OSR 67-0423

AD647196

BEST AVAILABLE COPY

D D C
FEB 8 1967
B

UNIVERSITY of PENNSYLVANIA

The Moore School of Electrical Engineering

PHILADELPHIA, PENNSYLVANIA 19104

Distribution of this document is unlimited.

[redacted] 20040831037

University of Pennsylvania
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING
Philadelphia, Pennsylvania

Technical Report

DESIGN PRINCIPLES FOR AN ON-LINE INFORMATION RETRIEVAL SYSTEM

Prepared for

Air Force Office of Scientific Research
Office of Aerospace Research
Washington, D.C.
Contract No.: AF 49(638)-1421

and

Army Research Office-Durham
Durham, North Carolina
Contract No.: DA-31-124-ARO-D-352

December 1966

The following report was prepared by

Thomas C. Lowe

Morris Rubinoff
Principal Investigator

APPROVED:

John G. Brainerd, Director
The Moore School of
Electrical Engineering

Moore School Report No. 67-14
Distribution of this document is unlimit

INDEX

access time	5, 10
accession number	3
ACM Repository	8, 15
added information codes	1
background computation	80, 81
balanced tree	45
batch processing	6
bibliographic elements	1
Bowlden, Henry J.	38
bucket	5, 24
card format	118
character	4
Cheydleur, Benjamin F.	41
column 1 codes	119
communications	73, 114
complementation	71
concatenation	72
concept descriptors	1
controlled scan	39
data file	3
Dataphone	80
decoding	38
deconcatenation	65, 109
directed scan	39
dribble posting	9

editing 81

executive program 113

FORTRAN 94

Fredkin, E. 41

index item 4, 64, 65

index term 64

intersection 71

inverted file 23

inverted list 23, 59, 67

key to address 41

transformation

Landauer, Walter I. 47

linear file 7, 36, 76, 110

linked list 9, 36

magnetic disk 4

magnetic tape 8

Mandlebrot's relation 15

man-machine communications 78

MAP 94

memory 4

memory utilization 24

message editing 80, 86

normalized response time 15

packing 23

paper tape 80, 85

Peterson, W. W. 41

Polish suffix notation	109
practice mode	80
program interface	94
Prywes, Noah S.	59
rank of items	15
record	5
remote console	1, 80
retrieval process	106
retrieval request	69, 97
rooted scan	39
Rubinoff, Morris	42
scan	39
Scidmore, A.K.	41
sector	65, 98, 120
segment	74
Shannon, C.E.	18
Storage and Retrieval System (SRS)	56, 59, 62
Sussenguth, Edward H., Jr.	40
symbol tree	40
Teletype	70, 80
temporary list	76
triplet search	42
truncation	48
union	71
updating	6, 9
Weinberg, B. L.	39

word 4

Zipf's law 15, 28, 34, 37

CONTENTS

1.	Introduction	1
2.1.	Data File Organization	3
2.2.	Organization of Slow Memory Data Storage	7
2.2.1.	The Linear File	7
2.2.2.	Linked List Organization on a Document Basis	9
2.2.3.	Inverted File Organization	23
2.2.4.	Illustration of Results on Inverted Files	34
2.2.5.	Conclusions on Slow Memory Data Organization	36
2.3.	Decoding to a Slow Memory Address	38
2.3.1.	Symbol Trees	40
2.3.2.	Key to Address Transformation	41
2.3.3.	Triplet Searching	42
2.3.4.	Balanced Trees	47
2.3.5.	Effects of Truncating Index Items	48
2.3.6.	The Modified SRS System	56
2.3.7.	Conclusions on the Decoding Mechanism	62
3.1.	Data List Structure and Operators	64
3.2.	Generation and Format of Inverted Lists	67
3.3.	Operations on Inverted Lists	69
3.3.1.	List Union	71
3.3.2.	List Intersection	71

3.3.3.	List Complementation	71
3.3.4.	List Concatenation	72
3.4.	List Storage and Retrieval	74
3.5.	List Types	75
4.1.	Provisions for User-Computer Communications	78
4.1.1.	Equipment Emplosyed	78
4.1.2.	Practice Mode, Paper Tape and Background Programs	80
4.2.	Communications from the User's Viewpoint	81
4.2.1.	The Input Alphabet and Short Editing	84
4.2.2.	Full Editing	87
4.2.3.	Examples of Editing	90
4.3.	Interface with the Retrieval Programs	94
5.1.	The Information System	97
5.2.	The Retrieval Request	97
5.3.	Options on Information Returned	99
5.4.	Examples of System Operation	102
5.5.	The Retrieval Process	106
5.6.	Data Return	110
5.7.	The Executive	113
6.	Conclusions and Suggestions for Future Work	116
	Appendix	118

ILLUSTRATIONS

Linked List Organization	11
Record 2 Expanded	12
Ratio of Total to Distinct Items	16
Frequency-Rank	17
Normalized Frequency	20
R-Ratio	22
Inverted List Memory Requirements	25
Slow Memory Response Time When p(j) is Uniform	29
Memory Utilization P	30
Slow Memory Respnse When f(j) and p(j) follow Zipf's Law	35
Truncation for K=4	50
Effect of Truncation on Item List Length	52
Schematic Illustration of Slow Memory Record Linking	54
Average Number of Slow Memory Accesses	57
Slow Memory Accesses	58
Memory Organization	61
Features of Decoding Mechanisms	63
Equipment Configuration	79
Editing Examples - Part 1	91
Editing Examples - Part 2	92
Editing Examples - Part 3	93
Schematic Illustration of the Retrieval Request	101

Operation of Information System - Part 1 103

Operation of Information System - Part 2 104

Operation of Information System - Part 3 105

Operation of Information System - Part 4 107

Operation of Information System - Part 5 108

BIBLIOGRAPHY

1. Bowlden, Henry J.: A list-type storage technique for alphameric information, Comm. ACM, 6:433, 1963.
2. Brillouin, Leon: Science and Information Theory, p. 42, New York: Academic Press, 1956.
3. Brooks, Fredric P., Jr., and Iverson, Kenneth E.: Automatic Data Processing, New York: John Wiley and Sons, 1963.
4. Cheydleur, Benjamin F.: ACM Repository, Computing Reviews, 6:437, 1965.
5. Cheydleur, Benjamin F.: Dimensioning in an associative memory, Vistas in information handling, Paul W. Howerton and David C. Weeks, ed., Vl. London: Cleaver-Hume Press, 1963.
6. Fischer, Stephen B.: An executive control system for information retrieval via a remote console, M.S. Thesis, The Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, 1966.
7. Fredkin, E.: Trie memory, Comm. ACM, 3:490, 1960.
8. Freedman, H.A.: A storage and retrieval system for real-time problem solving, Progress report on contract NOnr 551(48) between the University of Pennsylvania, The Moore School of Electrical Engineering, and the U.S. Navy Office of Naval Research. Philadelphia, 1965.
9. Hamblin, C.L.: Translation to and from Polish notation, Computer J., 5:210, 1962.
10. Head, Robert V.: Dribble posting a master file, Comm. ACM, 9:106, 1966.

11. Landauer, Walter I.: The balanced tree and its use in information retrieval, IEEE Trans. Electronic Computers, EC12:863, 1963.

12. Landauer, Walter I.: The tree as a stratagem for automatic information handling, Doctoral dissertation submitted to the University of Pennsylvania, Philadelphia, 1962.

13. Nonconventional technical information systems in current use, No. 3, National Science Foundation, Washington, D.C., 1963.

14. Peterson, W.W.: Addressing for random-access storage, IBM J. Research and Development, 2:130, 1957.

15. Prywes, N.S. and Gray, H.J.: The organization of a multilist-type associative memory, IEEE Trans. Electronic Computers, EC12:448, 1963.

16. Raver, N.: Remarks made at the Third Annual National Colloquium on Information Retrieval, Philadelphia, Pennsylvania, 13 May 1966. (Proceedings to be published.)

17. Rubinoff, Morris: Information system design for the information processing field. Proposal submitted by The Moore School of Electrical Engineering, University of Pennsylvania, to the U.S. Air Force Office of Scientific Research. Philadelphia, 1965.

18. Rubinoff, Morris and White, John F., Jr.: Description of cataloging and indexing system for the ACM Repository. Progress report on contract AF-49(638)-1421 between The Moore School of Electrical Engineering, University of Pennsylvania, and the U.S. Air Force Office of Scientific Research. Philadelphia, 1965.

19. Rubinoff, Morris and White, John F., Jr.: Establishment of the ACM Repository and the principles of the IR system applied to its operation, Comm. ACM, 8:595, 1965.

20. S.A. 520-1219 and unnumbered price list. Promotional literature, International Business Machines, Inc.

21. Sanders, Robert David: A search command structure for information retrieval via a remote console, M.S. Thesis, The Moore School of Electrical Engineering, University of Pennsylvania. Philadelphia, 1965.

22. Scidmore, A.K. and Weinberg, B.L.: Storage and search properties of a tree-organized memory system, Comm. ACM, 3:28, 1963.

23. Shannon, C.E.: Prediction and entropy in printed English, Bell System Technical J., 30:50, 1951.

24. Stewart, Joseph Robert: The use of abbreviated tables for addressing random access storage, M.S. Thesis, The Moore School of Electrical Engineering, University of Pennsylvania. Philadelphia, 1964.

25. Sussenguth, Edward H., Jr.: Use of tree structures for processing files, Comm. ACM, 6:227, 1963.

26. Zipf, George Kingsley: Human behavior and the principle of least effort. Cambridge, Mass.: Addison-Wesley Press, 1949.

1. Introduction

This dissertation presents the results of an investigation of certain criteria involved in the development of document-oriented information retrieval systems, and the mechanization of a primitive experimental system for test and evaluation.

The information system considered is one by which the user of the system may, with no other human intervention, retrieve information using a reactive typewriter as an input and display console, and common carrier telephone lines for communication with the system. The system is designed for a conversational mode of operation, in which the user receives results from his queries almost immediately after presenting them to the system. This rapid interaction between man and machine allows the user to formulate a retrieval request, inspect the results of the search specified by that request, and, if necessary, enter successive refinements of the original request. Thus, operating in real time, the user may gradually sharpen the requests that supply the desired information.

It is desirable that the user of the system be able to retrieve information with a minimum of concern about procedural detail. Further, it is anticipated that he will not be an expert typist and may make typing errors. To establish a hospitable environment for such man-machine communications the user is aided by cues provided by the system. Also, extensive editing features must be provided in order that the user may correct his errors in messages addressed to the information system.

The retrieval request specifies a set of documents in terms of conventional bibliographic elements, concept descriptors, and special added information codes*. Since the user of the system is expected to compose retrieval requests as he works at a remote console, the format of the

* The added information codes are described in references 18 and 19.

retrieval request must be rather simple. Further, it is likely that a user may know only part of a word phrase (such as part of a title), and the system must possess the ability to accomplish retrieval using such partial information.

Another requirement concerns the return of information retrieved. Conventional bibliographic data, concept descriptors, and added information are available, but a user does not necessarily want to retrieve all these data. Indeed, with typical speeds of reactive typewriter operation of ten characters per second, the user may definitely want to suppress much of the available information. It is therefore required that he be able to select the categories of information to be displayed after the execution of a retrieval request.

Because the system is an experimental one, intended for use in future information systems research, it is required that it be constructed in a modular manner so that additional features may be added to it and evaluated.

This paper describes first a study of data file organization and decoding mechanisms, the results of which are used to find a structure which is capable of retrieval in real time and is not wasteful of memory. Various lists and operations on lists necessary to the retrieval process are next defined. Following this is a description of the system features that make possible the man-machine communications, including the capabilities described above. The overall integration of these components, including a processor that transforms retrieval requests into operations on lists and executes these operations, is also described. Finally, examples are provided to illustrate the complete, working information retrieval system.

2.1. Data File Organization

In this chapter, several calculations are made in order to compare selected features of different file organization schemes. For some of these calculations, sample data are taken from the indexed documents forming the core collection of the Repository of the Association for Computing Machinery[4].

In the following sections of this chapter, memory organization will be treated under two separate headings. Section 2.3. considers the decoding problem, namely, finding a slow memory* address associated with a given index item. Since the decoding process does not occur in the case of a linear file, the latter is fully treated in Section 2.2.

In Section 2.2., it is assumed that some decoding mechanism exists and that the problem to be considered is that of the organization of the storage of accession numbers in slow memory along with the codes associated with index items. It is assumed that the decoding precessor discussed in Section 2.3. has supplied an entry designation into the slow memory structure.

In the system under consideration, each document is identified by an accession number and is characterized by index terms. The latter are strings of words such as authors' names, titles, publishers' names, dates, concept descriptors, and the like. Such strings are

---

* "Slow memory" refers to devices such as direct access mass storage and is described more fully below.

deconcatenated to form single-word index items.

All the characterizing information is stored in the computer system. Relevant accession numbers are retrieved upon execution of appropriate search commands, which specify a set of index items along with some specified relationships among the items.

In the present discussion of data file organization, two types of computer memory are assumed to be available--a limited amount of fast memory,such as ferrite core, and a much larger volume of slow memory,such as magnetic disk. The fast memory is commonly called "internal" to the computer. Depending on the computer organization, some organizational units of data may be fetched rapidly from fast memory into the processing registers. Indeed, in some machines the processing registers are structurally in fast memory. Even if all fast memory is not directly addressable because of a need for indirect or relative addressing or field indicators, it is still assumed to be addressable and accessible significantly faster than the slow or external memory. Indeed, "accessing" slow memory generally means the transfer of data from slow memory into fast memory, not into the processing registers of the computer.

In many machines, the processing registers are accessible to the programmer. The organizational unit of memory equal to the capacity of such a register will here be called a "word". Words typically range from eight to sixty-four bits in length. Another unit used in the following description is the "character", or group of bits representing a symbol from a selected alphabet. The number of bits used to represent

a character commonly ranges from five to eight. For the purposes of the present discussion, the terms "word" and "character" will be considered to be synonymous in the case of machines that perform only character-by-character processing. The term "record" will be used in a restricted sense to indicate a group of contiguous words chosen because of some relationship between the information conveyed by those words.

Generally, a slow memory access involves the transfer of a set of several words from slow to fast memory; such a set of words is called a bucket; will be used to designate the time required to transfer a bucket from slow to fast memory.

For example, consider the 32678-word IBM 7040 computer with the IBM 1301-2 disk memory unit at the University of Pennsylvania Computer Center. Measured in six-bit characters, the fast memory capacity is about 2.0×10^5 characters, while the slow memory holds 5.6×10^7 characters. The basic unit is a word of six characters. The time required to copy a word from fast memory into the accumulator of the computer is 16 microseconds. Times ranging from about 0.05 to 0.23 seconds are required to transfer a bucket of 2800 characters between slow and fast memory. The long access time for the IBM 1301 results from the latency associated with all rotating memory devices, and also the time required to position mechanically the movable reading and recording heads of the disk memory mechanism.

Owing to the cost of fast memory, economic considerations require that the bulk of the system's information be stored in slow

memory*.

In a batch processing system that is not operated in a real-time on-line environment, the overall turnaround time and the average time required for each query processed are quite different quantities. In a real-time on-line system in which man-machine dialogue is of major importance, each query is to be processed (and answered) as soon as possible after the user has formulated it, and the average system response time assumes great importance. A batch system may process several queries at once, resulting in a short average computation time per query. But in such a system the overall response time, or lag between query and response, may still be quite large.

In the system under consideration, the updating of files is to be performed infrequently; the data from recently indexed documents will be added to the files perhaps once a month. Therefore ease of file updating does not have the importance that it has in many other information retrieval and storage systems with rapidly changing files, such as inventory systems.

A further requirement in the implementation of a system for conversational operation is that the index items be permitted to be of variable length (although it is reasonable to assign an upper bound on the length of an index item).

---

* It must be kept in mind that this assumption is based on current commercially available memory devices and is subject to change as new devices and techniques become available. Based on purchase price and comparing the smallest available slow and fast memory increments for the IBM 360/44, the ratio of costs per character is about 600:1. Based on approximately equal capacity differences and comparing an IBM 360/44 model D with a 2314 disk and a 360/44 model H with no disk, the ratio is about 710:1. (20)

2.2 Organization of Slow Memory Data Storage

In general terms, the problem considered is one of organizing and manipulating data that describe the following: a function F which maps each element of a finite set I onto a non-empty subset of a finite set E. In the present document retrieval problem, I is the set of index items used to characterize documents. The accession numbers of the documents (combined with additional codes indicating the order of index items in multiple word phrases) constitute the set E. The function F is to be mechanized efficiently; that is, given the general characteristics of commercially available equipment, an acceptable scheme is to be found that will allow the retrieval of accession numbers associated with index items.

Using average response time and efficiency of slow memory utilization as criteria, different file organization schemes are compared below.

It is assumed where necessary that a decoding mechanism that maps elements of I one-to-one into slow memory addresses exists, and the elements of E may be uniquely identified by the use of a fixed-length field. In sections 2.2.2 and 2.2.3, design parameters are developed based on several different assumptions concerning both some characteristics of the function F and a probability function describing the use of the system.

2.2.1. The Linear File

This file organization has been used extensively[28] with batch processing systems using magnetic tape for slow memory. In it, one record of slow memory is associated with each document. Such records

include complete indexing information for the documents with which they are associated. In order to find the accession numbers of documents characterized by a given index item, the entire file must be searched. Although the linear file has the advantage that it is quite easily updated and in many applications may be implemented using relatively inexpensive magnetic tape units for slow memory, the response time tends to be much too large for use in a real-time system. This limitation becomes increasingly severe as the file grows larger.

Time required for retrieval from this file is proportional to the product of the number of documents in the file and the slow memory communication speed. In the ACM Repository, about 1200 characters are required to index a document - this figure including all the conventional bibliographic data, index terms, and the special added information, but excluding the abstract. Assuming a file organization for the IBM 1301 disk that allows very efficient packing and assuming reading of 2800 character buckets in 0.05 seconds (implying infrequent head motion), an overall response time of 0.02 seconds per document will result. Clearly, a situation in which a file consisting of only 1000 documents already takes 20 seconds to search would not be acceptable in the implementation of a man-machine dialogue.

Thus, with typical currently available equipment, if a small response time is required, linear file organization can only be used when the total number of documents is very small or when the total volume of information is low.

One advantage of the linear file is that file information needs to appear only once, whereas portions of the file must generally

be repeated when other types of files are employed. For example, in the course of finding the accession number of a document, the searcher may desire, say, the name of the author. In the case of a linear file, such information is easily extracted during the search process.

A second advantage of the linear file organization is that such a file may easily be updated, and if "dribble posting"[10] techniques are used a linear file need not be entirely regenerated every time data are added or changed. Dribble posting is a technique in which a small modification file is kept in addition to the master file, the master file being updated only when the modification file reaches some predetermined size. Since a short linear file may be searched in a relatively short time, a possible application of the linear file in a real-time system is the short term retention of small quantities of new material, awaiting the sufficient number of changes to justify complete reworking of a non-linear file system.

