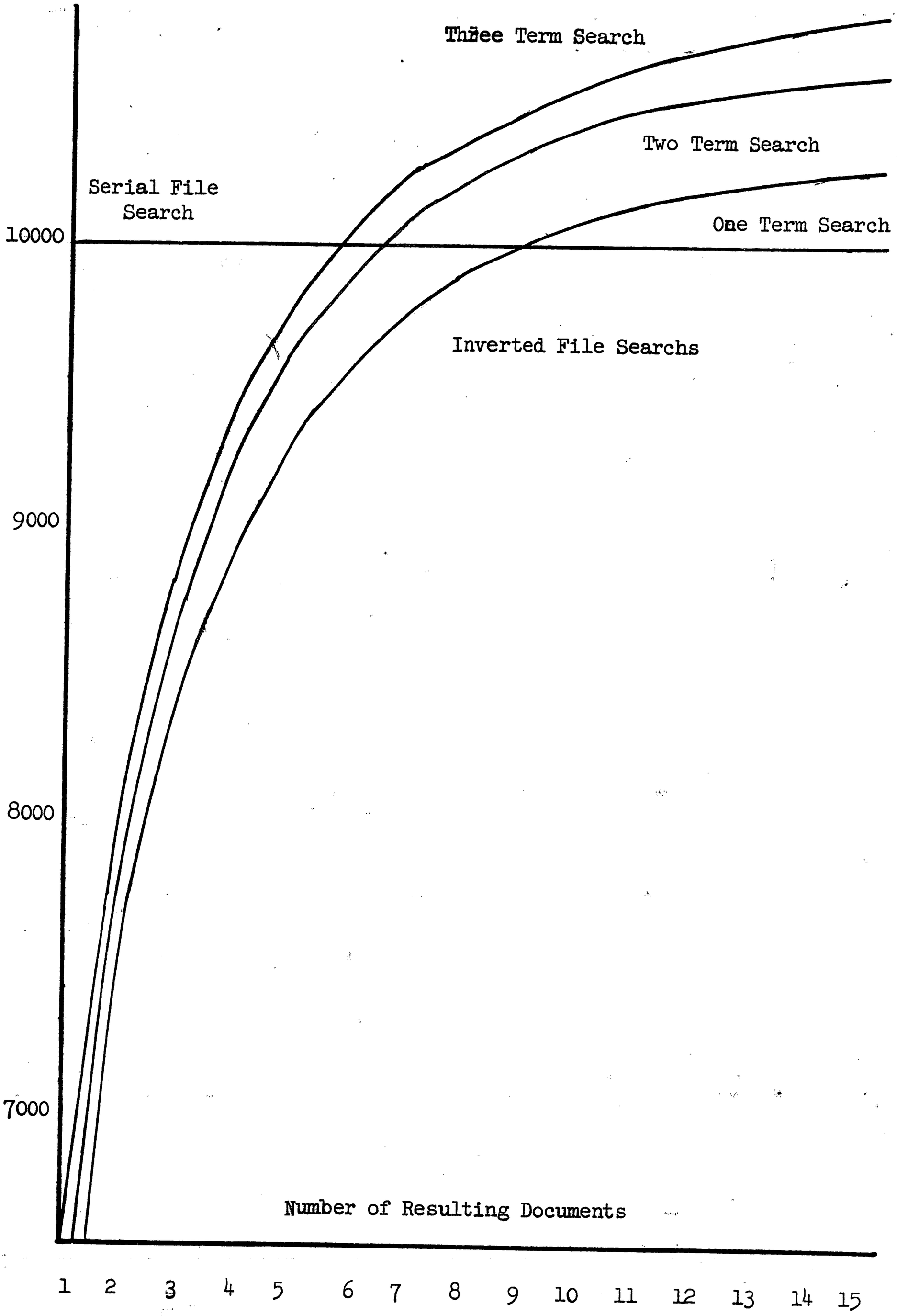


Figure 5.

Serial File Searching Vs. Inverted File Searching

The Sorted Serial File Organization

With respect to the conclusion presented above, the logical question concerns the possibility of reducing the number of records to be read in the serial file, without the maintenance of another file. Such schemes have been proposed by Vickery (18) and Libby (19). The main purpose in these schemes is to order the serial file in such a way that documents indexed by certain terms are known to appear in a certain portion of the file.

Generally the file organization requires that the term codes used to index a document appear in sequential order. The serial file is then organized with all documents indexed by the term with the lowest numerical code first. Within this group the documents are arranged according to the term with the next highest code. For example, assume the same collection of 6 documents shown in figure 3. Upon organization according to the scheme under discussion it would appear as in figure 6.

A	1	2	3	4
F	1	2	5	
C	1	3	5	
D	2	4	5	
B	2	4	6	
E	3	6		

Figure 6. The Sorted Serial File

A search for all documents indexed by terms 1 and 3 could end upon searching document C since any document indexed by term 1 must appear before any document not indexed by term 1. We have thus been

able to eliminate a portion of the serial file from further searching.

Updating this file is comparable to updating an inverted file. Once the input terms are arranged sequentially for a document, the documents are sorted by first term codes, then within each group by second term codes, etc., and finally merged into the master file.

Through this procedure we can create a single file which has all the advantages of a simple serial file, yet enables the average search to cease somewhat short of reading the entire file.

Illustration

To illustrate points made previously, and to make an actual comparison of two file organizations, we shall examine two file designs use on the Lehigh University Center for the Information Sciences document retrieval system. This system was programmed on the GE 225 computer mostly in FORTRAN II. The document collection had the following characteristics at the time of the study:

1) number of documents = 1664

2) average number of terms per documents = 4.3

3) total number of terms = 440

4) average use of terms = 16.3

5) total number of postings = 7198

The collection had an accession rate of approximately 100 documents per month.

The system has the capacity of producing term associations, which implies retrieving not only all documents indexed by certain terms but all the terms which index these documents as well. The degree of associativity between two terms is based on the co-occurrence

in the collection. A discussion on the theoretical basis for associativity as used in this system may be found in (20), and is not of concern here. However, any special considerations imposed by this aspect of the system will be fully explained.

The main features of the two types of file organization are listed in figure 7.