2.2.2. Linked List Organization on a Document Basis

The linked list organization[15] is an arrangement of data that, in this application, provides for the establishment of lists uniquely associated with index items. As the nodes of these lists are similarly associated with documents, tracing of any list implies the retrieval of documents associated with the index item of that list.

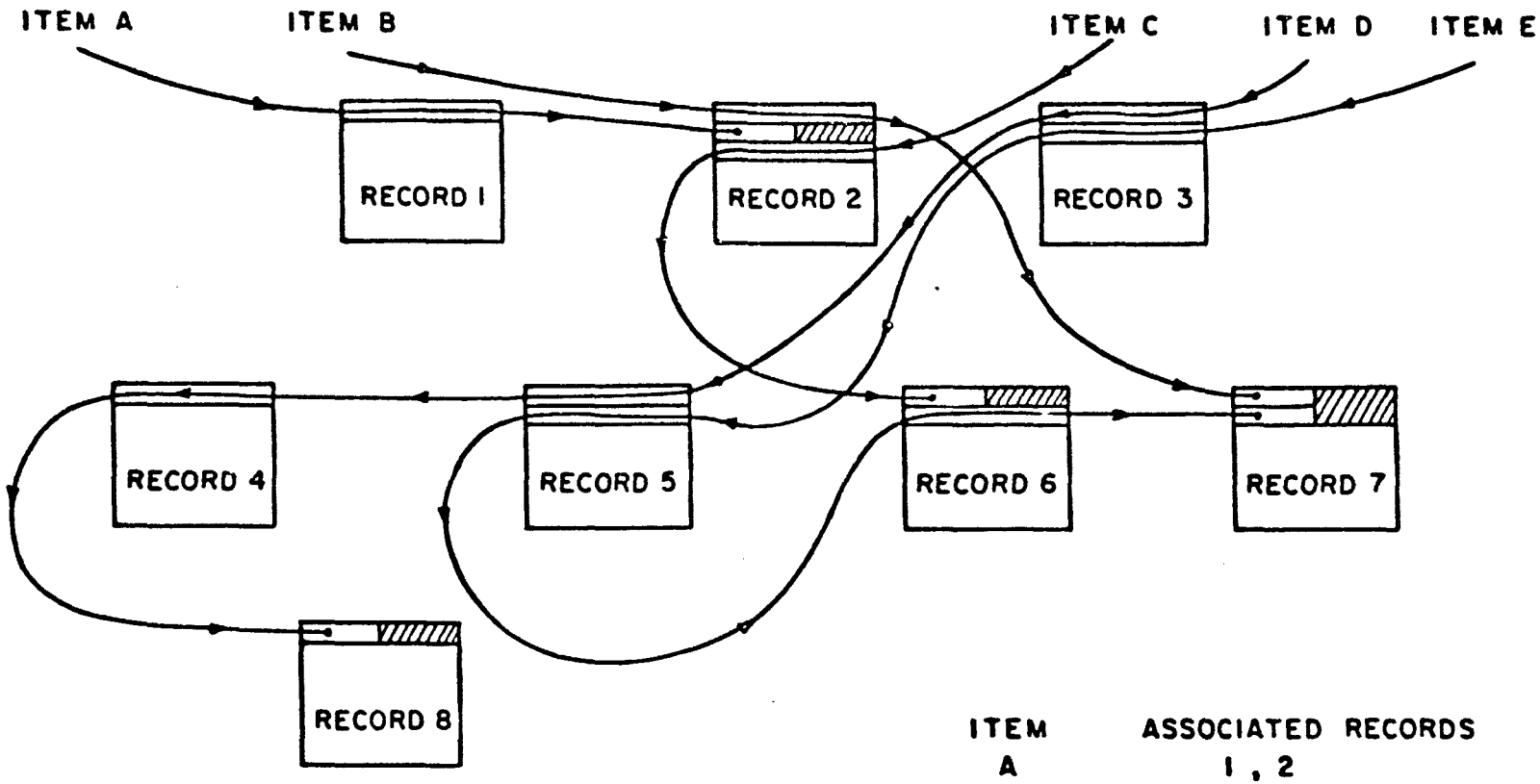
It is the purpose of this section to derive relationships which indicate some effects of file characteristics and file usage on retrieval time when a linked list organization of the type described below is used. These relationships will be used in a later section when the linked list is compared with other proposed file types.

Only the organization in slow memory, where the bulk of the information is to be stored, will be considered in this section. The decoding mechanism is one of those described in Section 2.3 and is assumed to exist.

Figure 1 shows the organization of slow memory into records with linkages. It is assumed that each bucket is large enough to accommodate one or more records. Every record corresponds to one document and to a node on a list; each list corresponds to one index item and all documents characterized by that index item are represented by nodes on that list. If a document is characterized by several index items, the document is represented by a node that is at the intersection of the lists corresponding to the several items.

Figure 2 illustrates schematically the structure of one record, record number 2 . This record pertains to a document that is characterized by items A, B, and C. Therefore the node corresponding to that document and represented by record 2 is on three lists, lists A, B, and C. In this case, the node marks the end of list A and immediately precedes record 7 on list B and record 6 on list C.

Before timing calculations can be made, a few parameters needed later in the discussion must be defined. Recall that t_s is the slow memory access time, and let the total number of distinct index items used to characterize the entire file be N. That is, the cardinality of the vocabulary of index items used to characterize the entire file is N. Further, let these distinct index items be temporarily numbered: $1,2,\ldots,j,\ldots,N$; if $f(j)$ is the total number of times that



ITEM	ASSOCIATED RECORDS
A	1, 2
B	2, 7
C	2, 6
D	3, 4, 5, 8
E	3, 5, 6, 7

FIGURE 1
LINKED LIST ORGANIZATION

ITEM B	ADDRESS OF RECORD 7 ITEM B LINKAGE
ITEM A	END OF LIST SYMBOL
ITEM C	ADDRESS OF RECORD 6 ITEM C LINKAGE
DATA, IN THIS CASE PRIMARILY ACCESSION NUMBER	

RECORD 2 EXPANDED VIEW

FIGURE 2

the index item numbered j is used in characterizing all the documents in the file, then f(j) documents are characterized by the jth index item. In the linked list structure, there are f(j) nodes on the list corresponding to the jth index item; it is therefore said that the jth list has length f(j).

The time required to retrieve all records on a list of length f(j) is $f(j)t_s$. Note that this is both the average and the maximum time required to retrieve all such items.

In the following calculations, it is also assumed that each record occupies not more than one bucket. If this were not the case, more than one slow memory access would be required for nodes represented by long records. A more complete analysis would take into account the effect of the bucket size, allowing for some records that might overflow a single bucket. This factor is neglected here because sufficient information for the present purpose may be obtained without considering it. However, the overflow problem is important to the file organization plan discussed in Section 2.2.3. and is taken up there.

The logical <u>and</u> and <u>and not</u> functions may be mechanized easily using this scheme; to find all records corresponding to documents characterized by index items A and B, the process simply follows one list (preferably the shorter), checking all items of all records on that list for an intersection with the second list. But again, the time required to search a single list of length f(j) is $f(j)t_s$.

Some formulas for the time required to retrieve all records on a list corresponding to a given item will be developed at this point. Let the total number of occurrences of all index items in

characterizing the entire file be S. Thus, while N is the total number of index items available (and used at least once), S is the total number of times that index items have been employed in the characterization of the file. It follows that

$$S = \sum_{j=1}^{N} f(j). \tag{1}$$

Now assume that a user enters a search command involving one of the N index items, and let p(j) be the probability that the jth item is specified in user-initiated searches. Then the average retrieval time per index item for the portion of the retrieval system described here is:

$$T_r = t_s \sum_{j=1}^{N} f(j)p(j). \tag{2}$$

The functions f(j) and p(j) may frequently be unknown or difficult to determine, depending on the particular information in the file, the method of indexing and cataloging that information, the suppression of very high- or low-frequency index items, and variable human factors. Three different assumptions about the functions f(j) and p(j) will be used in this analysis. Most combinations of f(j) and p(j) encountered in practice will, it is believed, lie somewhere between the extreme limits developed here.

In the first case, it is assumed that the frequency f(j) and the probability p(j) are uniformly distributed, and are constant for all j. Then f(j) = S/N, the number of items divided by the total

number of lists. Also, $p(j) = 1/N$, so that from (2)

$$T_r/t_s = \sum_{j=1}^{N} \frac{S}{N}\frac{1}{N}, \tag{3}$$

or

$$T_r/t_s = S/N \tag{4}$$

The ratio of T_r to t_s is a normalized response time. The ratio S/N will depend on the total number of documents indexed and the indexing and cataloging techniques. For the ACM Repository data, the gradual growth of the ratio as documents were cumulatively added to the collection is shown in Figure 3.

The distinct index items were previously identified by the assignment of a number j, $1 \leq j \leq N$, to each index item. Let the number now be reassigned so that j is also the rank of the jth index item, the item most frequently used being designated as item one, etc. In the event two or more index items have equal ranks, they are to be arbitrarily assigned consecutive numbers. The frequency function f(j) for the ACM Repository collection is shown in Figure 4.

Various qualitative rank-probability relationships have been proposed for spoken and printed communications. Zipf's law[26] is perhaps the best known of these; others, such as Mandelbrot's relation[2] have also been proposed. As originally stated, Zipf's law applies to printed English text and gives $p(j) = 0.1/j$. Persons working with the use of descriptors in indexing files have found empirical justification in the use of Zipf's law[16], but with a constant different from one tenth, $p(f) = k/j$, because of variations in the number of distinct

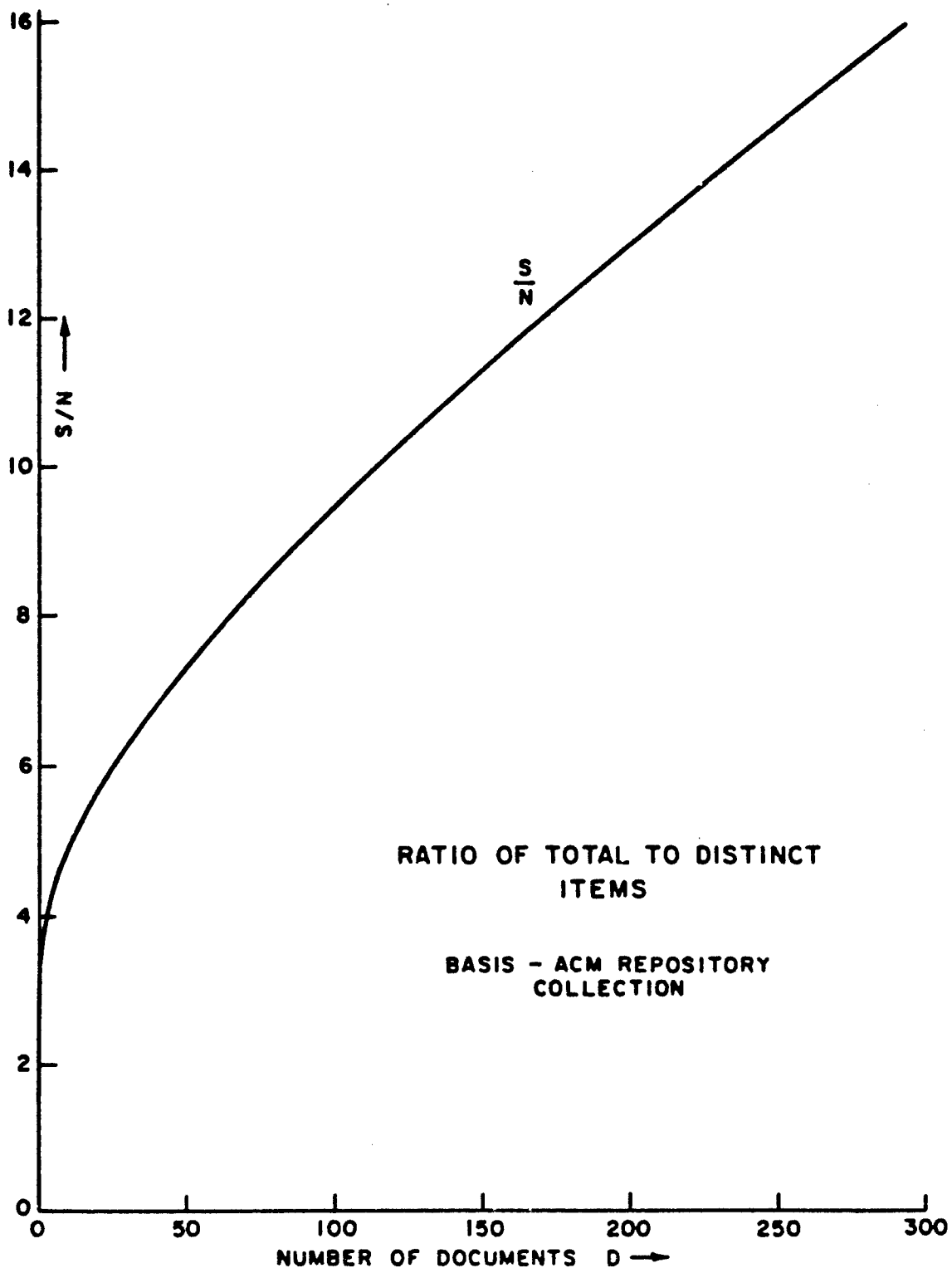

16
14
12
10
8
6
4
2
0
S/N
S/N
RATIO OF TOTAL TO DISTINCT ITEMS
BASIS - ACM REPOSITORY COLLECTION
0
50
100
150
200
250
300
NUMBER OF DOCUMENTS D

FIGURE 3

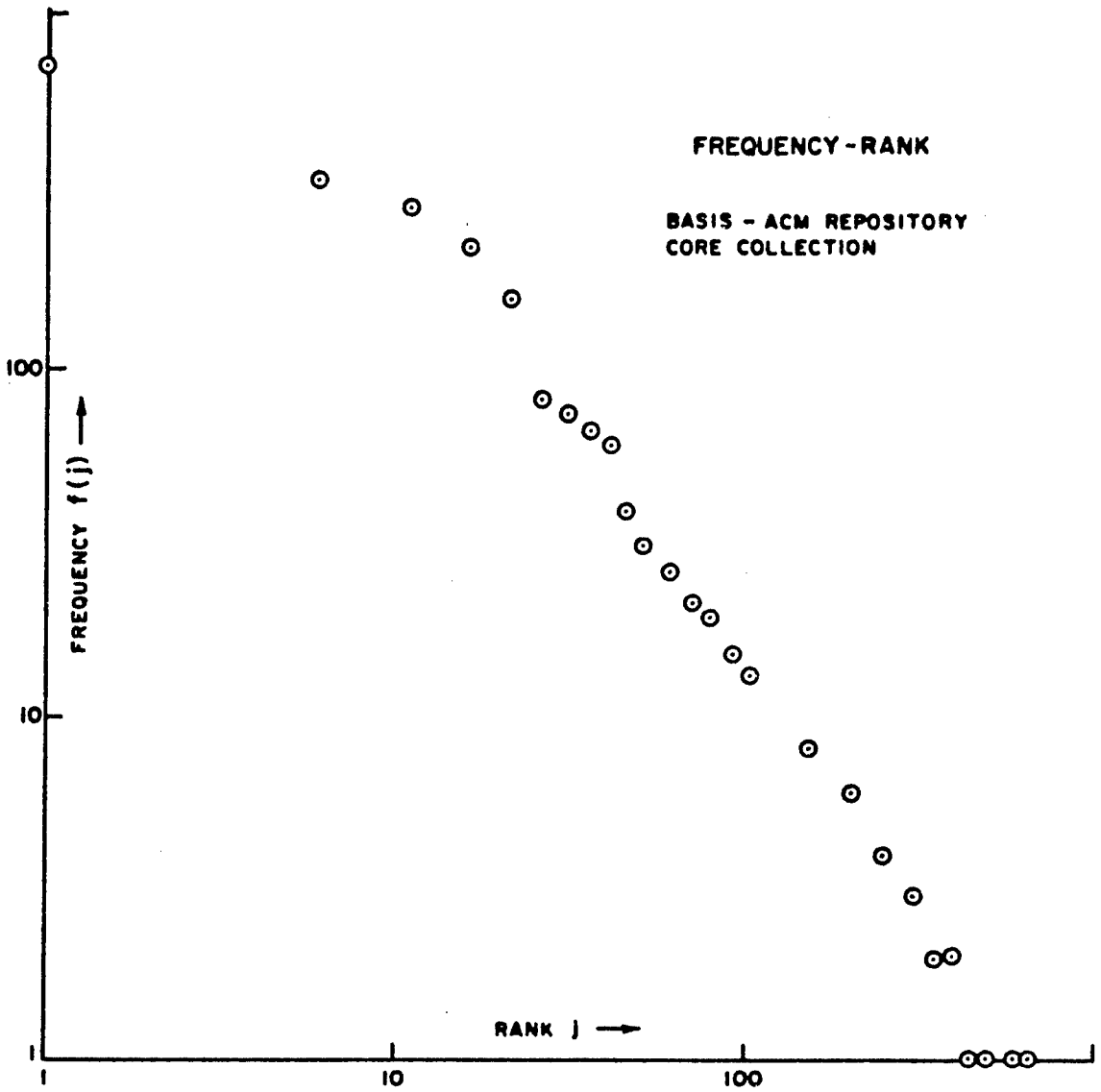

FREQUENCY - RANK
BASIS - ACM REPOSITORY
CORE COLLECTION
FREQUENCY f(j)
100
10
1
RANK j
1
10
100

FIGURE 4

items, N.

In order to obtain broad criteria for various types of files, three different sets of assumptions will be made concerning the rank-frequency characteristics of the occurrence of items in the data file and the rank-probability characteristics of item selection in the queries to the system. The first assumption, that f(j) and p(j) are uniform and constant, has already been illustrated. The second assumption is uniform p(j) and a frequency function f(j) following Zipf's law; the third assumption is that both f(j) and p(j) exhibit the behavior predicted by Zipf.

Now if Zipf's law is taken as originally stated, the sum of the probability of appearance of all words in a vocabulary is unity only when the vocabulary consists of 8727 words. In considering printed texts, Shannon[23] has elected to retain Zipf's constant of k = 0.1, and consider only the first-ranked 8727 items, arbitrarily setting p(j) = 0 for all j greater than 8727.

In the present case, such free text is not being considered. The number of words in the vocabulary is known (N). For the non-text situation, Zipf's value of one tenth for k is suspect. A modified law is easily obtained by calculating k subject to the constraints that

$$\sum_{j=1}^{N} f(j) = S \qquad \text{and} \qquad \sum_{j=1}^{N} p(j) = 1.$$

Recalling equation (1) and applying the approximation (good for large N, with a diminishing error which is about 2% at N = 10, and 0.06% at

N = 100)

$$\sum_{j=1}^{N} \frac{1}{j} \doteq \ln N + \gamma \tag{5}$$

gives

$$f(j) = \frac{S}{j(\ln N + \gamma)} , \tag{6}$$

where the symbol γ is used for Euler's constant, about 0.5772. This is the rank-frequency relationship when it is assumed that f(j) follows the form of Zipf's law. Since the probabilities must sum to unity, it also follows that the form for Zipf probability is

$$p(j) = \frac{1}{j(\ln N + \gamma)} \tag{7}$$

Consider now the second pair of assumptions, in which f(j) is Zipf and p(j) uniform. Figure 5 shows a normalized plot of f(j) for points obtained from the ACM Repository data.

For this second case, application of equation (2) yields

$$T_r = t_s \sum_{j=1}^{N} \left(\frac{S}{\ln N + \gamma} \frac{1}{j} \frac{1}{N} \right) , \tag{8}$$

or

$$T_r/t_s = S/N \tag{4}$$

Thus, the average response time is the same in both cases considered so far. Indeed, this must always be the case whenever p(j) is constant,

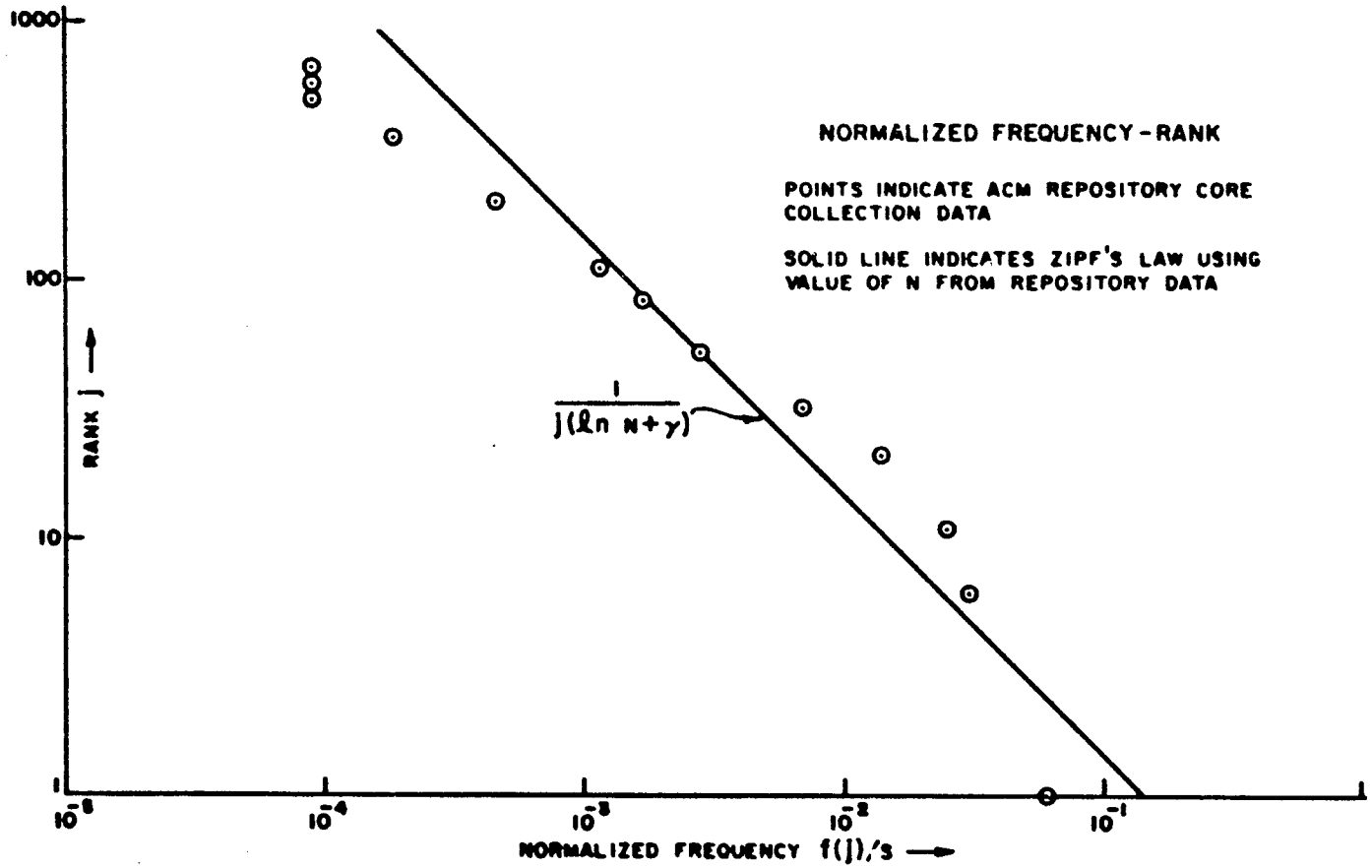

NORMALIZED FREQUENCY-RANK
POINTS INDICATE ACM REPOSITORY CORE COLLECTION DATA
SOLID LINE INDICATES ZIPF'S LAW USING VALUE OF N FROM REPOSITORY DATA
$\frac{1}{j(\ln N+\gamma)}$
RANK j
1000
100
10
1
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
NORMALIZED FREQUENCY f(j),'s

FIGURE 5

the sum of p(j) over all j is unity and of f(j) is S.

In the third case, where the inverse proportionality of Zipf's law holds for both f(j) and p(j), substitution of (6) and (7) into (2) gives

$$T_r = \frac{S\, t_s}{(\ln N + \gamma)^2} \sum_{j=1}^{N} \frac{1}{j^2} \quad . \tag{9}$$

Since the series 1 + 1/4 + 1/9 + 1/16 + ... converges rapidly and N is large, the value of the summation may be taken as the sum of the infinite series, yielding:

$$T_r/t_s = \frac{\pi^2 S}{6(\ln N + \gamma)^2} \quad . \tag{10}$$

The error caused by replacing $\sum_{j=1}^{N} \frac{1}{j^2}$ by $\frac{\pi^2}{6}$ is about 2% when N=30, 0.6% when N=100.

The ratio of the two values of T_r/t_s obtained when p(j) is uniform and when p(j) follows Zipf's law is given by

$$R = \frac{N\,\pi^2}{6(\ln N + \gamma)^2} \quad , \tag{11}$$

and is shown in Figure 6.

Thus, inverse-proportional behavior of p(j) can greatly increase response time. In an actual situation, p(j) will have a characteristic between the uniform and Zipf case. Hence, the response time will be something between the uniform value given by (4) and that value multiplied by R.

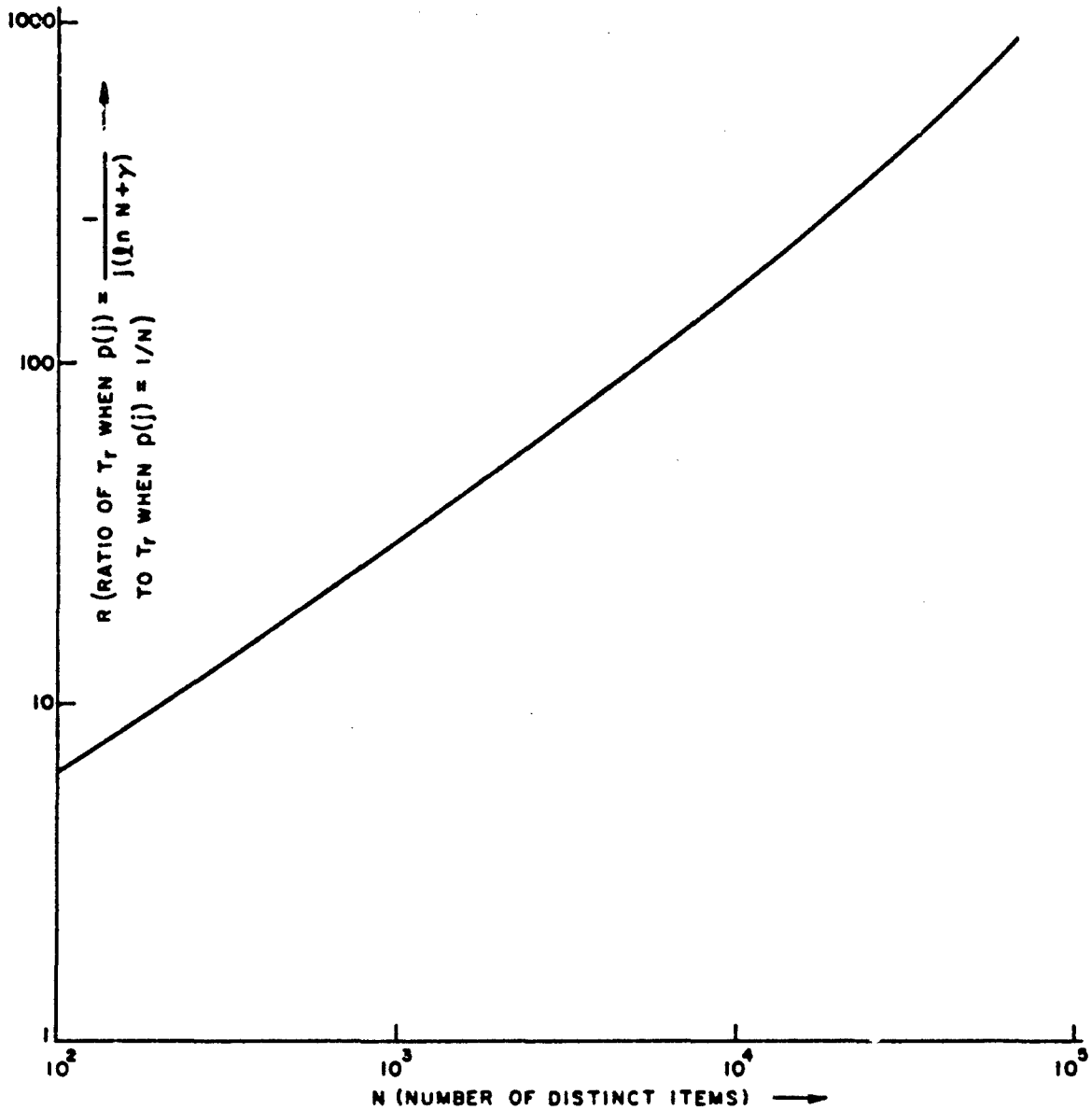

1000
100
10
1
R (RATIO OF T_r WHEN $p(j) = \frac{1}{j(\ln N + \gamma)}$ TO T_r WHEN $p(j) = 1/N$)
10^2
10^3
10^4
10^5
N (NUMBER OF DISTINCT ITEMS)

FIGURE 6

To a first approximation, the memory requirements of a linked list exceed those of a linear file only by the requirements for the linkages. Including list termination symbols, there are S such linkages.

2.2.3. Inverted File Organization

In inverted file organization every record is associated with an index item, and contains all accession numbers (and any needed additional codes) of documents characterized by the index item. The purpose of this section is to develop, for this organization, the relationship between retrieval time and design, document file and system usage parameters. Since it is possible to design an inverted file system in which memory space is dedicated but not actually used, a measure of memory usage efficiency will also be developed. This measure is interpreted as an indicator of the advisability of "packing" slow memory information in a manner described below.

The symbol C is used for the number of accession numbers (or other document reference codes) that may be stored in one bucket of slow memory. C is called the bucket capacity. A record associated with a particular index item may be such that it contains many more than C accession numbers, in which case the record must extend over several buckets. Conversely, if several short records exist, it is possible to pack such records into one bucket. If this packing is not done, the portion of the bucket unused is wasted memory space.

Let a convenient unit of slow memory, such as the word, be selected. Next let m be defined as the number of such units required to store one accession number. Then the minimum memory requirement is

the total number of accession numbers multiplied by the memory required to store one such number: Sm. Let V_s be defined as the total memory required for storage of the inverted files, including wasted memory. Then a measure of memory utilization that is zero when no memory waste occurs, unity when twice as much memory as the minimum requirement is dedicated, etc., is:

$$P = \frac{V_s - Sm}{Sm} \quad . \tag{12}$$

Note that this factor is memory utilization when no packing is done, and in fact is used to indicate potential memory capacity requirement reductions obtainable by means of packing.

Again, the decoding from an index item to an initial slow memory address is not considered in this section, but must be included in the overall problem.

In this organization, important parameters used in the calculations are the number of distinct index items, N, and the ratio of the total number of occurrences of all index items to the bucket capacity, S/C. The results may be used to determine, for a given file, the applicability of the inverted list organization. Also, they will aid in the choice of a bucket size, although the maximum and minimum bucket sizes are generally limited by the equipment used. Finally, a measure of the memory requirements with and without packing may be obtained from the memory utilization factor F, and this may be used in the decision on packing.

Figure 7 illustrates schematically the assignment of memory,

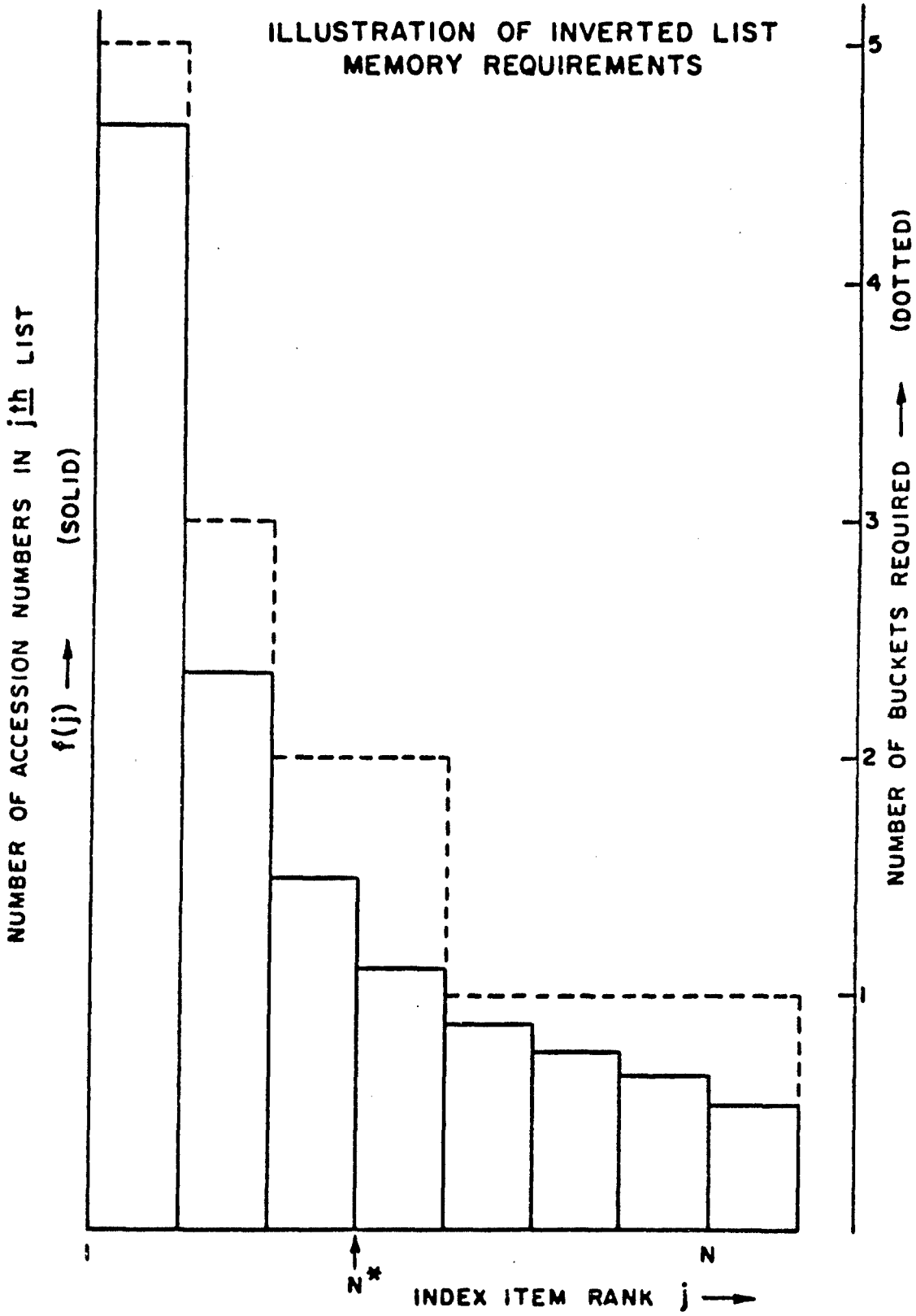


ILLUSTRATION OF INVERTED LIST
MEMORY REQUIREMENTS
NUMBER OF ACCESSION NUMBERS IN jth LIST
f(j) → (SOLID)
NUMBER OF BUCKETS REQUIRED → (DOTTED)
5
4
3
2
1
1
N*
N
INDEX ITEM RANK j →

FIGURE 7

with no packing. There are N distinct index items, and therefore N inverted lists. The longest list has entries requiring more than 4C but fewer than 5C units of slow memory, so five buckets are assigned to that list. The rank of the smallest inverted list requiring more than one bucket is called N*. In certain cases to be considered below, N* will be related directly to the fundamental parameters N and S/C and will be of use in developing formulas relating those parameters to system performance.

Inspection of Figure 7 shows that reduction of C will cause the wasted memory space (the area enclosed by the tops of the solid bars and the dotted lines) to be reduced. However, every bucket access takes one slow memory access time t_s; and when C is reduced, more accesses are required to retrieve a single record that covers several buckets.

Adopting $\lceil x \rceil$ for the smallest integer greater than or equal to x, and $\lfloor x \rfloor$ for the largest integer smaller than or equal to x, it can be seen from Figure 7 that the total memory V_s assigned for inverted lists when no packing is done is

$$V_s = mC \left\{ \sum_{j=1}^{N} \left\lceil \frac{f(j)}{C} \right\rceil \right\} \tag{13}$$

Now the minimum possible response time for slow memory is equal to one slow memory access time t_s; the minimum value of the memory utilization factor P occurs when no memory is wasted and is zero. These two restrictions will not be explicitly stated at every applicable point

in this section but it must be kept in mind that all formulas developed below are subject to the requirements that $T_r/t_s \geq 1$, and $P \geq 0$.

From substitution of (13) into (12)

$$P = \frac{C}{S} \left\{ \sum_{j=1}^{N} \left\lceil \frac{f(j)}{C} \right\rceil \right\} - 1 \quad . \tag{14}$$

As the expression operated upon by the summation symbol cannot be less than unity, the sum cannot be less than N. This allows a second restriction to be placed on P:

$$P \geq N \frac{C}{S} - 1 \; . \tag{15}$$

In order to fetch an inverted list from slow memory, one memory access per bucket is required, so the average response time is given by

$$T_r = t_s \left\{ \sum_{j=1}^{N} \left(\left\lceil \frac{f(j)}{C} \right\rceil \quad p(j) \right) \right\} \quad . \tag{16}$$

As in Section 2.2.2., the first set of assumptions to be used designates that both f(j) and p(j) are uniform: $f(j) = S/N$ and $p(j) = 1/N$. Then substitution into (16) yields

$$T_r/t_s = \sum_{j=1}^{N} \left(\left\lceil \frac{S}{C} \frac{1}{N} \right\rceil \quad \frac{1}{N} \right) \; , \tag{17}$$

from which

$$\frac{1}{N}\frac{S}{C} \leq T_r/t_s < \frac{1}{N}\frac{S}{C} + 1 \quad . \tag{18}$$

A similiar substitution into (14) gives P, the memory utilization factor:

$$P = \frac{C}{S}\left\{ \sum_{j=1}^{N} \left\lceil \frac{S}{N}\frac{1}{C} \right\rceil \right\} - 1 \quad , \tag{19}$$

or

$$0 \leq P < N\frac{C}{S} \quad . \tag{20}$$

Subject to the restriction that $P \geq 0$, the application of (15) to (20) yields

$$\left(N\frac{C}{S} - 1 \right) \leq P < N\frac{C}{S} \quad . \tag{21}$$

This establishes limits on P in the uniform case.