	Organization No.1	Organization No.2
No.of tapes required:	3	2
Number of files:	4	2
Search on:	inverted file with bibliographic look up	serial file
Block data:	no	serial file 8 records per block
Files, Lengths of records, Contents:	1) Inverted File -200 words/record -term number -document numbers 2) Direct File -30 words/record -document number -term numbers 3) Term File -15 words/record -term number -alphabetic term -class codes 4) Bibliographic File -25 words per record -document number -author + title	1) Serial File -64 words/record -document number -author + title - date -location code -term numbers 2)Inverted File -128 words/record -term number -No. of documents in record -alphabetic term -scope note -class code -document numbers

Figure 7.
Two File Organizations

In organization No.1, each of the terms in the search specification is retrieved from the inverted file. The terms are then coordinated and the resulting document numbers are used to search the bibliographic file for final printout.

In organization No.2, each record in the serial file is examined to see if it meets the search specification, and is accordingly printed out or not printed out.

It should be noted that organization No.1 which searches an inverted file contains essentially a serial file (called here a direct file since it lacks bibliographic data). This file is used in producing term associations only. Also, organization No.2 which searches a serial file contains an inverted file. Again, this file is utilized in associative searches. Deriving term associations essentially involves passing of the file <u>not</u> searched on for each organization.

The most obvious improvement in organization No.2 is the consolidation of the data from 4 files into 2 files. The advantage of this economy becomes evident only when we realize that the need for information contained in the Term and Bibliographic files is based on decisions made with information at the corresponding point in the Inverted or Direct files. For example, in searching the serial file, we need information about author and title for printout when a document satisfies the search, and by including this information in the serial file at that point it becomes available with no further searching. In addition organization No.2 requires one less physical tape.

Blocked records in organization No.2 will aid in faster searching. Additional data has been added to the files in organization No.2, but

does not concern us here.

The lengths of records in organization No.2 were chosen with conversion to a disc storage device in mind and will be discussed in the section on disc files.

The above differences in the two organizations are standard techniques already discussed. The effect of these differences on the search and update functions of the system will now be examined.

The input to the search program, naturally, specifies which terms are to be searched with. The program which operates on organization No. 1 can perform logical AND searches with a maximum of three terms, while the program operating on organization No.2 can perform and, NOT and or searches on a maximum of 4 terms. Also this second program can batch process up to 10 search requests. These three additional capabilities (more logical operations, more terms, and batch processing) are more feasible with file organization No.2. This result stems from the fact that searching with organization No.1 requires more core storage, and each of these additional capabilities requires an additional amount of storage space if utilized with organization No.1, while they require little additional space with organization No.2 The combination of introducing or searches and increasing the term capacity to 4 per search requires more storage with the inverted file search because it requires storage of the entire inverted record for terms of a search which cannot be combined until other terms have been located. For example, consider the search

$$(A \vee B) + (C \vee D)$$

If terms A and C are located first we cannot combine them, until say term B is found so. It can be added to term A. This means saving A

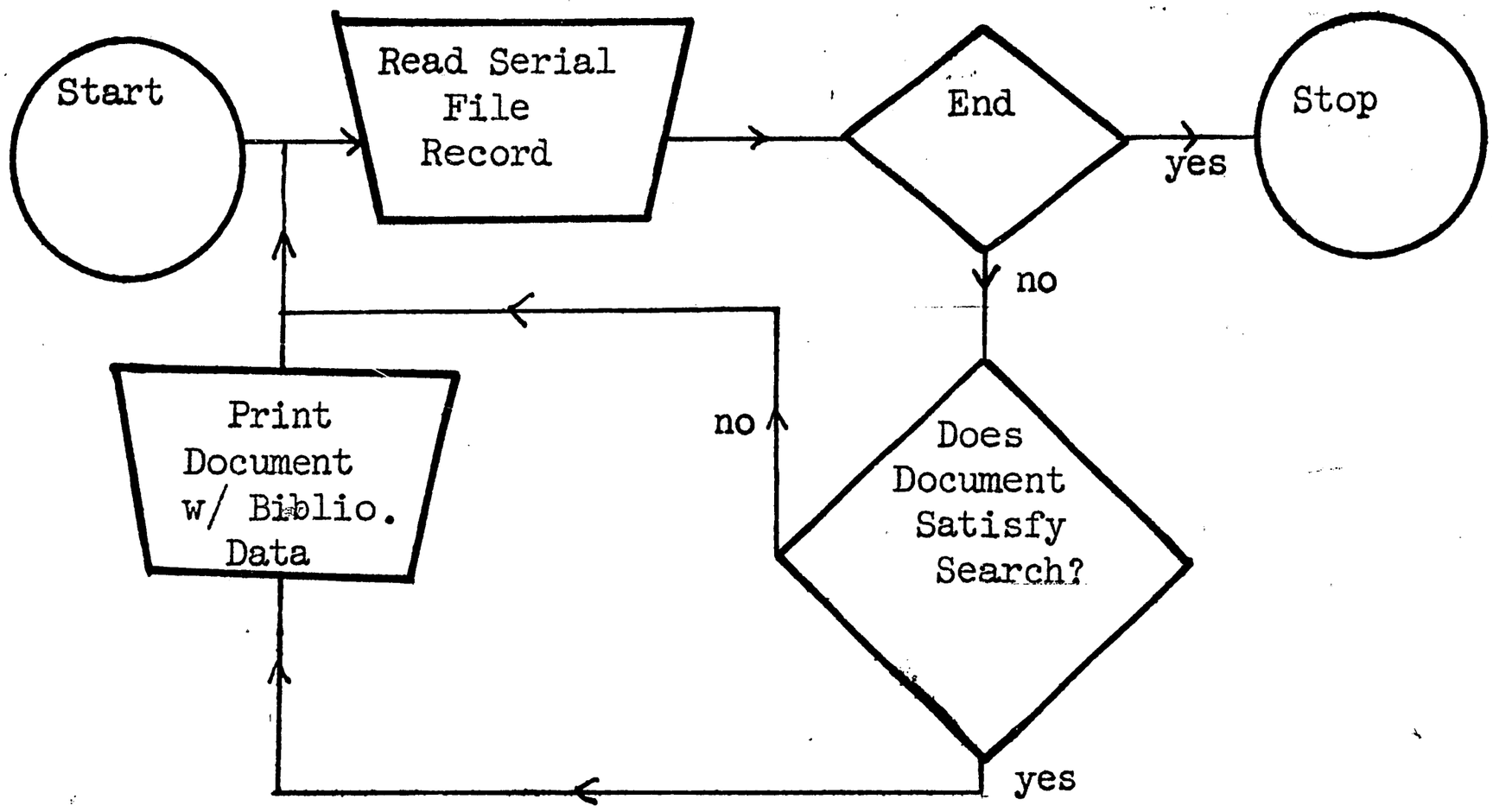
and C. Batch processing more than one search at a time will increase this storage requirement. Using the serial file on the other hand, does not require any storage of the data while searching, save the search specification itself.