T_r/t_s is illustrated as a function of N for various values of S/C in Figure 8 , and P is shown in Figure 9 . The outer lines, marked "uniform", correspond to the case just developed. In any case in which f(j) is uniform, P may be minimized by selecting a bucket size C so that C = S/N.

Since the choice of C is limited by the physical equipment it may not always be possible to pick an optimum value in a specific situation.

For the second set of assumptions, p(j) will be taken as uniform and f(j) will be assumed to follow Zipf's law (as illustrated by equation (6)). Then

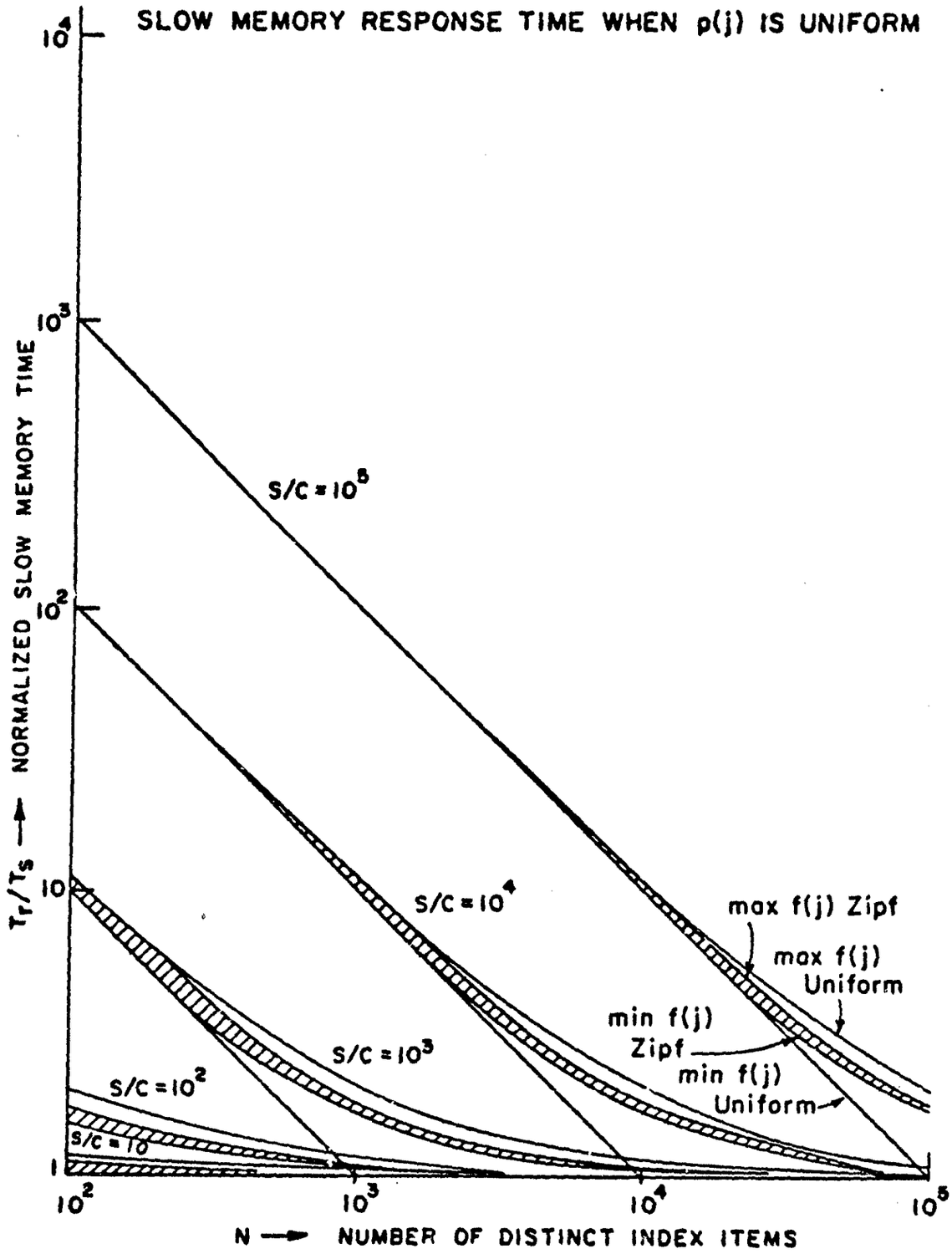

SLOW MEMORY RESPONSE TIME WHEN p(j) IS UNIFORM
T_r/T_s → NORMALIZED SLOW MEMORY TIME
10^4
10^3
10^2
10
1
S/C = 10^5
S/C = 10^4
S/C = 10^3
S/C = 10^2
S/C = 10
max f(j) Zipf
max f(j) Uniform
min f(j) Zipf
min f(j) Uniform
10^2
10^3
10^4
10^5
N → NUMBER OF DISTINCT INDEX ITEMS

FIGURE 8

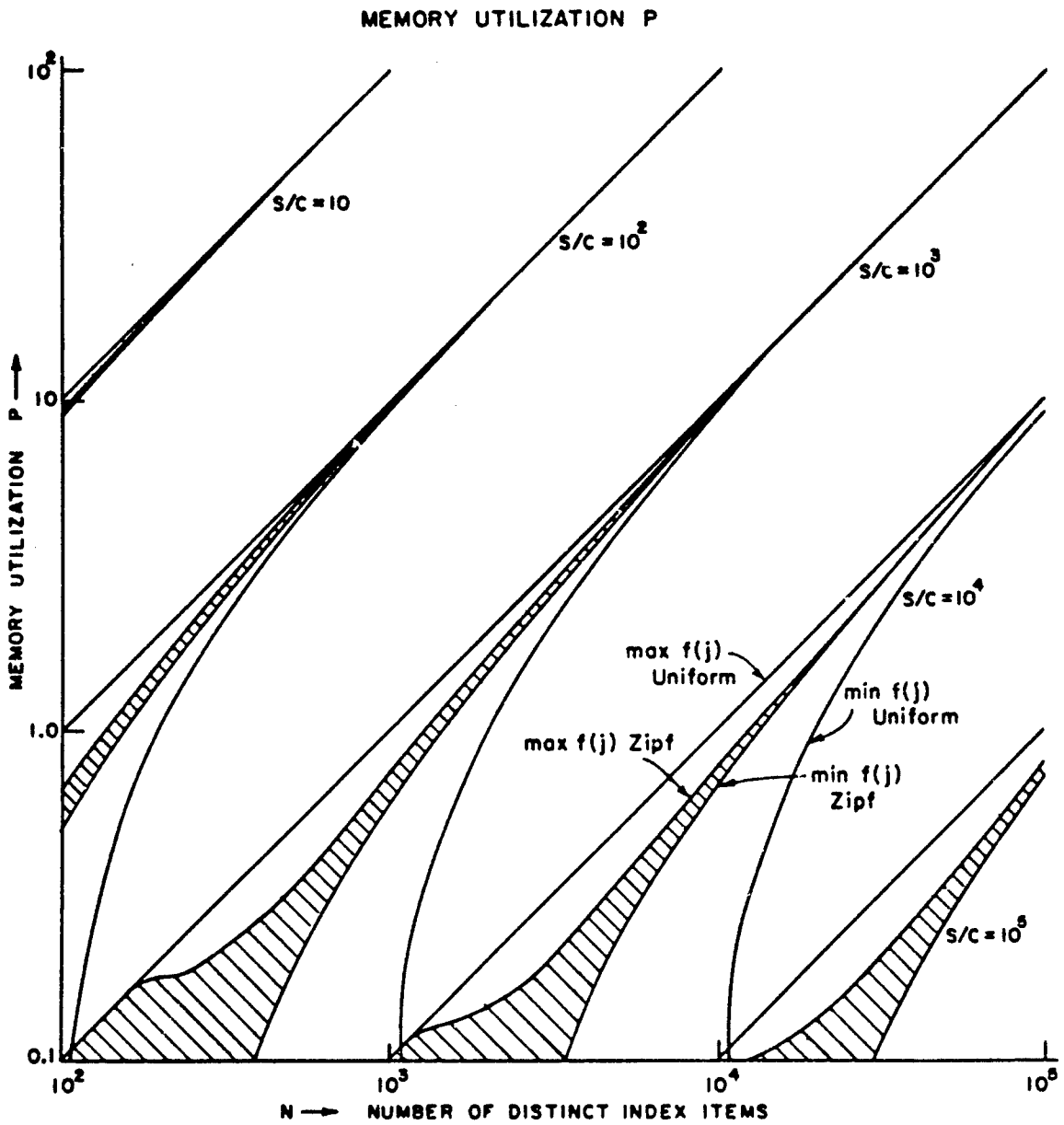

MEMORY UTILIZATION P
MEMORY UTILIZATION P
N → NUMBER OF DISTINCT INDEX ITEMS
S/C = 10
S/C = 10^2
S/C = 10^3
S/C = 10^4
S/C = 10^5
max f(j) Uniform
min f(j) Uniform
max f(j) Zipf
min f(j) Zipf
10^2
10
1.0
0.1
10^2
10^3
10^4
10^5

FIGURE 9

$$T_r/t_s = \frac{1}{N}\left\{\sum_{j=1}^{N}\left\lceil \frac{S}{C}\ \frac{1}{j(\ln N + \gamma)}\right\rceil\right\} . \qquad (22)$$

Equation (22) leads to the same limits on T_r/t_s as those expressed in equation (18), which resulted when a uniform f(j) was assumed for the response time. However, for some ranges of the parameters, finer limits may be obtained in the present case. N* has been defined previously and is illustrated in Figure 7. From (6)

$$N^* = \left\lfloor \frac{S/C}{\ln N + \gamma}\right\rfloor . \qquad (23)$$

In all expressions which are developed in terms of N*, it is assumed that $1 \le N^* \le N$. The alternate expressions (18, 21) must be used when N* is outside of this range.

In equation (16), for $j > N^*$, $\lceil f(j)/C\rceil = 1$. Applying this to (22) yields

$$T_r = \frac{t_s}{N}\left\{\sum_{j=1}^{N^*}\left(\left\lceil \frac{S}{C}\ \frac{1}{j(\ln N + \gamma)}\right\rceil\right) + N - N^*\right\} . \qquad (24)$$

This gives a set of limits applicable when N* meets the above requirements

$$\frac{1}{N}\left\{\frac{S}{C}\ \frac{1}{\ln N + \gamma}\left(\sum_{j=1}^{N^*}\left(\frac{1}{j}\right)\right) - N^*\right\} + 1$$

$$\le T_r/t_s < \frac{1}{N}\left\{\frac{S}{C}\ \frac{1}{\ln N + \gamma}\left(\sum_{j=1}^{N^*}\left(\frac{1}{j}\right)\right)\right\} + 1 . \qquad (25)$$

Note that in general $\sum_{j=1}^{N^*} \left(\frac{1}{j}\right)$ may not be expressed in an approximate closed form, since N* is known only to lie between 1 and N. Of course in the numerical evaluation of the expressions for the limits on P and T_r/t_s the sum is approximated whenever possible. The results for selected values of the parameters are shown in Figure 8.

Calculation of the storage utilization factor, P, by substitution of the assumed Zipf form of f(j) (equation (6)) into (14), subject to the restriction (15), yields (21), the same equation that resulted in the case where f(j) was uniform. When

$\lceil f(j)/C \rceil = 1$ is applied for $j > N^*$, there results

$$P = \frac{C}{S} \left\{ \sum_{j=1}^{N^*} \left(\left\lceil \frac{S}{C} \frac{1}{\ln N + \gamma} \right\rceil \frac{1}{j} \right) \right\} - 1. \quad (26)$$

The limits obtained from (26) are illustrated in Figure 9 . They are:

$$\frac{C}{S}(N - N^*) + \left\{ \frac{1}{\ln N + \gamma} \sum_{j=1}^{N^*} \left(\frac{1}{j}\right) \right\} - 1$$

$$\leq P < \frac{C}{S} N + \left\{ \frac{1}{\ln N + \gamma} \sum_{j=1}^{N^*} \left(\frac{1}{j}\right) \right\} - 1, \quad (27)$$

which completes the derivation for the case when f(j) obeys Zipf's law and p(j) is uniform.

In the final case, p(j) and f(j) both obey Zipf's law. Note

that there will be no difference in P between the present and the immediately previous cases, since only p(j) has changed and P is not a function of p(j).

Substitution of (6) and (7) into (16) gives the response time for the present case:

$$T_r/t_s = \sum_{j=1}^{N} \left(\left\lceil \frac{S}{C} \frac{1}{\ln N + \gamma} \frac{1}{j} \right\rceil \frac{1}{\ln N + \gamma} \frac{1}{j} \right) . \qquad (28)$$

Use of the method employed in the derivation of (10) from (9) yields limits on T_r/t_s that hold for all N*, again subject to the further restriction that $T_r/t_s \geq 1$:

$$\left(\frac{S}{C} (\ln N + \gamma)^{-2} \frac{\pi^2}{6} \right) \leq T_r/t_s \leq \left(\frac{S}{C} (\ln N + \gamma)^{-2} \frac{\pi^2}{6} \right) + 1 . \qquad (29)$$

As before, when $1 \leq N^* \leq N$, a new set of limits may be found. Noting that in (28) the expression inside the "ceiling function" brackets becomes unity for all $j > N^*$, there results

$$T_r/t_s = \frac{1}{\ln N + \gamma} \left\{ \sum_{j=1}^{N^*} \left(\left\lceil \frac{S}{C} \frac{1}{\ln N + \gamma} \frac{1}{j} \right\rceil \frac{1}{j} \right) + \sum_{j=1+N^*}^{N} \left(\left(\frac{1}{\ln N + \gamma} \right) \frac{1}{j} \right) \right\} ; \qquad (30)$$

and hence

$$\frac{S}{C}(\ln N + \gamma)^{-2}\left[\sum_{j=1}^{N^*}\left(\frac{1}{j^2} - \frac{C}{S}(\ln N + \gamma)\frac{1}{j}\right)\right] + 1$$

$$\leq T_r/t_s < \frac{S}{C}(\ln N + \gamma)^{-2}\left[\sum_{j=1}^{N^*}\left(\frac{1}{j^2}\right)\right] + 1 . \qquad (31)$$

Figure 10 shows the behavior of T_r/t_s when the assumptions of this third case, that is that both f(j) and p(j) follow Zipf's law, are made.

2.2.4. Illustration of Results on Inverted Files

Consider the choice of a value of C, the bucket capacity, for a file which contains 10^6 index item occurrences and 10^4 distinct index items. Then $N = 10^4$ and the parameter S/C is equal to $10^6/C$. By use of equations 20, 21, 23, 25, 27, 29 and 31, limits on the memory utilization factor and the normalized response time may be found for various C's. Illustrated below are some typical values:

C	P f unif.	P f Zipf	T_r/t_s f unif. p unif.	T_r/t_s f Zipf p Unif.	T_r/t_s f Zipf p Zipf
100	1.0-0.0	.77-.66	2.0-1.0	1.8-1.7	170-170
200	2.0-1.0	1.7-1.6	1.5-1.0	1.3-1.3	87-86
300	3.0-2.0	2.7-2.6	1.3-1.0	1.2-1.2	58-58
400	4.0-3.0	3.6-3.5	1.2-1.0	1.2-1.1	44-43
500	5.0-4.0	4.6-4.5	1.2-1.0	1.1-1.1	35-35

The two columns showing P for uniform and for Zipf's law behavior of f(j) demonstrate the potential file space wasted if packing

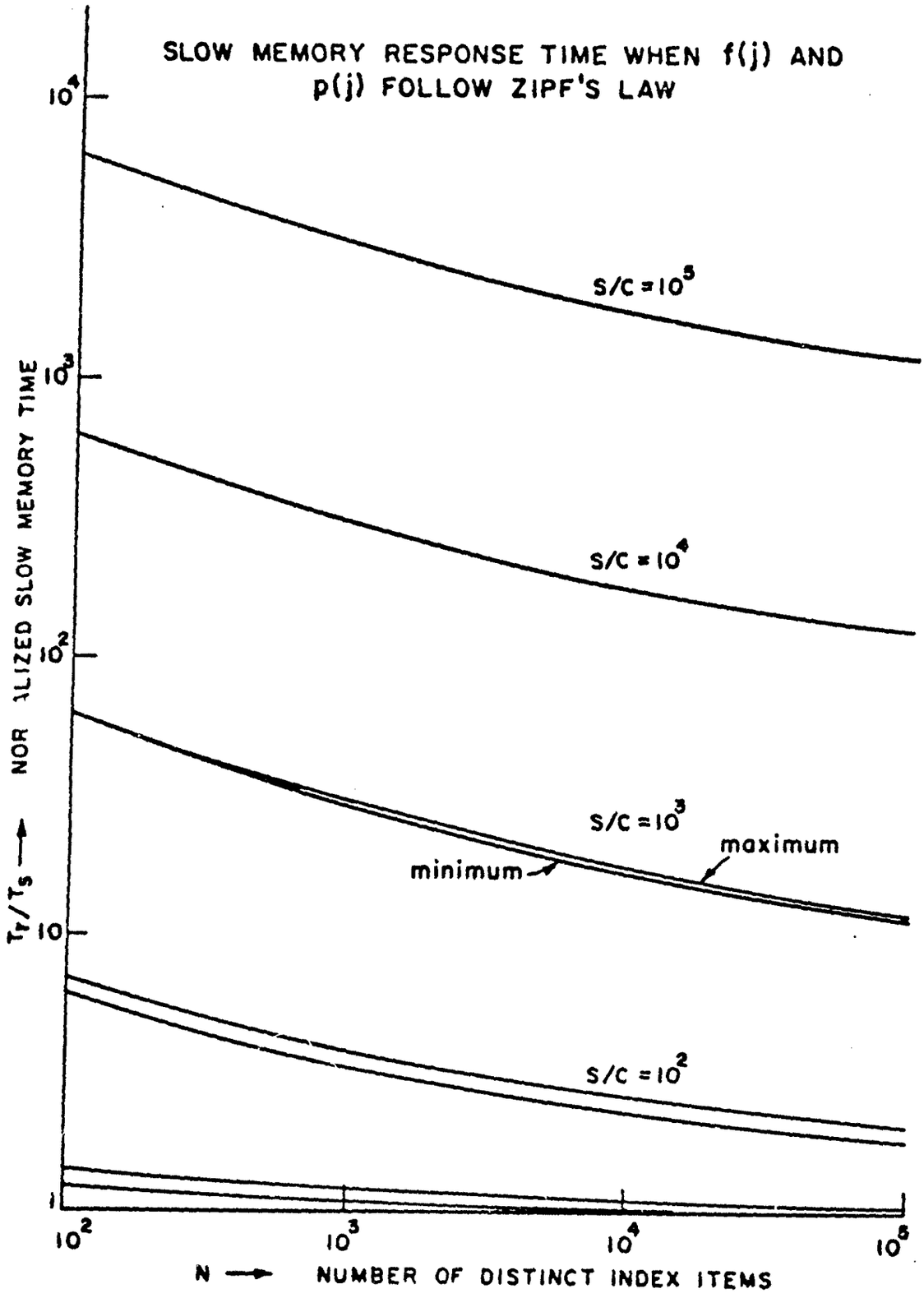

SLOW MEMORY RESPONSE TIME WHEN f(j) AND
p(j) FOLLOW ZIPF'S LAW
10^4
10^3
10^2
10
1
T_F/T_S → NOR ILIZED SLOW MEMORY TIME
S/C = 10^5
S/C = 10^4
S/C = 10^3
maximum
minimum
S/C = 10^2
10^2
10^3
10^4
10^5
N → NUMBER OF DISTINCT INDEX ITEMS

FIGURE 10

is not performed. Similarly, the effect of C on average response time for three different combinations of f(j) and p(j) is illustrated in the last three columns. Thus, using the equations mentioned above and an appropriate assumption concerning f(j) and p(j), a value for C may be selected and a decision concerning the packing of more than one list into single buckets may be made.

2.2.5 Conclusions on Slow Memory Data Organization

The conclusions to be drawn from the preceding discussion are summarized here.

First, the linear file simply is not suited for use in an environment in which real time response is required. This is illustrated by the sample figures given in Section 2.1.

In order to compare the linked list with the inverted list, attention must be given to the fact that variable length data records may be stored at the nodes of the linked list. The effects of this are considered later in this section.

If only accession numbers are to be retrieved, then the inverted list structure will require one slow memory access to retrieve up to C accession numbers, while in the linked list structure one access will be required for every accession number retrieved. Thus, only when the number of documents characterized by an index item approaches unity will the linked list accession time approach the inverted list accession time.

When the intersection of lists is desired--that is, the location of documents described by each of several different index items--a similar effect is noted. If the shortest linked list thread is followed for intersections with all other threads, there still will have to be a slow

memory access for every node on that list. The inverted list structure would have to access slow memory once for every C entries on every list corresponding to an index item. Thus, a generally superior response time is expected when an inverted list structure is used.

The linked list does have certain important advantages. In particular, it is much more easily updated than the inverted list, as mentioned in section 2.2.2. This is of great importance in some applications, but not in the particular information retrieval problem considered here.

Because of the capability of the linked list structure to store variable quantities of data as well as fixed-length items, it is of considerable value in applications in which variable-length data records are necessarily going to be used. But in the current application, may operations may be made on lists of accession numbers (to which have been added the short qualifying fields discussed in chapter 3). After all these operations have been performed, only a small amount of document data will generally be needed. As will be explained later, it is true that for every document record eventually retrieved, one additional slow memory access will be required. But since considerable more processing is to be made using the inverted list data and all this processing is expected to result in the retrieval of little document data, this advantage is relatively unimportant in the present application.

Introducing the assumption that p(j) has a characteristic predicted by Zipf's law in the form used here creates a considerable change in the response time. This is shown in Figure 10 and is similar to the effect illustrated in Figure 6 for the linked lists. However,

in the case of the inverted file there is a control over the runaway response time. T_r/t_s may be reduced by reducing S/C, to the extent that the maximum physical record size permits one to do so.

From a response time standpoint, then, the choice of an inverted list structure with the largest possible ratio of S/C allowable is dictated. However, Figure 9 shows that it is possible to waste enormous quantities of memory by the use of this structure; in the schematic diagram of Figure 7 the area between the solid and dotted lines represents this wasted memory. But by placing short records in that space (i.e. "packing") such waste can be greatly reduced.

The finally chosen memory data organization, then, is an inverted list structure with packing of the records and the largest values of S/C possible, the limiting value being imposed by the equipment used. It must be recalled that this conclusion has been reached in the light of currently available commercial equipment and is subject to re-evaluation whenever equipment with different capabilities becomes available or radical changes in the relative costs of large memory capacities occur.

2.3. Decoding to a Slow Memory Address

In the previous sections, with the exception of the linear file, it has been assumed that some mechanism exists that, when provided with an index item, provides a slow memory address. That slow memory address might be the location of the head of a list, linked or inverted.

It is the purpose of this section to survey several of the

most promising of these decoding mechanisms, in order that one appropriate to the task at hand may be selected. Ramifications arising from the use of fixed length representations of index items are also considered.

Many workers have investigated problems in the area of address decoding. The basic problem is easy to understand: for example, assume that an index item consists of 28 characters from an alphabet of 37 symbols. There are then about 10^{44} possible index items. In a typical situation at most 10^4 or 10^5 items will actually be used, so that the decoding problem is one of mapping the index items used into appropriate slow memory addresses.

Several schemes for decoding will be considered in brief. Not all of these methods were originally proposed for this purpose, but they are applicable to it.

Some of the terminology of Brooks and Iverson[3] will be used here. In particular, a scan will be called <u>directed</u> if it continues in one direction always advancing by the same increment, and <u>controlled</u> if some algorithm dictates the direction and increment based on the outcome of previous comparisons. If the normal forward directed scan encounters the first element of a list immediately after the last, then the scan is said to be <u>cyclic</u>; if the initial element of a scan is always the same (generally the first element of the list) then the scan is said to be <u>rooted</u>.

It is necessary that in this section the discussion be particularly oriented toward the characteristics of the equipment available.

Not only do such factors as word size, fixed word length operation as compared to variable word length (or character oriented), and character handling features in fixed word length machines enter the picture, but as Bowlden[1] has pointed out, some machines, such as the IBM 650 and the Control Data RPC-4000 have instructions that may make vast differences in the methods used.

2.3.1. Symbol Trees

In symbol trees each node represents a symbol (that is, a single character) of an index item, and branches lead to following symbol nodes. If two or more items have the same initial symbol sequence they share an initial path down the tree until the first symbol differentiating the two symbol sequences is encountered. The average branching factor at the nodes and the tree depth may be controlled by considering an index item as a string of bits, and generating a new symbol sequence by partitioning the bits in a way different from the original character representation scheme. The size of the alphabet used in the tree representation can thus be varied. Fredkin has proposed the use of two bits per node in his "binary Trie"[7]. In the decoding application being presently considered, the slow memory address could be placed at the last node of every path through the tree corresponding to an index item.

The efficient mechanization of such a tree structure has been given considerable attention. Sussenguth[25] has proposed a doubly chained memory structure that eliminates the reservation of computer memory space for the unused nodes that are represented by blank cells in Fredkin's Tries. But passing through a node via the second

(continuation) chain does not involve that node's symbol in the symbol sequence. Scidmore and Weinberg[22] have proposed a tree structure which has no unused nodes also, although three linkage addresses are required at each node and frequently not more than one is used.

Cheydleur[5] has proposed the physical construction of a memory element utilizing not only sharing of initial symbol strings, but the sharing of polygrams in general.

Such symbol trees are of particular advantage in working with data of variable length. Some tree implementations may be updated easily, but that is not an item of major importance in the present application. Recalling that a "symbol" need not correspond to a character of the original alphabet, at least one comparison and address calculation must be performed for every symbol of the symbol string being traced through the tree. The symbol tree cannot give a false slow memory address for an item not contained in the tree.

2.3.2. Key to Address Transformation

In this method, first described as "open addressing" by Peterson[14], the index item is transformed to an address by some algorithm. Such transofrmations include selection of digits or bits, "folding" by adding some parts of the index item to others, or taking a remainder after division. The address so found is one of a bucket and is taken as the starting point of a directed cyclic scan of that bucket and all following buckets. The scan continues until either the desired item is found, or a flag indicating an unused memory location in which the item would be placed had it existed in the file is encountered. Several transformation algorithms are discussed in one

of the references[26], in which the use of a table of partial index items is suggested to reduce the number of occurrences of identical addresses corresponding to different index items.

As the allocated memory becomes full, the number of searches for an item grows.

Chaining from full buckets instead of the use of cyclic structure has been suggested.

Such systems use a relatively small amount of high speed memory, but have the disadvantage that unless the complete file can be tested in advance the performance of the system is very difficult to predict.

2.3.3. Triplet Searching

At this method, proposed by Rubinoff[17], has not been described in the open literature, it will be covered in somewhat greater detail here than the other methods.

The first step in the triplet method is the arrangement of all the index items into lexicographic order and partitioning them into groups. The groups are not necessarily of the same length, as two restrictions are made on the partitioning*. The first restriction is that the groups may not exceed some predetermined number of index items and the second is that the initial K characters of the last item of a group may not be identical with the initial K characters of the first item of the following group. In other words, it is possible to distinguish the index items at a group division using only the first Q

* Another possible reason for modifying the length of groups is mentioned later in this section.

characters of the index items. K is chosen depending on the equipment used, and it is convenient to make K the number of characters contained in a word if a fixed word length machine is used.

In order to search for an index item, first the group in which the item lies must be located. This may be done by performing a binary search of the list of group identifiers, these group identifiers being the first K characters of the first index item of each group. That is, a relatively short list of single words is searched so that the group in which the index item falls may be identified.

After the correct group has been found, a list of <u>triplets</u> that have a one to one correspondence with the index items of the group is brought into fast memory. The triplets consist of an integer, which need never exceed the length of the longest index item, one symbol, and the linkage address (which may be a relative address) to slow memory. Such a list of triplets is illustrated below: in the sample shown the different linkage addresses are all represented by β; σ is used for a blank or space character immediately following the last non-blank character of an index item and $\emptyset$ indicates an unused character position in a computer word.

DIRECTION OF SEARCH

ORIGINAL INDEX ITEM	TRIPLET
TOIL	4LB
TOILE	5EB
TOILER	6RB
TOILET	6TB
TOILETRY	7RB
TOILETTE	7TB
TOILFUL	5FB
TOILSOME	5SB
TOILWORN	5WB
TOIT	4TB
TOKAY	3KB
TOKEN	4FB
TOKOLOGY	4OB
TOLA	3LB
TOLAN	6σØ3LB
TOLBOOTH	4BB
TOLD	4DB
TOLE	5σØ4EB
TOLEDO	5DB
TOLERABLE	5RB
TOLERANCE	7NB

Except for the two double triplets shown, with this scheme information is generally packed so that only one machine word is required for each index item reflected in a triplet group, if a fixed word length with a reasonably large word format is used.

Now, after the correct group of triplets has been found by a search of the group identifiers and the triplet group fetched into fast memory, the triplets are searched in the direction shown above. If α is an integer and X a symbol, then the comparison for the triplet $\alpha X \beta$ is as follows: the αth character of the complete index item is compared with the symbol X. If they are identical, then the search has been completed and β is the desired address. If not, then the next triplet in the group is so inspected. If the last triplet (at the top of the list in the example) is reached with no successful comparison, then the index item is not in the file.

In a few cases triplet pairs are required, which indicate two required conditions. The first is the length of the index item, and the second a comparison as before.

Referring to the example triplet group, suppose that the β address corresponding to the index item TOLD is required. (For this discussion TOLD will be considered to be justified left in a field containing as many blanks as necessary.) First, the list of group identifiers (initial K characters of initial index items of groups) is searched, and the triplet group shown above is brought into fast memory. Then the triplets are examined one at a time. Since the seventh character of TOLD is not an N, the first triplet is not the correct one.

Similarly, the second and third triplets indicate R and D as the fifth characters of the complete index item, which they are not. The fourth triplet indicates that the item is four characters long by 5ø (blank at end of word in the fifth position) but requires that the fourth character be an E by 4Eβ. This latter condition is not met by the word TOLD. Finally, the next triplet does indeed correspond to the correct index item, and is 4Dβ. The match occurs when it is found that the fourth character of TOLD is D.

If an index item is not in the system, a false hit is possible; entering the search process with TOLERABLY would retrieve the β address associated with TOLERABLE. For this reason, in most applications the full index item should be repeated at some location in the record specified by the β address.

Mention has been made before of the fact that the β address may be partial or relative. Since the group identifier is already determined when the triplet search begins, there may be an additional address segment or base associated with the group identifier.

The triplet group is generated in the reverse direction from that in which it is searched (that is, from the top down in the example). The first index item (TOIL) is assigned its last letter as its character identifier (4L) and an appropriate β. The second, third, and all succeeding index terms are compared character by character with the items above them, scanning left to right. When the first character position in which the item being inspected differs from all items above it is found, that character and column position become the first two elements of the triplet. Thus, TOILSOME is assigned the

triplet 5SØ because T has appeared before in column 1, O in column 2, etc., but nowhere before has S appeared in column 5. When, as in the case of TOLE, the scan goes to the end of the word, the first half of a double triplet is used to indicate the length of the word (5ØØ), and the scan starts again, from right to left, but this time comparing column positions with only the previous index items of the same length as the current index item. Note that it is possible for this scheme to fail:

ABC	3CØ
BCA	1BØ
CAB	1CØ
CBA	Cannot be assigned.

In such a case, it becomes necessary to shorten the group of index items.

2.3.4. Balanced Trees

A tree decoding mechanism quite different from those discussed before has been described by Landauer[6,11]. In this tree every branching node (called the "associative catena"), except those on levels of the tree not completely filled, has a fixed number of branches. An index item is also part of each node. Suppose that index items are arranged in lexicographic order: $I_1, \ldots, I_N$. If the branching factor is m, then m branches come from every node. In particular, suppose that a node associated with item I_j appears on level k of the tree. Then the filial set of node I_j on level k is the set of nodes I_j, I_{j+1}, ... , I_{j+m-1} on level k+1. If the tree has T levels, then there are m^T branches leaving the bottom of the tree;

in the particular case being considered these branches would be links to inverted files.

Such a tree may be traced rapidly. Landauer has developed theories of searching, generating and updating them. The primary disadvantage of such a tree for the proposed application is that it is not well adapted to the use of variable length index items.

2.3.5. Effects of Truncating Index Items

In the brief discussions of the decoding mechanisms just presented, it appears that some schemes are better equipped to handle variable length index items than others. With fixed word length computers, such as the one on which the present system is implemented, it is convenient to limit the size of index items in the decoding mechanism to one word. A simple method of doing this is the truncation of every item so that only the first few characters of the item are used. Because of the possible convenience of using truncated index terms, the effects of truncation have been investigated using the ACM data.

What is being considered in this section is not only a key-to-address problem in the usual sense. It is proposed that the truncated index item be used in most processing, with a check against the complete index items (which are to be carried complete only in slow memory) being made as records are retrieved from slow memory. If the penalty resulting from loss of uniqueness due to the use of truncated index items is found to be relatively small compared with the advantages resulting from using fixed word length techniques, then truncation of index items may be allowed before the decoding of those items to slow memory addresses.

Let the set of N distinct index items be $D = d_1,\ldots,d_N$. Every item consists of a string of a maximum of L characters $c_1\ldots c_E\ldots c_L$ such that the characters $c_1\ldots c_E$ are taken from some alphabet A and $c_{E+1}\ldots c_L$ are a each special (blank) character. If an index item consists of L non-blank characters, then E = L and no blank is required. Thus, it is assumed that no item contains more than L characters, and when convenient every item may be represented by a field of L characters with all of the non-blank characters justified left in that field.

Now the index items forming the set D may be replaced by a set D = $d_1,111,d_M$. This set if formed from D by replacing every index item in D by its initial K characters. This set will have M elements where M N. The generation of such a set D from a set D is shown in Figure 11, where N = 20, K = 4, and after truncation only 14 distinct strings exist, so M = 14. The elements of D have been arranged in lexicographical order for convenience; this is not necessarily required.

Let $S_1,\ldots,S_m$ be the largest subsets of D which upon truncation reduce to single elements $d_1,\ldots,d_M$ of D . Such subsets are illustrated in Figure 11. Note that

$$\bigcup_{i=1}^{M} S_i = D . \tag{32}$$

Now if h(j) be the number of sets S_i such that S_i = j, and J be the smallest number such that for all j J, h(j) = 0, then in the example

ABSO|LUTE	S_1
ACCE|SS	S_2
ACCO|UNT ACCO|UNTING	S_3
ACCR|UAL	S_4
ACCU|MULATOR	S_5
ACOU|STICAL	S_6
ACTI|VE ACTI|VITY	S_7
ADAM|S	S_8
ADDE|R	S_9
ADDI|NG ADDI|TION ADDI|TIVE	S_{10}
ADDR|ESS ADDR|ESSING ADDR|ESSOGRAPH	S_{11}
ADVA|NCING	S_{12}
AFFI|XES	S_{13}
AGIN|G	S_{14}

Figure 11 - Truncation for K = 4

of Figure 11: $h(1) = 10$, $h(2) = 2$, $h(3) = 2$, and $J = 3$. In general

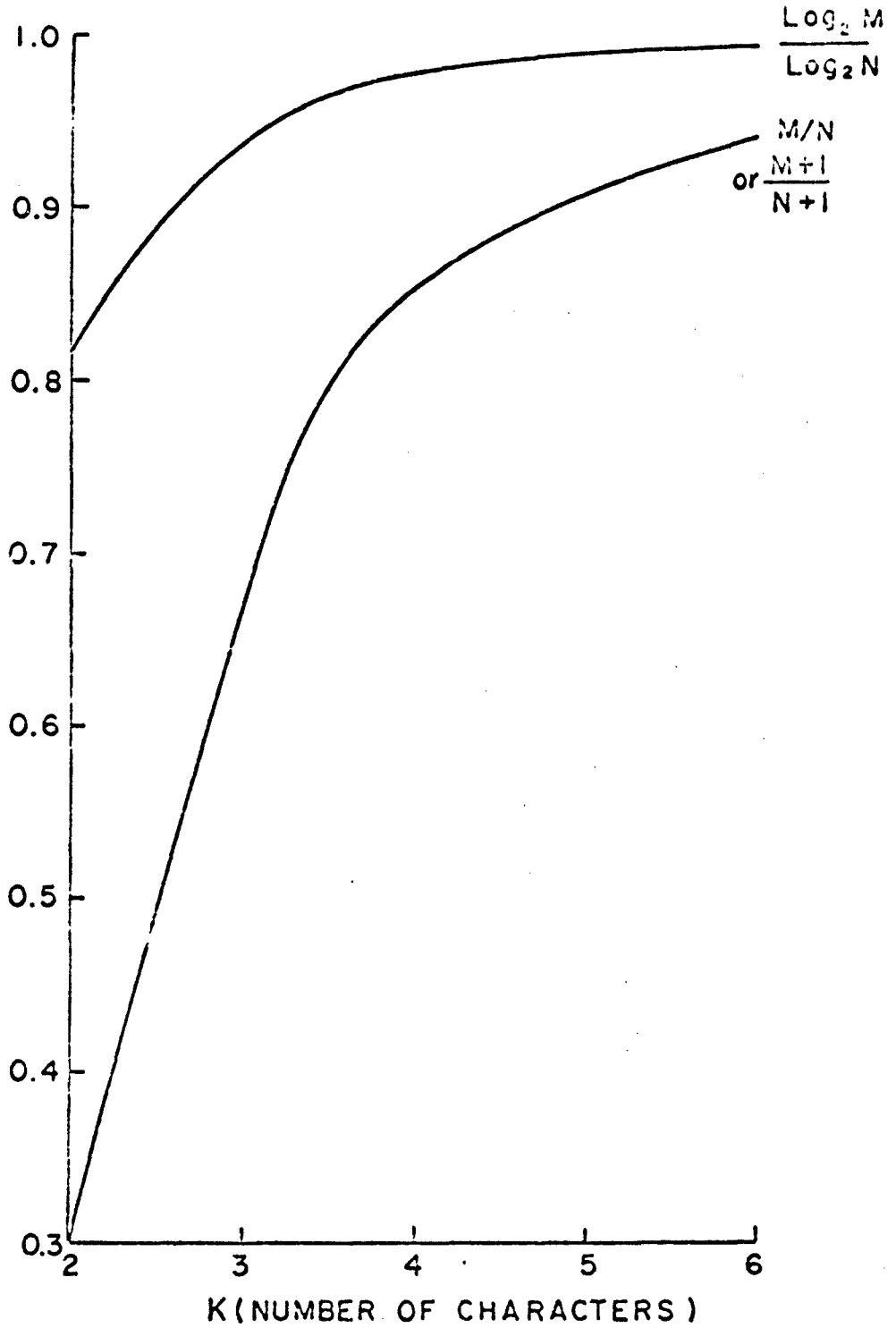
$$N = \sum_{j=1}^{J} jh(j) \tag{33}$$

and

$$M = N - \sum_{j=2}^{J} (j-1)h(j) \quad . \tag{34}$$

If index items are stored in memory, the length of such a list is reduced by a factor of M/N if the list is truncated, and the corresponding number of comparisons when such a list is searched is reduced by a factor of $(M+1)/(N+1)$ in the event of an ordinary directed scan, or by about $(\log M)/(\log N)$ in the event that a binary search is performed. This is illustrated in Figure 12.