These storage requirements are shown in Figure 8, illustrating the major data manipulations necessary while searching with each file organization. The complications shown for the inverted scheme will reflect in the programming effort.

The effect of using blocked records in organization No.2 increases the speed of the search by introducing fewer inter-record gaps, and therefore less tape to read. With 2000 records, for example, and 3/4" record gaps, 1500 inches of tape are required for inter-record gaps. At a reading speed of 75 inches per second, 20 second will be needed to read over these gaps. In addition, if we assume 10 milliseconds (.01 seconds) for tape start-stop time then 2000 start-stops will require another 20 seconds. By blocking records into blocks of 8, both these functions would require only 5 seconds; a net saving of 35 seconds.

In one specific example the following test was made: A one-term search was performed using both file organizations described above, and the term chosen appeared at the mid point of the inverted file. The term indexed 30 documents, and each program operating on both types of organization retrieved these documents correctly. The time for searching with the inverted scheme was 7.1 minutes. With the serial file organization the time was 2.8 minutes. In another test in which associative terms were produced the inverted scheme required

Organization No. 2



Organization No.1

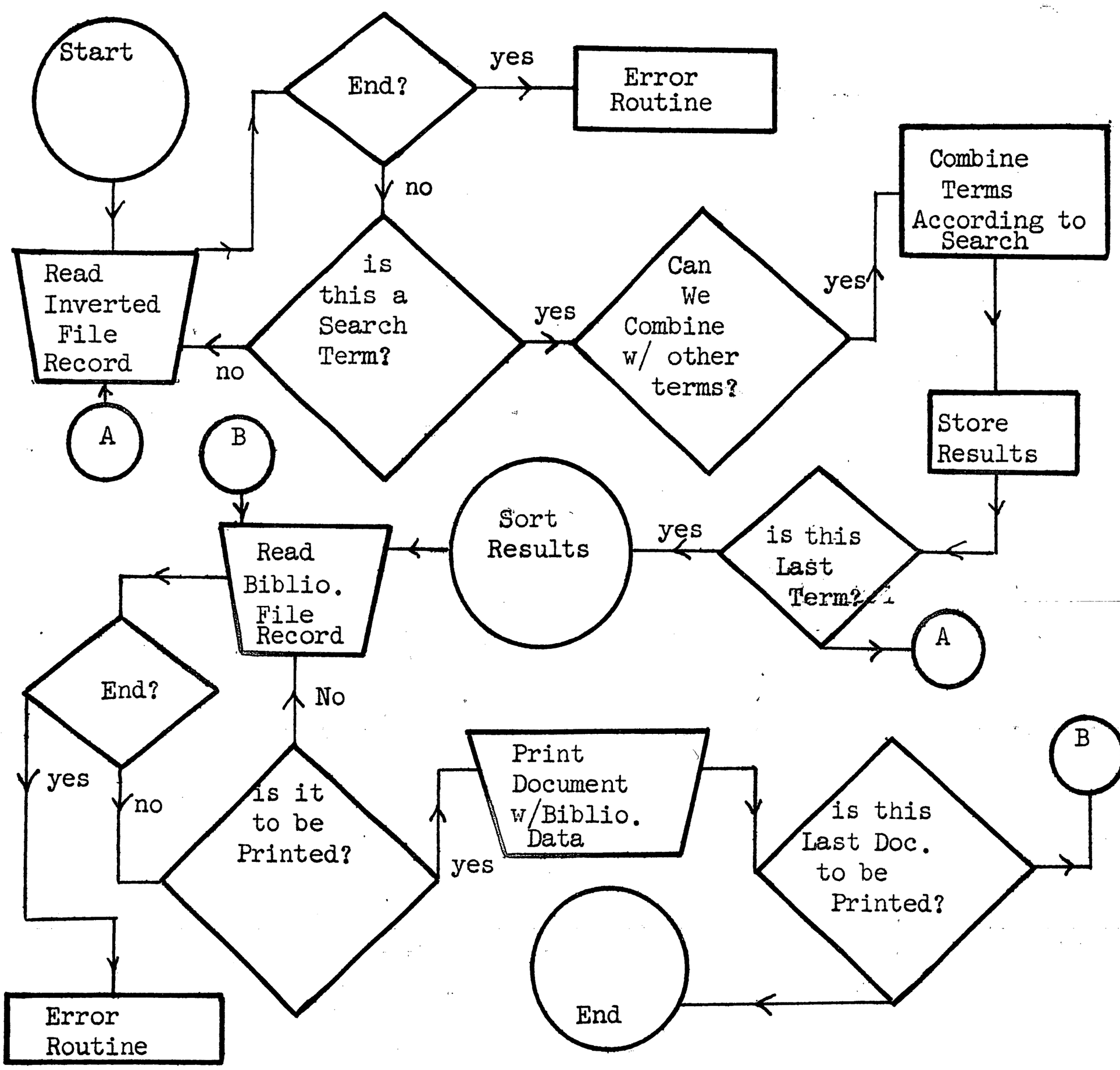


Figure 8.

Macro-Flowcharts for Two File Organizations

12 minutes while the serial scheme required only 8.5 minutes.

These tests are included in the following tabulation:

No. of Resulting Documents	Associative?	Printout?	Organ.No.1 Time	Organ.No.2 Time
19	no	yes	4.5	
150	no	yes		4.3
30	no	yes		2.8
30	no	yes	7.1	
30	no	yes		3.4
	Average search time:		5.8	3.5
149	yes	no	12.0	
104	yes	no	9.6	
178	yes	no		8.5
78	yes	no		5.5
128	yes	no		6.9
34	yes	yes	12.4	
48	yes	yes	12.8	
	Average search time:		11.7	6.9

The improvement in running time is a direct result of changes made in the file organizations. These results substantiate the claim for searching a serial file in the average tape oriented retrieval application.

The experimental results described in this section may appear unconvincing since the files were short and were slightly different in content. However, they were the only 'real' files available.

Although no references in the literature were found, the possibility of simulating a file exists. The reasonableness of such a simulation may serve as a topic for future study.