Now assume that some decoding mechanism exists that can give the address of a record in slow memory, given the truncated index item associated with that record. That is, given any one of the M elements of D^* (such as d_i^*), the decoding mechanism will generate the slow memory address of a corresponding record $R_{i,1}$. This record contains a complete index item d_j which is an element of the set S_i. When truncated, d_j becomes d_i^*. If S_i has more than one element (that is, if any elements of D other than d_j become d_i^* upon truncation), then there must be a link from the record $R_{i,1}$ corresponding to d_j to records $R_{i,2}, \ldots$ corresponding to the other complete index items which are elements of S_i.



EFFECT OF TRUNCATION ON ITEM LIST LENGTH
BASIS - ACM REPOSITORY CORE COLLECTION
FIGURE 12

A schematic diagram is shown in Figure 13, illustrating this linking of records. Only the linkages are shown; the inverted lists or other data are of course also present in the records.

The steps in bringing into fast memory a record (if any exists) associated with an index item θ of length L, which may or may not be an element of the original set D, are:

1) θ is truncated to length K, forming θ^*.
2) A decoding mechanism determines if θ^* is an element of D^*. If it is not, the process terminates. If θ^* is an element of D^*, for convenience let that element be called d_i^*. The decoding mechanism provides the slow memory address of record $R_{i,1}$.
3) Record $R_{i,1}$ is brought into fast memory. In its control section there is a complete index item, which is compared with θ. If θ and this index item are identical, the search terminates.
4) If θ and the index item in the control section of the record in fast memory are not identical, the link of the record is checked. If there are no further records ($R_{i,2}$, etc.) the search terminates. If there are any other records, they are brought into slow memory and the process continues.

So, assuming that a record is associated with an index item, the chain of slow memory records associated with items that become identical to the given item upon truncation must be followed. Of course there are alternate arrangements, such as placing a chain table of contents at

i	d_i^*	$R_{i,1}$	$R_{i,2}$	$R_{i,3}$
1	ABSO	ABSOLUTE		
2	ACCE	ACCESS		
3	ACCO	ACCOUNT →	ACCOUNTING	
4	ACCR	ACCRUAL		
5	ACCU	ACCUMULATOR		
6	ACOU	ACOUSTICAL		
7	ACTI	ACTIVE →	ACTIVITY	
8	ADAM	ADAMS		
9	ADDE	ADDER		
10	ADDI	ADDING →	ADDITION →	ADDITIVE
11	ADDR	ADDRESS →	ADDRESSING →	ADDRESSOGRAPH
12	ADVA	ADVANCING		
13	AFFI	AFFIXES		
14	AGIN	AGING		

FIGURE 13

SCHEMATIC ILLUSTRATION OF SLOW MEMORY RECORD LINKING

the head of every chain. But even with the proposed linkage scheme, analysis will show reasonably efficient operation, and the scheme is a simple one.

In the process of bringing the correct record into fast memory, the important factor to be determined here is the effect of truncation insofar as it increases the number of slow memory accesses required. For any $\Theta = d_i \in S_g$, the average number of slow memory accesses required is $\frac{|S_g| + 1}{2}$. Since for $1 \leq j \leq J$, there are $jh(j)$ records total in chains of length j, then the average number of slow memory accesses required to retrieve a record corresponding to some $\Theta \in D$ is

$$K_r = \sum_{j=1}^{J} \frac{j(j+1)h(j)}{2N} . \tag{35}$$

If $\Theta \notin D$, but $\Theta^* \in D^*$, and the chains have no table of contents and are not necessarily arranged with records in lexicographical order of their index items on the chain, it is necessary to scan the entire chain in order to determine that $\Theta \in D$. An average number of slow memory accesses, K_r, has been found (35) for $\Theta \in D$. K_n is the number when $\Theta \notin D$:

$$K_n = \sum_{j=1}^{J} \frac{j^2 h(j)}{N} . \tag{36}$$

The worst case, which might occur whether the index item Θ is or is not an element of D but $\Theta^* \in D$, results in

$$K_m = J . \tag{37}$$

For use with empirical data, it is convenient to tabulate the functions h(j) for $2 \le j \le J$. Then the following forms are convenient:

$$K_r = 1 + \sum_{j=2}^{J} \frac{j\,(j-1)\,h(j)}{2N} \tag{38}$$

and

$$K_n = 1 + \sum_{j=2}^{J} \frac{j\,(j-1)\,h(j)}{N} \quad . \tag{39}$$

These may be obtained from (35) and (36) by substitution for h(1) from (33), separating the first term from the remaining terms of the summation in all equations. In the forms (38) and (39), the extra slow memory accesses caused by truncation are clearly evident.

The values of the K's obtained are shown in Figures 14 and 15 . Since the K's are multiples of slow memory accesses, they may also be considered to be multipliers of the T_r/t_s ratios previously developed, representing the penalties resulting from incomplete decoding of slow memory addresses prior to starting slow memory accesses.

2.3.6. <u>The Modified SRS System</u>

Consider a tree similar to that described in Section 2.3.4., but with a branching factor that may vary from level to level. Its disadvantages are a memory requirement somewhat larger than those of the other schemes and an inability to process variable length index items with ease. Because of the structure of the tree, however, groups of filial sets may be stored in slow memory records so that the memory requirement is not a serious one. The results of Section 2.3.5.

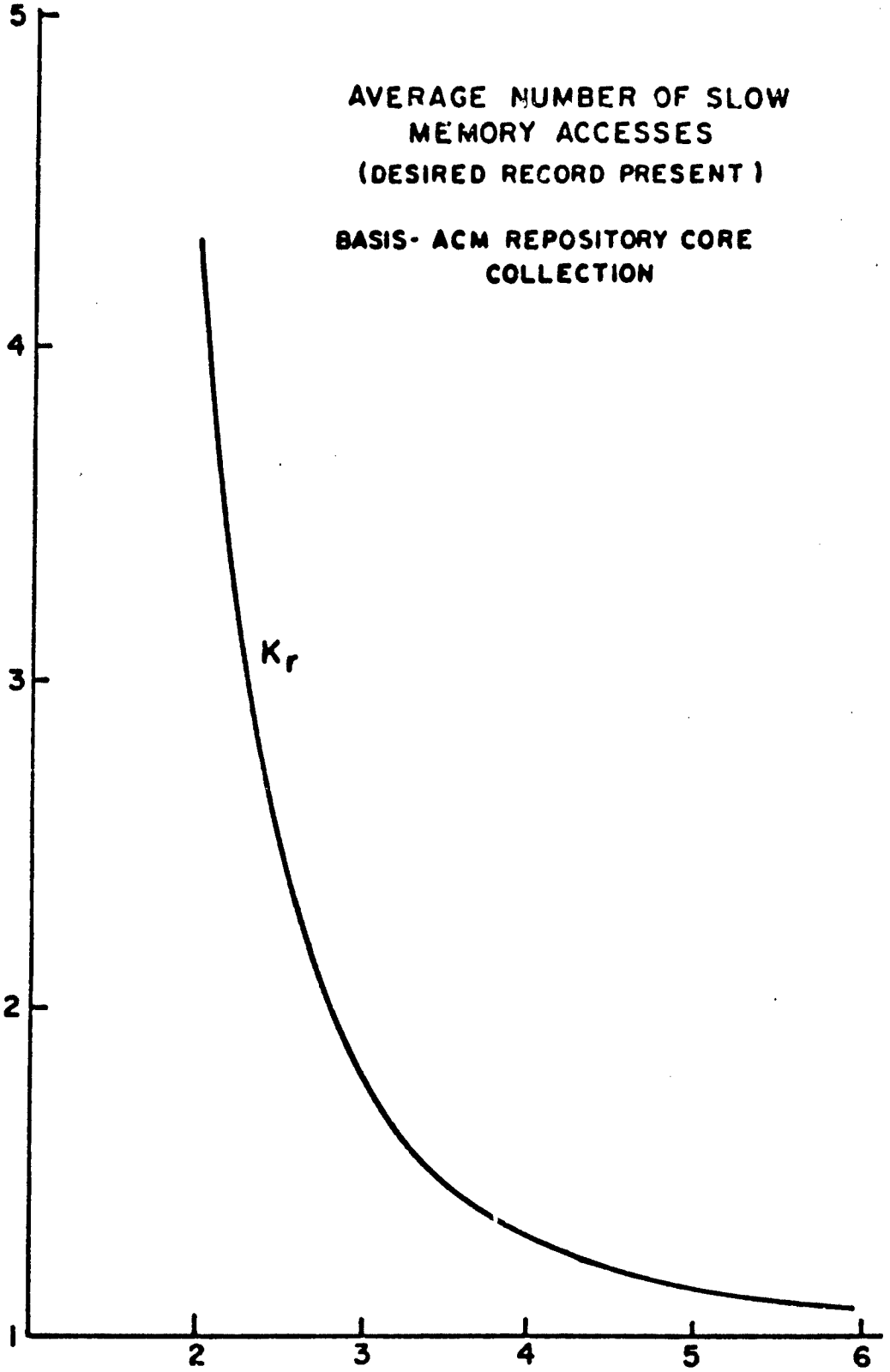

AVERAGE NUMBER OF SLOW
MEMORY ACCESSES
(DESIRED RECORD PRESENT)
BASIS- ACM REPOSITORY CORE
COLLECTION
K_r
5
4
3
2
1
2
3
4
5
6
K (NUMBER OF CHARACTERS)

FIGURE 14

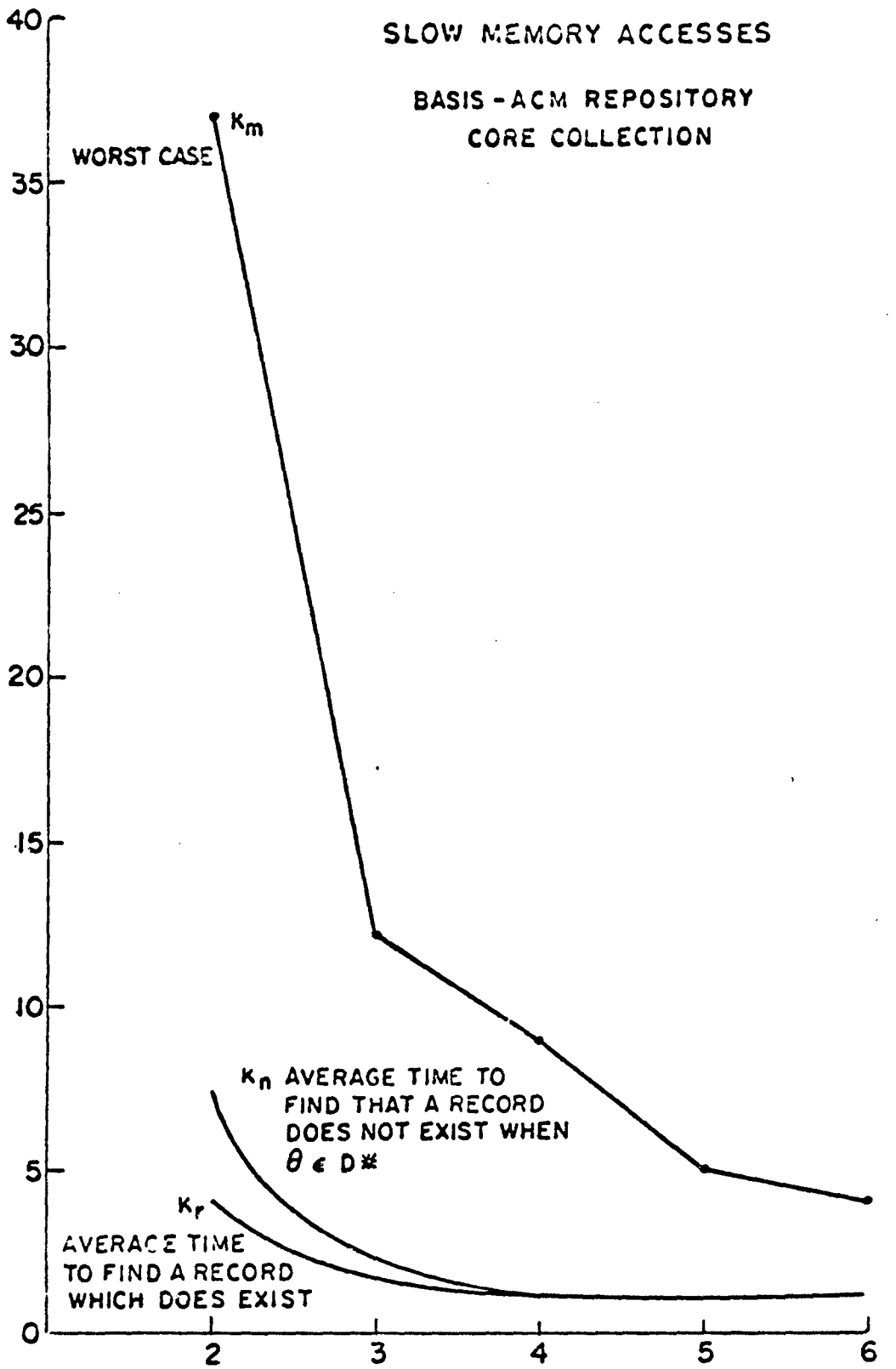

SLOW MEMORY ACCESSES
BASIS - ACM REPOSITORY
CORE COLLECTION
WORST CASE
K_m
K_n AVERAGE TIME TO FIND THAT A RECORD DOES NOT EXIST WHEN $\theta \in D^*$
K_r
AVERAGE TIME TO FIND A RECORD WHICH DOES EXIST
40
35
30
25
20
15
10
5
0
2
3
4
5
6
K (NUMBER OF CHARACTERS)

FIGURE 15

indicate that the use of index items truncated to only five characters will increase the number of slow memory accesses by only about three percent, and allow the use of fixed word length. These very few extra accesses are judged reasonable, so such a tree may be used.

A group working under Professor Noah S. Prywes at the Moore School of Electrical Engineering of the University of Pennsylvania has developed a problem-solving facility that runs on the same computer equipment as the information system described here. A service routine operating within this facility is called SRS - Storage and Retrieval System[8], and with some modifications the SRS system has been made to include both a decoding mechanism and the inverted files. This modified system uses five-character index items for the initial record access, comparing complete index items with data stored in the records as described in Section 2.3.5. But whenever it is possible to do so, inverted lists that have the same truncated index items are stored packed in the same bucket. The effect of this is to reduce the penalties created by truncation of the index items.

Actually, the modified SRS routine has been integrated into a complete list handling package. This is described in a later section, so only a short description indicating how the package retrieves inverted lists will be given here.

It should be noted that the SRS structure is essentially one of linked lists. In this particular application, the links are used for continuation flags and for connecting lists that have identical truncated index items. Because of automatic packing, the bucket size is taken as large as possible to reduce the S/C ratio.

The organization of the inverted files and the decoding mechanism is illustrated schematically in Figure 16. In fast memory there is a single tree level. The required section of the second level of the tree may be brought into fast memory from slow memory with one slow memory access, and provides the information necessary to decode to one packed slow memory bucket. The tree is therefore not a balanced tree, since it does not have a constant branching factor.

In order to find an inverted file corresponding to an index item, first the tree is scanned for the path to the slow memory level. In the case of the index items QUADRANT, QUADRAT and QUALITY, which when truncated become QUADR and QUALI, the slow memory information lies between PINT and ROAR.

The segment covering index items from PINT to (but not including) ROAR is then searched. If the truncated item is QUALI, the bucket containing (among other things) the QUALITY inverted list may be fetched. The table of contents of this bucket indicates the location of the QUALI material in the bucket. There is no link from QUALI, so the only inverted list whose name's initial letters are QUALI is the list named QUALITY. The program checks the complete index item, however, since the user might introduce a new word (such as QUALIFY) that is not currently in the system. At the end of the inverted list for QUALITY there is a symbol that indicates the end of the list.