THE DISC FILE ORGANIZATION

Direct Access Schemes and the GE 225 Disc Storage Unit

One restriction imposed by magnetic tape is the use of a serial searching scheme. Serial searching is characterized by having to transmit records from tape to the central processor which are not necessarily the records being searched for. Usually, this searching is accomplished by checking each record read to see if it contains the desired data. However, even if, for example, we knew the desired record was the 17th record on the tape, the first 16 would still have to transmitted to the central processor before reaching the desired record. Direct access schemes, on the other hand, are characterized by the fact that a desired record may be retrieved without having to read other records. Utilization of a disc makes direct access searching schemes possible.

The GE 225 disc storage unit is a standard random access unit with 16 discs which rotate about an axis. Each separate disc is divided into 96 frames, with each frame containing 64 tracks.

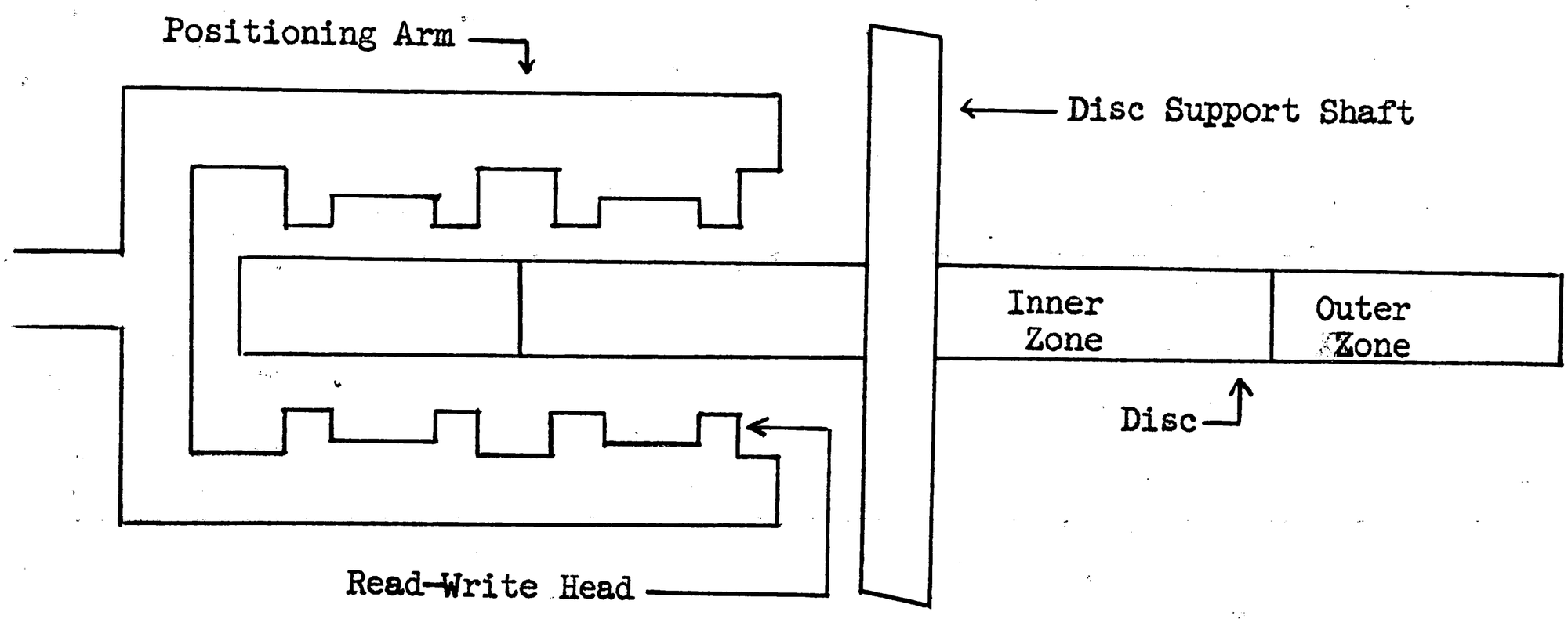


Figure 9

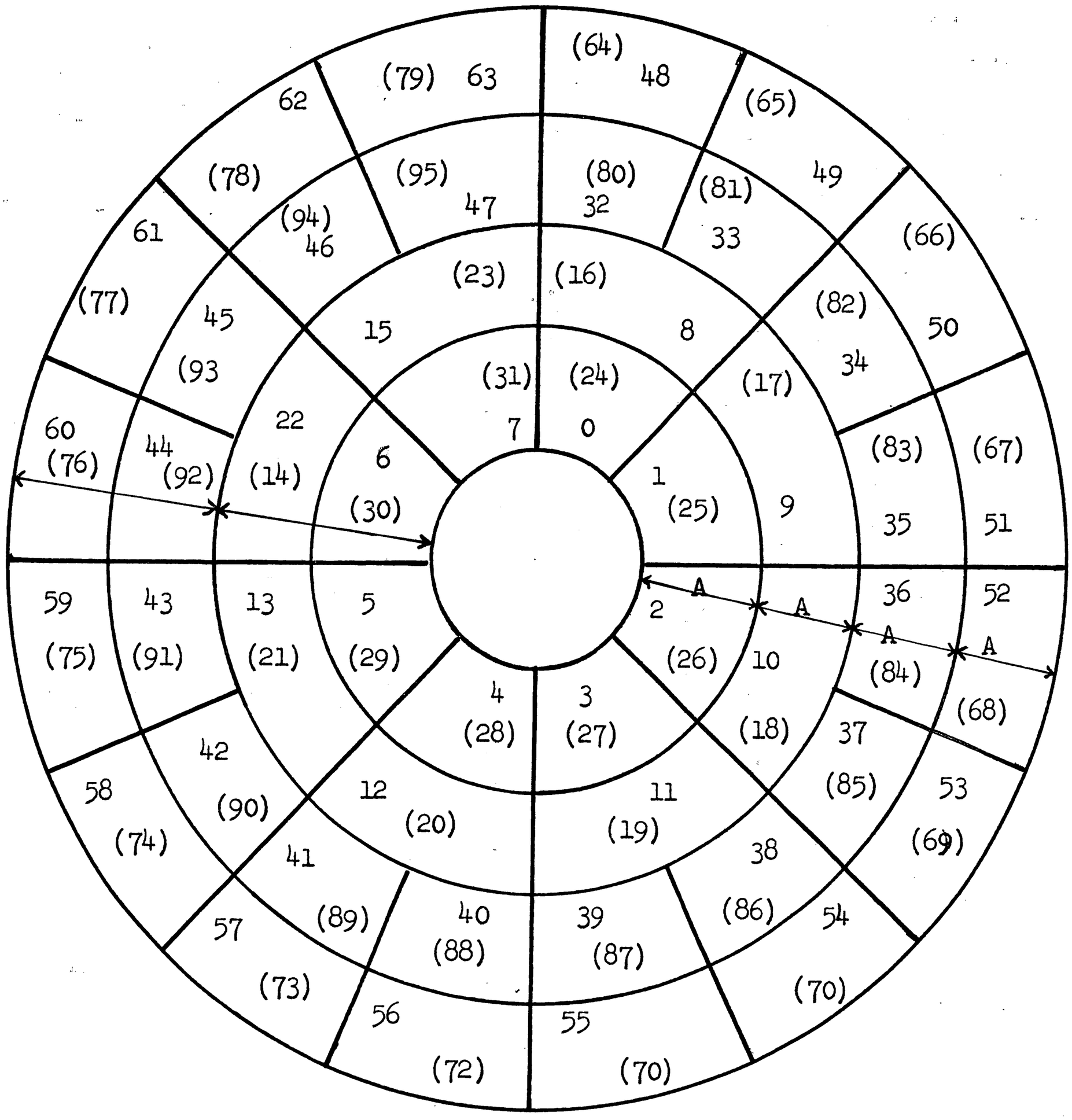
The Read and Write Heads

There are 64 computer words contained in one frame on one track. Figure 10 shows the frame numbering sequence for the storage surfaces of the disc.

Data is transmitted to or from the disc by one of 8 read-write heads. The positioning of these heads is shown in figure 9. Further discussion concerning the physical characteristics of the disc storage unit may be found in an article by Phillips (21) and in the GE Disc Manual (22).