The data associated with QUADR illustrate other points. Tracing through the system for QUADRAT, the first entry in the QUADR section is QUADRANT. But a pointer indicates a second bucket, showing that more QUADR data exist. The first section of the second bucket is

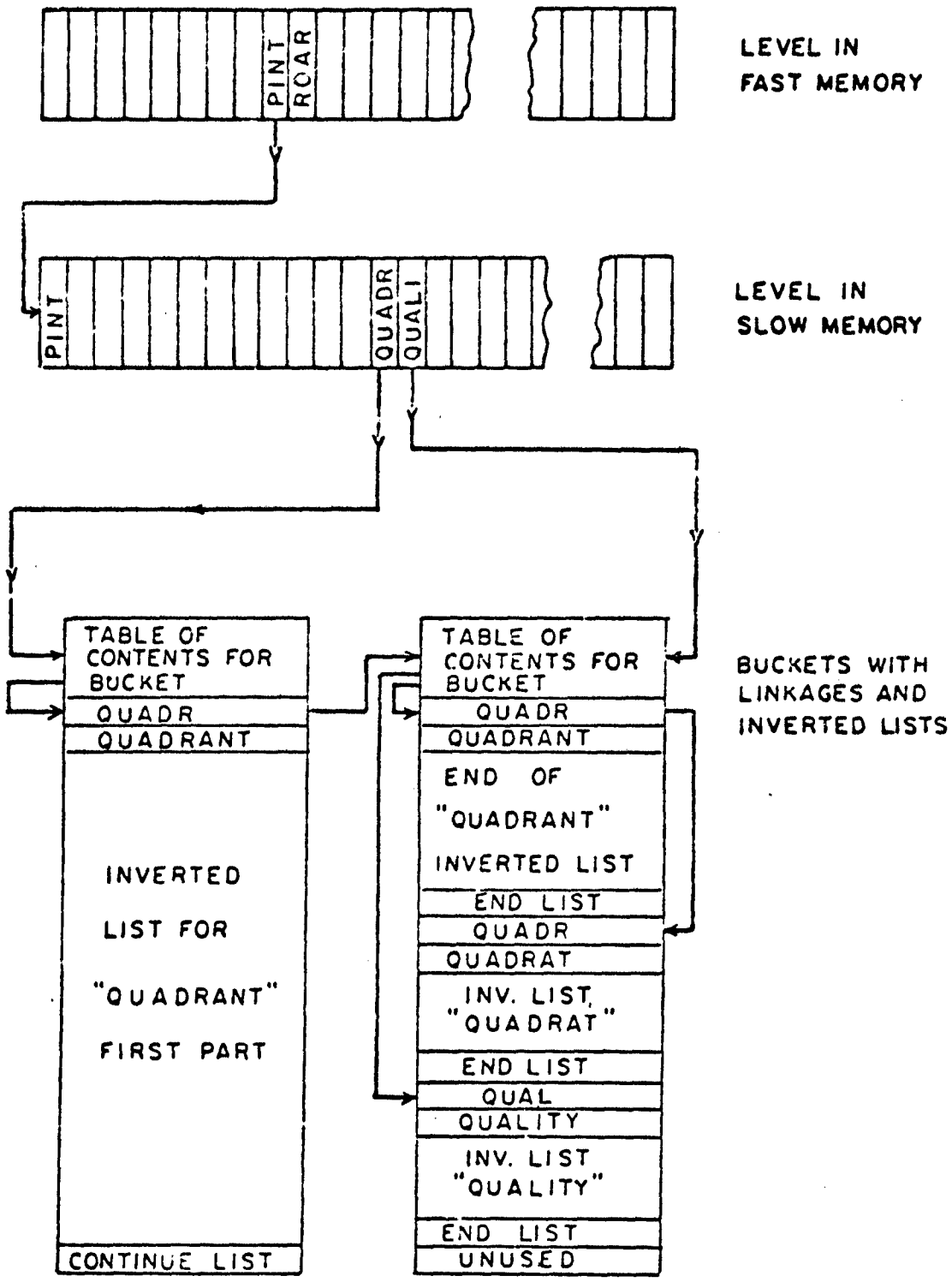


FIGURE 16

MEMORY ORGANIZATION

again for QUADRANT, because the inverted list for that index item required more space than was available in the first bucket (this is indicated by a continuation symbol, rather than an end of list symbol, at the end of the first bucket). However, a pointer indicates the location of the QUADRAT inverted list.

2.3.7. Conclusions on the Decoding Mechanism

The modified SRS system, described in the previous section, is used in the present implementation of the information retrieval system. One factor in this selection was that the original SRS system was available for the particular computer employed, and could be modified as described in the previous section. The problem of false hits was exchanged for a slight additional average delay; analysis shows that the use of truncated index items with a provision for additional searching in the event of a mismatch on the full items does not cause excessive delays. The delay does become more severe as N increases, but is small for the files anticipated. The tree used has advantages when mechanized using slow memory for storage of part of the tree. It is partitioned so that every search requires just one slow memory access before data buckets may be retrieved. Thus the fast memory requirements are small.

The modified SRS system is well suited to the present application where speed of retrieval and small fast memory requirements are important, and ease of updating is not a major consideration.

Figure 17 summarizes the more important features of the decoding mechanisms considered in this chapter.

	Symbol Tree	Key-to-Address Transformation	SRS-Type Tree	Triplet Search
Processes Variable-Length Items	Yes	Yes	No	Yes
False hits Possible when Item in File	No	Yes	No	No
False Hits Possible When Item not in File	No	Yes	Yes	Yes
Extensive Character Processing Required	Yes	No	No	Yes
Easily Partitioned for few Slow Memory Accesses	No	-	Yes	Yes
Easily Updated	Yes	Yes	No	No
Large Fast Memory Requirement	Yes	No	No	No

FEATURES OF DECODING MECHANISMS

Figure 17

3.1. Data List Structures and Operators

This chapter describes the inverted lists used in the information system and the generation of these lists. Fundamental operations upon lists used in the retrieval process are covered, as well as the use of temporary working lists. Also described is a set of files that supply the user with additional information concerning documents retrieved.

Detailed information on input data specifications, formats, and similar information may be found in the Appendix.

In the process of indexing, documents are characterized by words, word phrases and codes under a wide variety of categories, such as author's name, article's title, and descriptors* (e.g., "ordinary differential equations", "ring counter"). Such a characterizing unit is called an index term. All conventional bibliographic terms, descriptors and added information codes** are index items. Index terms have been structured by a convention that forces the indexer to specify that an index term falls into one of the following categories:

---

* Descriptors are words or word phrases that describe the subject matter of that document. Characterization of documents by descriptors may be likened to the generation of subject heading cards in a conventional library but on a more extensive scale. The word "descriptor" is used here to embrace the transients, free terms and descriptors of reference 18.

** See references 18 and 19 for an explanation of the indexing scheme used in establishing the data file. To the scheme presented in those references, the identification of the indexer and the date of the indexing have been added for use in quality control. See also the Appendix

Date of indexing and indexer's identifying code

Author's name(s)

Date of issue

Title

Editor's name(s)

Identification of issuer

Page size

Number of illustrations

Number of pages

Collection in which document appears

Descriptors

Added information codes.

These categories are called sectors. For any particular sector, there may be more than one index term associated with a single document. A document with more than one author is an example, each author's name being a separate index term.

An index item is a single code or word that serves as an element of an index term; any index term consisting of more than one word (set of contiguous characters not including a space) may be deconcatenated into an ordered sequence of index items. Additional processing is also performed when index terms are transformed into index items.

Index items may be obtained as follows:*

By deconcatenation of an index term; e.g., "RING COUNTER" becomes "RING" followed by "COUNTER";

* The sector code tags, introduced in Section 3.2., are not shown in this example.

By deconcatenation and substitution of a standard abbreviation; e.g., "AUGUST 1966" becomes "AUG" followed by "1966".

By expansion of a condensed or abbreviated index term; e.g., "(P.L.)" becomes "PROGRAMMING" followed by "LANGUAGE".

By deconcatenation and elimination of common words, e.g. "RETRIEVAL OF INFORMATION" becomes "RETRIEVAL" followed by "INFORMATION".

Retrieval is performed not on a basis of index terms, but by use of index items. It would of course be possible to retrieve by index terms rather than index items, but by using index items retrieval is accomplished even if only part of a word phrase is known. The means by which this process has been mechanized are discussed in Section 3.3.4., but the effect is described here. The discussion immediately below does not apply to the special added information sector, since in that category information is represented by distinct codes whose order is unimportant and in which the space character is used only for providing ease in reading.

Let an index term be represented by a string of words $W_1, W_2, \ldots, W_n$. If documents were characterized by complete index terms, only that string could be used to retrieve the document or documents characterized by it. Now let $i_1, i_2, \ldots, i_m$ be integers such that $1 \leq i_1 \leq i_2 \leq \ldots \leq i_m \leq n$. Then in the system using index item characterization, the string $W_{i_1}, W_{i_2}, \ldots, W_{i_m}$ will retrieve the document or documents originally characterized (and indexed) by the

string $W_1, W_2, \ldots, W_n$. Thus, retrieval is possible when incomplete knowledge of a word phrase exists.

Consider, by way of further illustration, the following. Let α, β, γ, σ, ϵ be index items used to form index terms such as $\alpha\ \gamma\ \epsilon$. Let three documents, A, B, and C, be characterized by the index terms $\alpha\ \beta\ \gamma\ \sigma\ \epsilon$, $\alpha\ \gamma\ \epsilon$ and $\alpha\ \gamma$, respectively. It is assumed that no other documents exist. The strings $\alpha\ \beta\ \gamma\ \sigma\ \epsilon$, $\alpha\ \beta$, and σ all will retrieve only document A. The string $\alpha\ \gamma\ \epsilon$ will retrieve documents A and B. Either α or $\alpha\ \gamma$ or γ will retrieve all three documents. A string containing an index item other than α, β, γ, σ and ϵ will retrieve no documents. The order of index items is taken into account, so that the string $\gamma\ \alpha$ will retrieve no documents.

With appropriate sector codes, the descriptor "FORTRAN" would retrieve all documents indexed with any phrase in which the word "FORTRAN" appears. "IBM 704 FORTRAN II" would retrieve a smaller number of documents dealing with a more restricted concept.

3.2. Generation and Format of Inverted Lists

Inverted lists are formed for every index item used in the characterization of documents, excluding common words (at, the, etc.) and such information as page size, number of pages and number of illustrations. In the generation of these lists and in their use, the sector of each index item is coded as an additional character and appended to the item. Thus every inverted list is named by an index item and the context (sector identification) of the index term from which the index item was generated.

The inverted list named by an index item (with a sector specification) contains the set of accession numbers of all documents characterized (under that sector specification) by an index term containing that index item. It also contains the order of the index item in the index term in which it appears. This is done by including in every inverted list entry not only the accession number of the document but also the position of the index item in the index term characterizing that document. Inverted lists thus consist of accession-position number pairs.

By way of illustration, suppose that two documents have the following information included in their indexing:

Accession Number	110
Author	BORTEK, CHARLES
Title	LETTER TO PACT POLICY COMMITTEE ENCLOSING FINAL REPORT OF PACT II WORKING COMMITTEE

Accession Number	113
Author	WAGNER, FRANCIS V
Title	LETTER TO MEMBERS OF SHARE, ON PACT 1A COMPILER SYSTEM

Using numerical tags "1" to indicate the author and "3" to indicate title, part of the information from the above document indexing would be added to inverted lists as follows:

List Name	BORTEK1	CHARLES1	LETTER3	PACT3	FINAL3	1A3
Accession-	110-1	110-2	110-1	110-2	110-6	113-5
Position			113-1	110-8		
Number Pair				113-4		

In the example, since the word "FINAL" appears in the title (sector three) of document number 110, the accession number 110 appears in the inverted list named "FINAL". The word "FINAL" is the sixth index item in the index term "LETTER TO THE PACT POLICY COMMITTEE ...", not counting common words, so the position number, six, is added to the accession number, forming "110-6".

Every accession-position number pair is, in the present mechanization, stored in one 36-bit computer word. The second through thirtieth bits are used for the accession number. As accession numbers are not necessarily integers, they are carried in a modified character-by-character representation. The last six bits of the word are used to store the position number. The first bit is present only when the word represents not an accession-position number pair, but an internal "housekeeping" information word indicating either the end of the inverted list or the end of that portion of the list currently held in fast memory.

3.3. Operations on Inverted Lists

Requests for retrieval of information are stated in terms of strings of words modified by sector indicators and related by logic operators, and are translated into operations upon inverted lists. This translation, covered in Chapter 5, is similar to the operation of an algebraic compiler that translates arithmetic statements composed of operators and variable names into arithmetic operations upon

the values identified by the variable names.

In this Section, the list operators will be discussed. There are four operators that are used to process inverted lists. All are binary, operating on two lists to produce a third (temporary) list, and are represented in infix notation by the symbols & , + , ↑ , and Δ . These symbols have been chosen from the rather limited set of available symbols on the standard four-row Teletype model 33 or 35, with the exception of the Δ symbol. Δ represents an operation not explicitly named in the input message.

A convenient representation of the inverted or temporary list A_j, i.e., the list in the inverted file corresponding to the jth index item, where A_j contains n_j members, is the set of n_j ordered pairs:

$$A_j = \{ (a,p)_1, \ldots, (a,p)_{n_j} \} \; ; \; |A_j| = n_j \; .$$

If n_j is zero, A_j is said to be null*. Every ordered pair $(a,p)_i$ corresponds to an instance of the use of the particular index item, with its section code, which names list A_j. The left element a of the $(a,p)_i$ pair is the accession number of a document characterized by an index term from which the index item associated with list A_j was derived. The right element p indicates the position of the index item in the index term, except in the case of some sectors corresponding to inverted lists where position numbers are not meaningful, and similarly for certain temporary lists where p is always set equal to zero.

In the computer, all lists are stored so that they are in numerical order when considered as integers (p containing less significant digits than a).

* In the computer, a null list is represented by a single end of list flag.

3.3.1. List Union

The "union" of lists X and Y is represented by X+Y. The union operator is commutative, but is not identical, although similar to, commutative union of sets. As position numbers are not needed in lists generated by this operation, the position number part of every resulting position-accession number pair is set equal to zero. The elements of the resulting temporary list are defined by:

$$X+Y = \{ (a,0) \mid [\exists p] [(a,p) \in X \vee (a,p) \in Y] \} .$$

If X is the list of documents characterized by index item α (such as the author Smith) and Y by item β (such as the author Jones), then X+Y is the list of accession numbers of papers characterized by either α or β or both (written by Smith or Jones or co-authored by them).

3.3.2. List Intersection

The "intersection" of lists X and Y is represented by X & Y. The intersection operator is commutative. As position numbers are not needed in lists generated by this operation, they are treated as in union. The elements of the resulting list are given by:

$$X \,\&\, Y = \{ (a,0) \mid [\exists p][\exists p'] [(a,p) \in X \wedge (a,p') \in Y] \} .$$

Using the example at the end of Section 3.3.1., X & Y is the list of document accession numbers characterized by both α and β (papers co-authored by Smith and Jones).

3.3.3. List Complementation

The "complement" of a list on the universe of all accession-position number pairs in the system would be unwieldy to mechanize and

of little value. The operator which is used generates the list of accession numbers that appear in one list but not in a second; it is an "and not" operator. The intersection of X with the complement of Y is represented by $X \uparrow Y$. The resulting list contains exactly once every accession number appearing in list X but not in Y. The operator is not commutative. Position numbers are treated as in union. The elements of the resulting list are given by:

$$X \uparrow Y = \{ (a,0) | [\exists p][\forall p'] [(a,p) \in X \wedge (a,p') \notin Y]\} .$$

From the example of the past two sections, $X \uparrow Y$ is the list of accession numbers of documents characterized by a α but not by β (papers written by Smith alone or in collaboration with anybody other than Jones).

3.3.4. <u>List Concatenation</u>

This operation is used in connection with retrieval by a string of index items which may not contain all the items appearing in the characterizing index term. The concatenation of two lists, X and Y, is represented by $X \Delta Y$.

In the resulting list, no two accession-position number pairs have identical accession numbers. An accession number appears in the pairs of the resulting list if and only if that accession number appears in both list X and list Y and the position number of the accession-position number pair in list X does not exceed the position number of the accession-position number pair in list Y. In the resulting list, each accession-position number pair is assigned the

position number appearing in list Y, which is not necessarily zero.

It is possible that two or more accession-position number pairs exist in list Y having the same accession number but different position numbers and that more than one of these meets the requirements stated above. In this event, only the pair having the smallest position number and meeting the other qualifications appears in the resulting list.

This is the operation that makes it possible to retrieve documents characterized by word phrases, using only parts of those phrases. The operator does not commute, and its action is more easily stated in formal notation than in English:

$$X \Delta Y = \{ (a,p) | [\exists p'] [(a,p') \in X \wedge (a,p) \in Y \wedge (p' \leq p) \wedge [\forall p''] [(a,p'') \in Y \Longrightarrow (p'' \geq p)]] \} .$$

Some examples of the effects of the concatenation operation (including repeated application) are given at the end of Section 3.1. Consider the document with accession number 1328 in the ACM Repository, which has the title (sector 3), "A Table of Characteristics of Soviet Digital Computers". If an inverted list associated with any word (other than a common word) in the title is X, and a second list associated with a second word in the title is Y, then accession number 1328 will appear in the retrieved $X \Delta Y$ list if and only if the first word appears before the second in the title. Extending this, the document with accession number 1328 will be specified by repeated applications of the concatenation operator (applying it from left to right) to any

string $W_1,\ldots,W_n$ if that string of index items may be generated from the original title (not including common words) by deleting zero or more words from the title. (It is not permitted that all words be deleted.) As specifications become less and less complete, more and more accession numbers may be included in the resulting list: e.g., "digital" is surely less specific than "Soviet Digital Computers", and will exclude fewer documents from the retrieved list.

3.4. List Storage and Retrieval

The four list operators described in Section 3.3 all require two lists as arguments, and generate a resulting list. Lists are of variable length. Long lists must be segmented, obtained from slow memory segment by segment, and stored in slow memory as segments are generated. The structure of inverted list storage is described in Chapter 2, and in particular the overall structure is presented in Section 2.3.6. The routines with which the list operation routines (and some others) must communicate in order to obtain or store list segments are described in this section.

A maximum segment length is one bucket, less all normal bucket control words except one. The control word which must be retained follows all data in a list segment, indicating the end of data or that there are additional segments on the list.

Four routine entry points are used for list storage and retrieval. One routine takes successive list segments from a buffer area and stores them in slow memory. Another fills one of two buffers in fast memory with list segments from slow memory. The latter has two entry points, each one corresponding to a different list and fast

memory buffer.

The routines above, when called, do not receive an explicit reference to a particular list. They have been designed this way so that once the name of a list has been decoded, and the head of the list located in slow memory, the decoding mechanism does not have to be used when additional segments of the same list are called for. Since the list processing routines all generate just one list, there is only one entry point for list storage routine. As the list processing routines all require two arguments, the retrieval routine has two entry points.

A third routine "primes" the list storage and retrieval routines. Before a list may be stored or retrieved, the third routine is called and provided with the name of the list in question and the desired action. Then subsequent calls to the storage and retrieval routines store or retrieve consecutive segments of the list named in the call to the "priming" routine.

The four entry points discussed in this section--one routine with a single entry point which sets up the identities of lists to be stored or retrieved, one with one entry point, which stores list segments,and one with two entry points, which retrieves list segments--provide an interface between the routines that operate on lists and part of the modified SRS system discussed in Section 2.3.6.

3.5. List Types

Three types of lists exist in the system: inverted, temporary,and linear. The inverted lists, previously described, are named by their index items. Index items are derived from index terms

and sector codes, and are (including blanks) twenty eight characters long.