To store or retrieve any information on the disc it is necessary to indicate on which disc (numbered 0-15), in which frame (numbered 0-95) and on which track (numbered 0-63) the information is stored. Naturally, if this information concerning location is unknown, we could search disc 0, frame 0, track 0; then disc 0, frame 1, track 0, etc., until we located the particular record for which we were searching. Previous reference was made to this method as serial searching. Effective use of a direct access device, however, necessitates the use of an addressing technique. This method uses a key associated with some data to compute the numbers necessary to store or retrieve that data on the disc. The key could be a name, code, etc., associated with the stored information. In the Center for the Information Sciences (CIS) system, the document number would be a good key for the serial file entries, while the term code number would be the key for the inverted file records.

Several obvious general rules apply to a scheme to compute a disc address based on a key. The scheme should be as simple as possible,



A = 64 Tracks

B = Inner Zone

C = Outer Zone

FRame numbers enclosed by parentheses are on the bottom side of the disc.

Figure 10

Frame Numbering Sequence for Disc Surface

easy and quick to compute, and should produce few, if any, duplicate disc addresses from different keys. See Buchholz (23) for further discussion of addressing techniques.

Illustration

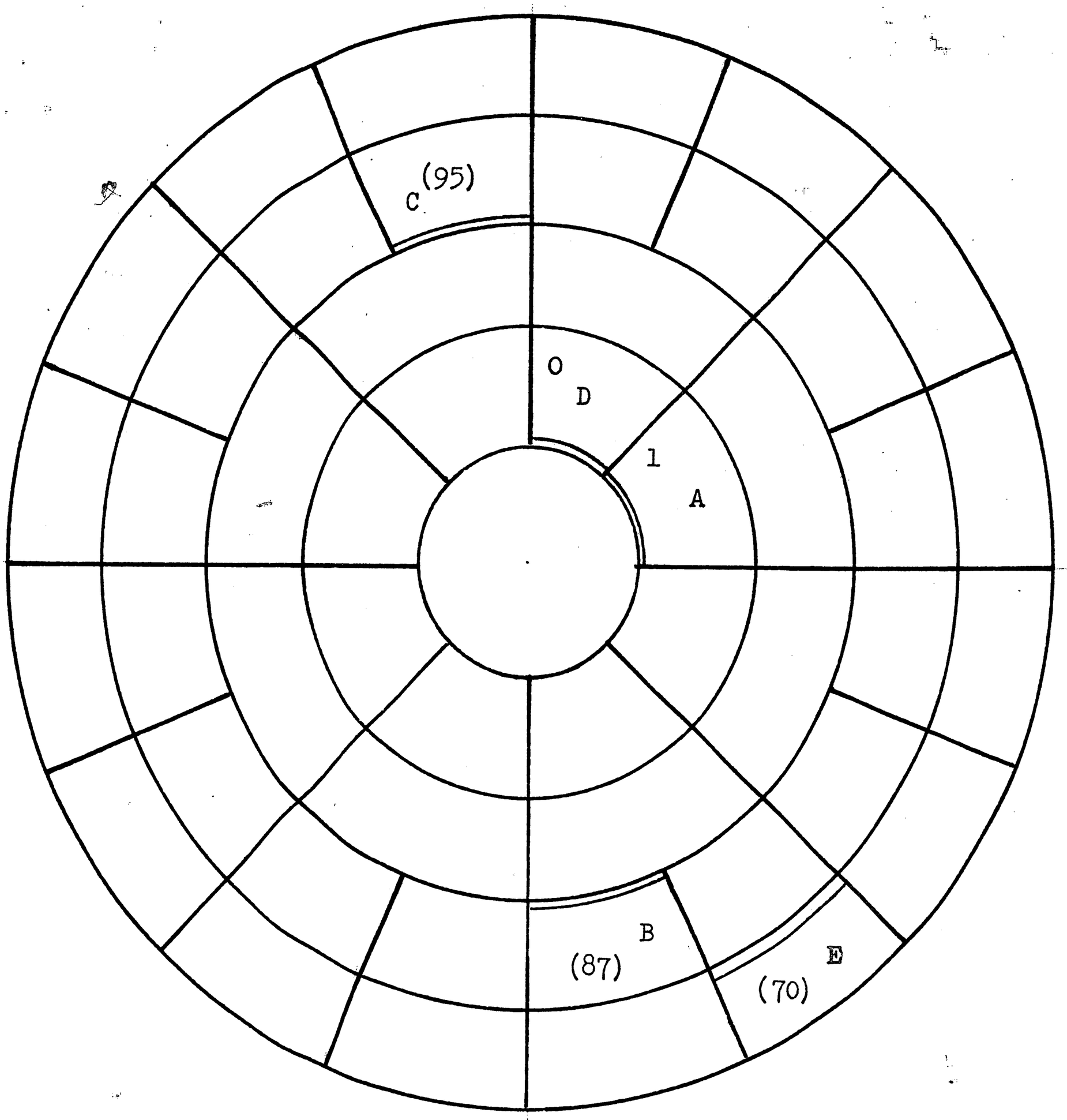
The general rules covering the record length in the discussion of magnetic tape file organizations again hold for the disc with one exception. The unit record length imposed by the disc itself is 64 words. Although the tape files of the CIS system utilized only every other word, the disc files will be 'packed'. Thus, all 64 words of a disc record become available for data, and record lengths should be whole multiples of 64 words.

The records of tape organization number 2 are multiples of 64 words. Since the serial file records are fixed length, one disc record is sufficient. In the inverted file, records are growing. Certainly a conservative record length could be chosen, and use of chaining techniques incorporated for long records. However, since space is available for quite lengthy records, it is best to distribute the inverted file records over one entire disc. Since there are 6144 records on 1 disc, eight 64 word records will be used for the storage of each of the 440 terms. This scheme allows for a total of 768 terms on one disc, with 512 words for each entry. The miscellaneous data will occupy 38 words, leaving 474 document entries before chaining becomes necessary. Naturally, the disc address of 'chained' records must lie outside the area of computable addresses.

The document numbers (keys) range from 1–1664. The scheme chosen to produce disc addresses is to divide the key by 96, i.e., the number of frames per disc. The quotient becomes the track number and the remainder becomes the frame number. Since all serial file entries will be contained on one disc, computation of the disc number will not be necessary. Some examples of the locations of documents are shown in figure 11.

Since eight locations are saved for each inverted file entry, the addressing scheme must produce only every eighth address. Here the term code number (key) is divided by 12, i.e., 96/8) and the quotient becomes the track number. The remainder multiplied by eight yields the frame number. Some examples of these addresses are shown in figure 12.

Each time the programs are to be run it will be necessary to load all the files onto the disc from tape. Naturally, it is desired that this transfer be as fast as possible. It should be noted that movement of the positioning arm shown in figure 9 is relatively time-consuming, as it is a partially mechanical device. The arm positions the read-write head over the proper track on the disc. Since there are eight heads per disc, (4 for the inner, 4 for the outer zone) and the inner zone contains 8 frames per track, the outer zone 16 frames per track, it becomes possible to write a total of 96 frames without changing the arm position (i.e., to write in a fixed track, in all 96 frames). Both the addressing schemes described above have the following property: when loading the disc from tape, if items are loaded sequentially by key, then a minimum of arm positioning is required. For



A = Document No.1, track 0, frame 1.

B = Document No.87, track 0, frame 87.

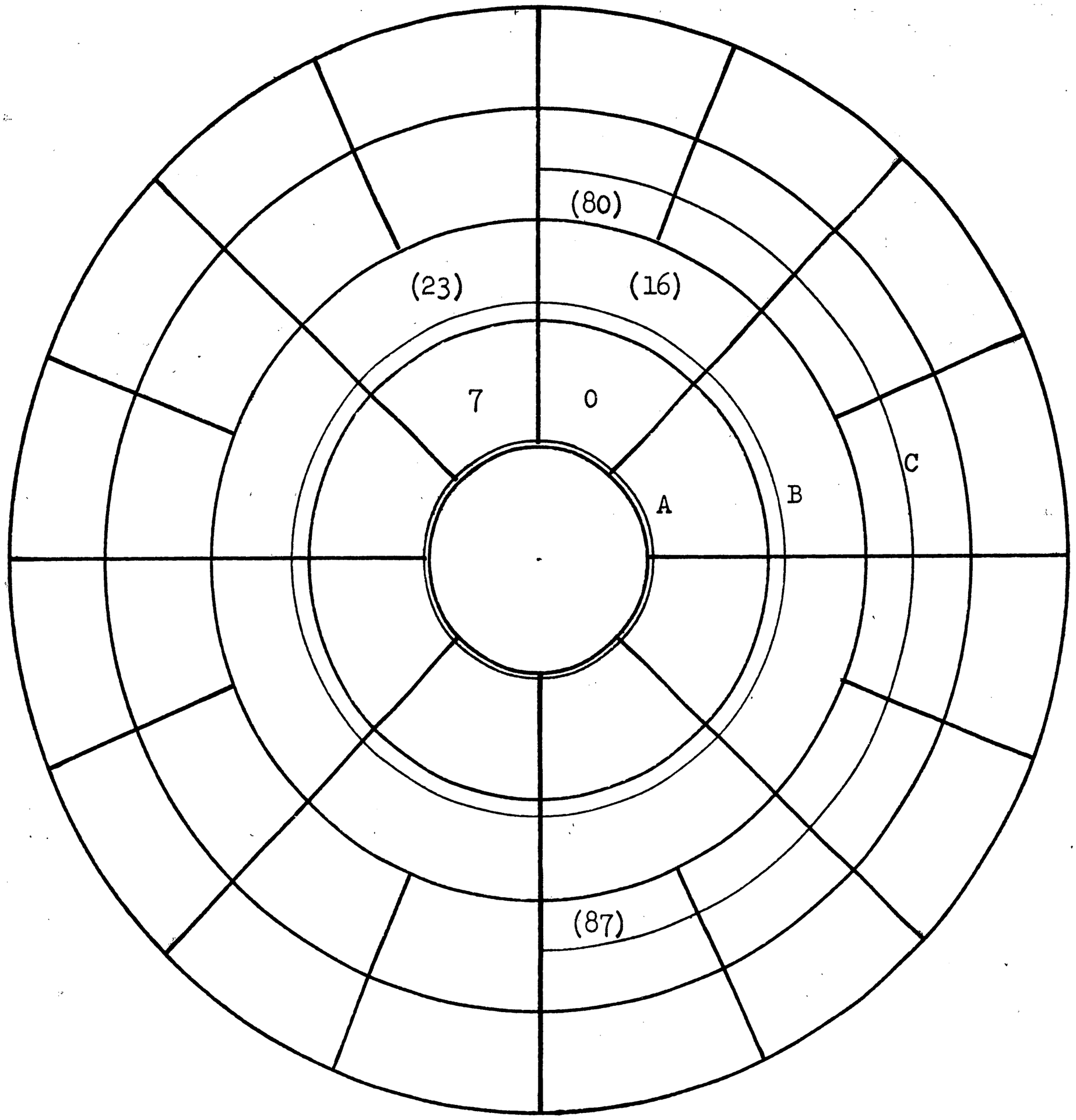
C = Document No.95, track 0, frame 95.

D = Document No.96, track 1, frame 0.

E = Document No.742, track 7, frame 70.

Figure No. 11

Example of Serial Record Locations



A = Term No. 0, track 0, frames 0–7.

B = Term No. 50, track 4, frames 16–23.

C = Term No. 356, track 28, frames 80–87.

Figure No. 12

Examples of Inverted Record Locations

example, in the serial file case documents 1–95 are written in frames 1–95, all in track 0. This writing requires no movement of the positioning arm.

A similar technique can be useful in chaining. Since there are 16 arms (one for each disc) and they all move together, chaining could be to a record on the same track, in the same frame but on a different disc than the record from which it was chained. Thus, when going from one record to the next in the chain, no movement of the arm is required.

A one–time operation will be necessary to initially construct the files on the disc. This will be accomplished by first reading all serial file records from tape, computing their addresses and storing them in the proper locations on the disc. Similarly the inverted file records are placed on the disc. Before storage, however, all data must be packed so that every word location is used. Once all files have been stored properly, a standard disc–to–tape routine can create the tape loader.

The General Assembly Program (GAP) instructions which control the disc operations require that all data concerning the disc, frame and track numbers be in a single memory word as follows:

Bits 2–5 : disc number
Bits 6–11 : track number
Bits 12–18 : frame number

For example, suppose we had the document number 742 in symbolic location DOC and we wish to retrieve it from the serial file on the disc. Assume that the serial file was loaded onto disc 3, and this information was previously entered into the program by the

following pseudo-instruction:

```
        SER     DEC     3     (Put decimal 3 in location SER)
```

Since this will appear in memory location SER as 0011 in bits 16-19, the following instructions will be necessary to shift to the proper bit locations for indication of the disc number,

```
        LDA     SER           (load SER into accumulator)
        SLA     14            (shift left A, 14 bits)
        STA     SER           (store into SER)
```

We may now compute the address of document 742 as follows:

```
NINSX   DEC     96            (put 96 in location NINSX)
        LDA     DOC           (load doc. number in accumulator)
        MAQ                   (put accumulator in Q register)
        DVD     NINSX         (divide by 96)
        SLA     8             (shift left 8 bits)
        ORY     SER           (put result in proper bits in SER)
        LAQ                   (put remainder in Q, into A)
        SLA     1             (shift left 1 bit)
        ORY     SER           (put result in proper bits in SER)
```

The contents of memory location SER will now contain the proper information to reference document 742, which is located on disc 3, frame 70, track 7.

Assuming the disc selector was on input-output channel 5, the following instructions would be required to read the record of document 742 into memory beginning at location START:

```
        BCS     BRN     5       (branch if disc not ready)
        BRU     *-1             (branch to above line)
        LDA     SER             (load SER into accumulator)
        STA     HERE            (store it in location HERE)
        SEL     5               (select channel 5)
        PRF     1               (position disc)
HERE    OCT     0               (positioning data stored here)
        BCS     BRN     5       (branch if disc not ready)
        BRU     *-1             (branch to above line)
        SEL     5               (select channel 5)
        RRF     1       1       (read disc, 1 frame)
                START           (read into location START)
        BCS     BRN     5       (branch if disc not ready)
        BRU     *-1             (branch to line above)
```

The general strategy when using the disc will be to retrieve search terms from the inverted file to determine those documents which satisfy the search, and to retrieve the bibliographic references from the serial file on the disc for printout.

Why will searching the inverted file be more efficient on the disc, when it wasn't with tape operations? In the section on tape organizations we showed that the philosophy behind first searching the inverted file in order to reduce the searching of the serial file was in general not a sound one, as the time involved in searching the inverted file was too great. However, with the disc file the time to retrieve a term is very short and the net saving (in being able

to go directly to the correct serial file entries) results in the shortest search time. A similar search strategy, using both files, is discussed by Warheit. (24).

The total time for retrieval on the disc, therefore, will be merely the time to look up the terms and the resulting documents. The average access time for the GE 225 disc unit is .225 seconds. With a 10,000 document collection a one term search will require one access for the term and an average of 65 accesses for the documents. The total access time will be approximately 15 seconds. In an application which requires quick response time the implementation of the above file organization on a disc will produce output fast enough to keep an output channel busy.

Conclusions

The automation of an information retrieval system requires that the system designer make many decisions involving the structure of the retrieval files. These decisions will have a direct effect on the operating efficiency of the system. Such items as block and record sizes are important.

The material presented in the first section indicates that for a magnetic tape oriented retrieval system, a serial file organization is more efficient than an inverted file organization, although neither is optimal. This conclusion is based on formulas derived from existing retrieval files. Experience with a small file has substantiated this result.

Discussion of disc file organizations in the second section showed that an inverted file scheme was most efficient for that storage medium. Although no disc was available at the time of this study, the implementation of the experimental file on a disc was outlined, and search time estimations were given. Future work on the Center for the Information Sciences Document Retrieval System will involve conversion of the retrieval files to the disc organization.

APPENDIX

Assume there exist x number of records in a random serial order. Then, if we want either one of the two previously specified records, we will have to search through $\frac{x+1}{3}$ of them, on the average.

PROOF:

We list the expected value of the number or records read at each stage, and sum the expected values.

Records Read	Expected Value
1	(1) $\frac{2}{x}$
2	(2) $\frac{x-2}{x}\ \frac{2}{x-1}$
3	(3) $\frac{x-2}{x}\ \frac{x-3}{x-1}\ \frac{2}{x-2}$
.....	.....
n	$(n)(2)\ \frac{(x-2)!}{(x-1-n)!}\ \frac{(x-n)!}{(x)!}$

$$= \frac{2n\ (x-n)}{x(x-1}$$

$$= \frac{2nx-2n^2}{x(x-1)}$$

Now sum from n=1 to n=x:

$$\frac{2nx}{x(x-1)} - \frac{2n^2}{x(x-1)} = \frac{x^2+x}{x-1} - \frac{x(x+1)(2x+1)}{3x(x-1)}$$

using the sum of the first n numbers = $\frac{n^2+n}{2}$

and the sum of the squares of the first n numbers = $\frac{n(n+1)(2n+1)}{6}$

This last result is:

$$\frac{3x^3 + 3x^2 - 2x^3 - 3x^2 - x}{3x(x-1)} = \frac{x+1}{3}$$

For the case of searching for any of three specified records out of $\underline{x}$ we use the same technique:

Records Read	Expected Value
1	(1) $\frac{3}{x}$
2	(2) $\frac{x-3}{x}\ \frac{3}{x-1}$
3	(3) $\frac{x-3}{x}\ \frac{x-4}{x-1}\ \frac{3}{x-2}$
.....	.....
n	$(n)(3)\frac{(x-3)!}{(x-n-2)!}\ \frac{(x-n)!}{(x)!}$

$$= 3n\ \frac{(x-n)(x-n-1)}{x(x-1)(x-2)}$$

Now sum from n=1 to n= x:

$$\frac{3n^2(1-2x)}{x(x-1)(x-2)} + \frac{3n(x^2-x)}{x(x-1)(x-2)} + \frac{3n^3}{x(x-1)(x-2)}$$

Using the sum of the cubes of the first $\underline{n}$ numbers = $\frac{n^2(n+1)^2}{4}$

we get:

$$\frac{3}{x(x-1)(x-2)}\ \frac{x(x-1)(2(1-2x)(2x+1) + 6x(x-1) + 3x(x+1))}{12} = \frac{x+1}{4}$$

If the general formula for the number of records to be searched when looking for any of t records out of a possible x total records is

$$\frac{(x+1)}{(t+1)}$$

then we can find the total number of records needed to be searched to find all of t records out of x:

$$\frac{x+1}{t+1} + \frac{\frac{x+1}{x-t+1+1}}{t} + \frac{\frac{2x+2}{x-t+1+1}}{t-1} + \cdots\cdots\cdots$$

$$= \frac{x+1}{t+1} + \frac{tx+x-x+1+t-1}{t(t+1)} + \frac{tx+x-2x+2+t+1}{(t+1)(t-1)} + \cdots\cdots\cdots$$

$$= \frac{x+1}{t+1} + \frac{x+1}{t+1} + \frac{x+1}{t+1} + \cdots\cdots\cdots$$

Since there will be t such factors the formula becomes:

$$\frac{t(x+1)}{t+1}$$

REFERENCES

1. Arthur D. Little, Inc., Centralization and Documentation, Second Edition, Cambridge, Massachusetts, 1964, p. 19.

2. Gregory, R., and R. Van Horn, Automatic Data Processing Systems - Principles and Procedures, Second Edition Wadsworth Publishing Co., Inc., Belmont, California, 1963, p. 626.

3. International Business Machines Corporation, General Information Manual, An Information Retrieval Experiment by Saint-Gaban Company, Paris, France, IBM Technical Publications Department, White Plains, New York, 1962, p. 10.

4. Swid, R., Linear vs. Inverted File Searching on Serial Access Machines, Automation and Scientific Communication, American Documentation Institute Annual Meeting, 1963, p. 170.

5. International Business Machines Corporation, An Information Retrieval Experiment.... op. cit., p. 10.

6. Miller, L., Rosen, S., and R. Swid, "Data Processing Techniques in an Information System", Proceedings of the American Documentation Institute, Vol. 1, Spartan Books, Washington, D.C., 1964, p. 149.

7. Ibid., p. 149.

8. Andrews, D., "Problems of Preparing an Inverted File", General Information Manual, Information Retrieval Systems Conference, September 21-23, IBM Technical Publications Department, White Plains, New York, 1959.

9. Whaley, F., "The Manipulation of Non-Conventional Indexing Systems", American Documentation, Vol. 12 (2) 1961, p. 101.

10. Kent, A., and J. Perry, "Operational Division of Files", Tools for Machine Literature Searching, Interscience Publishers, New York, N.Y., 1958, p. 482.

11. Goffman, W., and G. Badger, "An Experiment with File Ordering for Information Retrieval", Proceedings of the American Documentation Institute, Vol. 1, Spartan Books, Washington, D.C., 1964, p. 379.

12. Swid, R., "Linear vs. Inverted...." op. cit., p. 169.

13. Vickery, B., On Retrieval System Theory, Butterworth and Co., London, England, 1961, p. 81.

14. Arthur D. Little, Inc., op. cit., p.13.

15. Wall, E., "Further Implications of the Distribution of Index Term Usage", Proceedings of the American Documentation Institute, Vol.1, Spartan Books, Washington, D.C., 1964, p. 457.

16. Ibid., p. 463.

17. Arthur D. Little, Inc., op. cit., p. 21.

18. Vickery, B., op. cit., p. 81.

19. Libby, R., "Automated Storage and Access of Bibliographic Information for Libraries", Libraries and Automation, Library of Congress, Washington, D.C., 1964, p. 67.

20. Curtice, R. and V. Rosenberg, Optimizing Retrieval Results with Man-Machine Interaction, Center for the Information Sciences, Lehigh University, Bethlehem, Pennsylvania, 1965.

21. Phillips, J., "Disc Files in a Military Supply Depot", Disc File Applications, American Data Processing Inc., Detroit, Michigan, 1964.

22. General Electric Corporation, GE Compatibles/200 Disc Storage Unit Subsytem, Pheonix, Arizona, 1964.

23. Buchholz, W., "File Organization and Addressing" IBM Systems Journal, Vol. 2, 1963.

24. Prentice, D., deGraw, G., Smith, A., and A. Warheit, 1401 Information Storage and Retrieval System,(The Combined File Search System), Version II, International Business Machines Corporation, 1401 General Program Library, 1965, p. 1.01.

Vita

Robert Morris Curtice was born in Englewood, New Jersey, on November 19, 1943, the son of Forence and Edward Curtice of Fort Lee, New Jersey. He was graduated from Fort Lee High School in 1960 and received a B.A. in mathematics from Lehigh University in 1964. He is married to the former Joan Molaro of Fort Lee. He is co-author of:

Curitce, R., and V. Rosenberg, Optimizing Retrieval Results with Man-Machine Interaction, Center for the Information Sciences, Lehigh University, February 1965.