Temporary lists are available for use as working storage, and are the only type of list which may be written upon. Temporary lists are given numerical names, expressed internally in the computer as binary numbers. (Because it is possible that the name of a temporary list might be identical with the name of some other list, the "primer" routine,which associates list names with list storage and retrieval routine entry points must be given information indicating whether or not a list specified to it is a temporary list.)

Consider the execution of an information retrieval command by the system. A specification consisting of index terms, sector identifiers and operators is processed, generating a temporary list of accession numbers of documents meeting the specifications. To the user of the system, who initiated the retrieval command, the accession numbers are of some value. But of considerably greater value is the ability to retrieve part or all of the information used in the characterization of the document--author, title, and all the other information shown at the beginning of this chapter. This information is stored in a linear file, within the list structure of the system. One list is associated with each document. The selective use of the data that are stored in these lists is covered in Chapter 5.

The name of a list in the linear file is simply the accession number of the document with which the list is associated, preceded by a dollar sign. Since accession numbers in the present system consist of five or fewer characters, and the capacity of a word is six

characters, the name of a linear list may be contained in one IBM 7040 word. As no inverted list names may contain a dollar sign, the retrieval programs may easily recognize a linear list and decode only six characters, as opposed to twenty-eight.

4.1. Provisions for User-Computer Communications

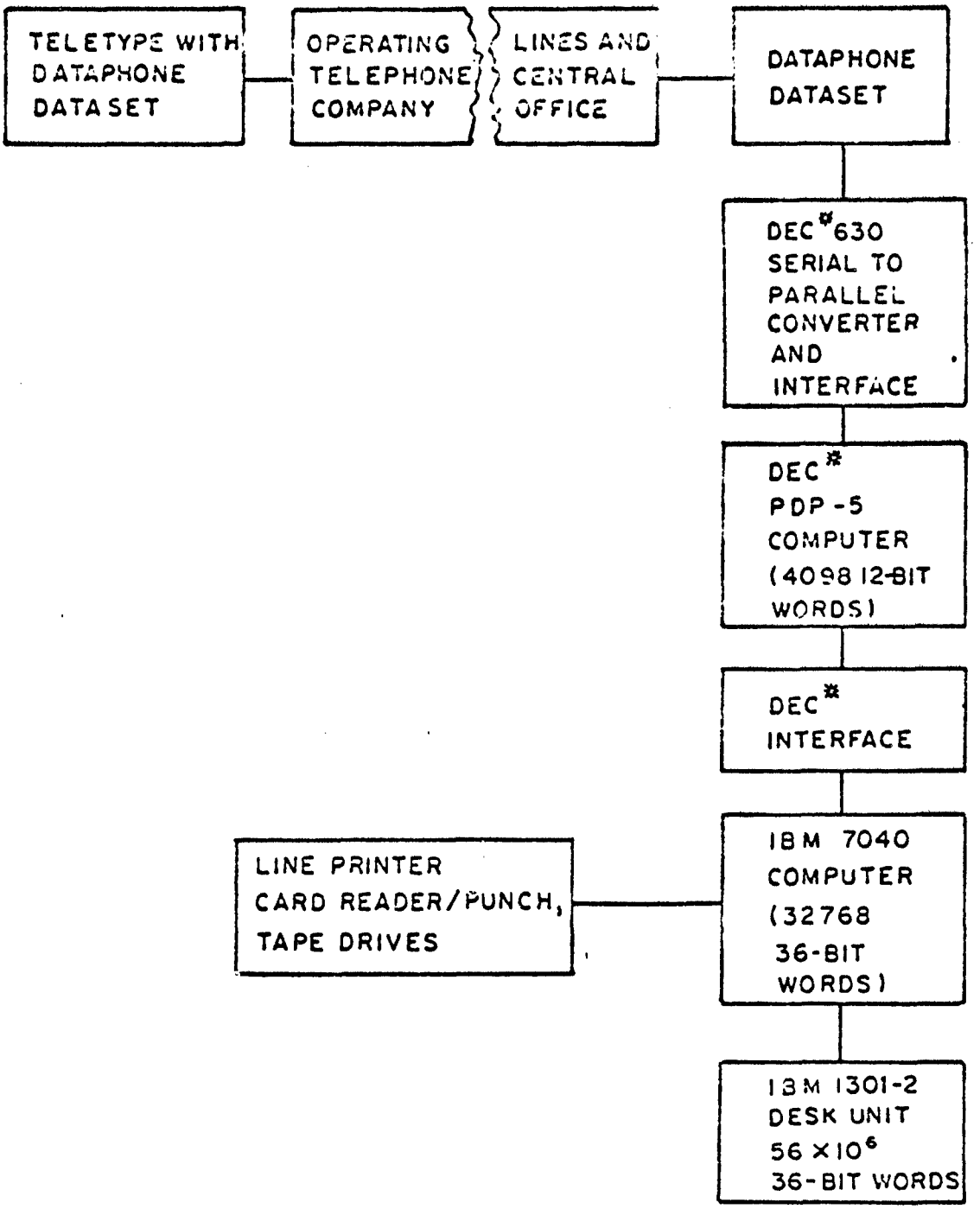
In order to facilitate real-time man-machine communications using a remote reactive teletypewriter terminal, the information retrieval system has been provided with message generation functions. These allow the user of the system to edit, proofread, and correct his messages before their transmission to the processor that performs the actual information search. The editing routines are combined with the general programs that control communications between the user and the system (and also control the execution of background programs).

It is the purpose of this chapter to describe the functions of these communications routines as seen from two viewpoints. One is that of the user, at the remote terminal. The second is that of a systems programmer writing the central information retrieval programs.

Before the communications program functions can be explained, a brief description of the equipment and several available options is needed.

4.1.1. Equipment Employed

Units are interconnected as shown in Figure 18. Two computers are employed, the larger IBM 7040 being used for the information retrieval processing. The smaller computer, a Digital Equipment Corporation model PDP-5, provides the user of the system with the facility to generate, inspect, and edit messages before they are



* Digital Equipment Corporation

FIGURE 18

EQUIPMENT CONFIGURATION

submitted for processing to the IBM 7040.

The remote Teletype-Dataphone terminals can be set to either half- or full-duplex mode. A sense switch* on the PDP-5 console makes the corresponding selection at the computer equipment. Full-duplex operation is preferred, since with it the PDP-5 computer controls all the printing at the remote station. This permits suppression of characters that are inadvertently typed by the user but are not in the set of allowable characters. It also insures that the user of the system knows whether or not he is communicating with the computer. An additional advantage is one that arises because a user may, in error, attempt to send a message to the system at the same time that the system is trying to send a message to him. This causes no trouble with full-duplex equipment, but with half-duplex it may terminate the telephone connection.

Remote stations may have parity bit generation features, but it is not required that they be so equipped.

Although this chapter is written in terms of a remote console, the PDP-5 console may also be used as a terminal if it is so selected by sense switch option.

4.1.2. Practice Mode, Paper Tape, and Background Programs

In order that the users of the system may become familiar with the editing procedures and the use of the remote Teletype units, a sense switch option is provided that allows operators to formulate, proofread and correct input messages without having the 7040 process

* A description of the options which may be selected by use of the sense switch controls is printed out at the PDP-5 console when programs are first loaded.

them. This is the "practice mode".

For test and demonstration purposes, sense switch options are provided that allow the use of punched paper tape as an input medium.

Since the user of the system must compose, type, and correct input messages, and since the standard Teletype unit is limited to ten characters per second, the large computer may not actually be processing information retrieval programs for much of the time during which it must be available to the system. But it must be immediately available whenever its use is requested. For this reason, the 7040 has been programmed to process "background" computations, executing other programs when it is not needed by the information retrieval system. When the PDP-5 signals the 7040 for attention, the 7040 stops processing the background program, waits for any input and output processes to terminate, stores the status of the program counter and various working registers, and gives control to the information retrieval programs. When the 7040 is no longer needed for retrieval, the execution of the background program is resumed at the point where it had been interrupted previously.

4.2. Communications from the User's Viewpoint

In this section, the formulation and editing of messages generated by the user and addressed to the information retrieval system will be discussed and illustrated. The subprograms that accomplish message handling have been designed with ease of use as a primary objective. In particular, the user should have to remember as few rules as possible. When a user response is required during the editing

ocess, he is always provided with some cue. Because of the speed limitations imposed by the use of slow Teletype units, such prompting messages must be short; but experience has shown that the use of even very short messages establishes a simple "quasi-English" dialogue between the user and the system.

The discussion below is presented in the following sequence: initial connection procedures, delay and disconnect features, and the man-machine dialogue provisions.

In order to establish initial contact with the system, a remote user must dial the number of the Dataphone connected to the PDP-5 computer. When the Dataphone answers, the user must type a space character to open the conversation. This is required because upon establishment and termination of telephone connection, unwanted characters are sometimes generated. By requiring a specific character before dialogue may begin, and by rejecting all "noise" characters preceding that specific character, the effects of undesired characters are greatly reduced.

As mentioned above, the PDP-5 computer may be set for practice mode, in which case only the editing features may be used and no messages are sent to the 7040 computer for processing. In practice mode, once contact has been established via the Dataphones and the space character transmitted, the remote operator is given the instruction "ENTER MESSAGE". After all message formatting and editing has been completed in practice mode, the PDP-5 causes the remote station to print the statement: "THIS IS THE MESSAGE WHICH WOULD BE PRINTED TO THE 7040", followed by the final version of the message.

If practice mode is not selected, that is, when normal operation is desired, the first message printed at the user's console after contact has been established and the space character transmitted is "STANDBY". This is repeated once every minute until the 7040 computer becomes available. This delay for the 7040 is anticipated when a run is being started, since a user may dial into the system when the PDP-5 is loaded and ready for use, but the 7040 has not yet been loaded with the information retrieval and background programs. When the 7040 does become available, it is given control of the entire system, including communications with the user. (The first message transmitted to the user asks for his identifying user number, but that is a function of the 7040 executive programs, not the communications routines being discussed here.) After establishing contact in practice or normal mode, the user is asked to answer some question or enter some request.

Unless the user is operating in practice mode, the 7040 also has control over the termination of the connection between the remote console and the PDP-5 computer; and, therefore, it has the ability to disconnect a user from the system. The actual disconnection is accomplished when the PDP-5 sends a special end-of-transmission (EOT) code to the remote station, which causes the telephone connection to be broken and turns off the power at the remote terminal. A user may be disconnected if he requests termination. Or in the event that manual intervention or the interval timer of the 7040 indicates that alloted time has expired, a special routine which takes only microseconds of 7040 time causes the PDP-5 to transmit to the remote terminal "7040

NO LONGER AVAILABLE. CONNECTION TERMINATED.", followed by an EOT code.

The PDP-5 may also initiate such termination. Whenever a response is expected from a user, a timer routine in the PDP-5 starts. The timer is reset every time a character is received from the remote station. If the timer reaches two minutes, a warning message is sent to the user at the remote terminal. If there is still no response after another minute, the message "EXCESSIVE DELAY. CONNECTION TERMINATED." is sent, and the remote station is disconnected.

Whenever the system expects input, the user is informed of this by a message such as "YOU MAY PROCEED :=" or "SAME INFORMATION AS BEFORE? :=". He then formulates his message in answer to the system, using the "short editing" and on some occasions the "full editing" described below. All messages generated by the operator are terminated by the message termination symbol pair - two adjacent inequality symbols: "< >". Thus, if the last symbols appearing during the progress of the dialogue are ":=", the machine is waiting for the user; and if the last symbols are "< >" , the user is waiting for the machine.

Generally, when a machine question is to be answered "YES < >" or "NO < >", the question is followed by "?:=".

4.2.1. The Input Alphabet and Short Editing

The following characters may be used in any combination to form input messages, subject to restrictions on the use of the exclamation point, question mark and inequality symbols:

- the letters A, B, ..., Z;
- the digits 0, 1, .. , 9;
- the space;

- The special characters below:

!	$	'	(	)	+	?	&
↑	-	/	.	,	:	;	=
*	<	>					

The maximum length of an input message is 2700 characters; it is highly unlikely that messages of greater length will be encountered in normal use.

The characters used to delimit a new line of text are the carriage return and the line feed. They are treated identically, either one generating a new line and four initial spaces. The spaces are required because lines are later numbered by the full editing routine, and an area of the page must be allocated for the line numbers. The spaces do not become part of the actual text of the message. Consecutive line delimiters are stored in the PDP-5 as a single delimiter and, upon printing, appear as a single delimiter.

Whenever paper tape input is selected by a sense switch option, the carriage return, line feed, and four space characters are required on the input tape, and the codes that stop the paper tape reader (X-off) must be present in appropriate locations. Blank tape, rubout, and leader codes* are ignored.

During the composition of a message, the horizontal arrow symbol (←) may be used alone or with the question mark or exclamation

---

* Blank tape consists of tape with none of the information channels punched and has as its octal representation 000. The respective representations of rubout and leader are 377 and 200.

point for "short editing". This short editing is always available to the user; in the event that short messages are expected, it is the only editing available. There are three uses of the arrow:

1) "←", a string of one or more arrows such that the character immediately following the last arrow is not "?" or "!". This causes deletion of the arrows and the last, the last two, or the last N characters according as there are one, two, or N arrows.
2) "←?". This causes deletion of the entire message.
3) "←!". This causes deletion of the current line of the message, unless the only symbols on the current line are the arrow and exclamation point, in which case the preceding line is deleted.

The horizontal arrows do not become a part of the input message, but "!" and "?" may occur in a message if they are not immediately preceded by an arrow.

Note that it is possible to dictate the deletion of characters or lines that do not exist, by providing more occurrences of "←!" than there are lines or of "←" than there are characters in in the message. Whenever this is done or the "←?" option is used, the editing routines print "RE-ENTER MESSAGE:" at the remote console, indicating a fresh start.

All characters other than those mentioned above are illegal. If an illegal character is transmitted to the PDP-5, the computer ignores the character insofar as the message being formed is concerned. A bell is rung at the remote station, to indicate to the user that

some illegal character has been transmitted. In the event that the transmitting station is operating in full-duplex mode, the illegal character is not printed at the operator's console. It is not possible to suppress this printing when half-duplex equipment is in use.

One exception is the vertical form feed, which is treated as an illegal character but is registered at an originating station even if that station is operating full-duplex.

Some messages generated by users are always short, "YES < >" or "NO < >", or numbers, or the like. These are subject to short editing only. Other messages are subject to the full editing described below. Except for the communications with the full editing routine itself, which are subject to short editing only, the 7040 selects the editing options in normal mode. In practice mode, in which the 7040 is not used, full editing of the test messages always occurs.

4.2.2. Full Editing

In addition to the always present short editing, some user generated messages may be printed back, proofread and corrected after they have been completed (that is, after the " < >" has been typed). Messages subject to this kind of editing are potentially long ones, such as those specifying retrieval criteria.

After a complete message requiring full editing has been typed the operator receives the inquiry "PRINT? :=". He must answer

either "YES < > or "NO < >"*. If he answers "NO < >", then his message is transmitted to the 7040 for processing (or printed at the remote console if practice mode is being employed).

Suppose the user answers "YES < >". The message will then be printed at the remote Teletype exactly as it stands after the deletions caused by any short editing performed when the message was originally entered. Lines will be numbered by consecutive decimal numbers, the first line being numbered one. The format is: first a two digit decimal number, then a right square bracket, then a space and then the line of the input message. The square bracket is chosen because it is available; and, because it is not a legal character, it is unlikely that the line numbers will be confused with the text of the message. The line numbers do not become a part of the message itself. Recall that during the initial typing of a message, when a line delimiter (carriage return or line feed) is struck, the new line starts with four spaces; these spaces reserve the area in which the line numbers now appear for purposes of editing.

After the message has been printed, the question "CORRECTIONS? :=" is asked. Again, this must be answered "YES < >" or "NO < >". In the event that the answer is negative, the message is transmitted to the 7040 (or printed at the remote station if practice mode is selected).

---

* Short editing is allowed here, so "N ←YES < >" is allowed. "N < >" and "Y < >" are acceptable synonyms for "NO < >" and "YES < >". If the reply is not recognized, the operator is requested to "ANSWER 'YES' OR 'NO'.", and then "PRINT? :=" is generated again, so that the user may re-enter his answer to the question.

If the answer given to "CORRECTIONS? :=" is "YES < >", the user next receives the message "LINE NO.:=". There are two possible forms of answer to this query. First, the user may reply "ALL < >". If he does so, the entire message is deleted, and the operator is instructed to "RE-ENTER MESSAGE :=". When he has done so, he will again be asked "PRINT? :=".

As an alternative to answering "ALL < >", the user may type a one- or two-digit number, specifying a line of the message. If the message consists of N lines numbered from 1 to N, the line specified may be one between zero and N+1. The two extra positions are added in order that a line may be added before the original first line, or after the last one. Any other number will be rejected as an illegal response, and the request "LINE NO.:=" will again be asked.

After the line number has been specified, the user must respond to the request "LINE(S):=". He then indicates the replacement for the line indicated. Such replacements may be considered to fall into three categories, although to the user the division is an implicit rather than an explicit one.

First, the user may respond to "LINE(S):=" by typing only "< >". This indicates that a line is to be deleted, and that nothing is to be added in its place.

Secondly, the user may type one line, that is, a string of characters containing no line delimiters and followed by "< >". This line then replaces the specified line in the message.

Thirdly, more than one line may be typed. This causes the specified line of the original text to be replaced by the several new

lines specified. If reference is made to the same line before the revised message has been printed, the reference is in effect to all the new lines replacing the single line. This is because in all cases, the line numbers remain unchanged regardless of any editing performed until the message is next printed by use of an affirmative response to the query "PRINT? :=".

After the question "LINE(S):=" has been answered, the user is asked "MORE? :=", which is again a question to be answered "YES < >" or "NO < >". If he answers "NO < >", the editing programs return to the point where "PRINT?:=" was asked. If "MORE? :=" is answered "YES < >", the next question asked of the user is again "LINE NO.:=". New line numbers are assigned only after "PRINT? :=" is answered in the affirmative.

The message is transmitted to the 7040 (or printed at the remote terminal if practice mode is being used) only when either "PRINT? :=" or "CORRECTIONS? :=" is answered "NO < >".

4.2.3. Examples of Editing

In the following examples, full editing in the practice mode is illustrated. As the short editing options are always available, they also are illustrated. Figure 19 shows the use of " ←" and " ←!" in the initial message formation, replacement of a line under the "CORRECTIONS? :=" command, the use of short editing options in formulation of an answer to a query directing full editing.

The timer-activated disconnection of a remote user and warning messages which result if the remote station sends no information for a period of three minutes is illustrated in Figure 21.

```
ENTER MESSAGE:

    RETRIEVE ($A1 RUBINOV←FF & B←SB←←($B AUTOMATA +
    SEQUENTIAL MACHINES) + ($A1 PAULL AN←←& UNGRE & $B
    STATE REDUCTION FINITE STATE AUTOMATA←!
    STATE REDUCTION SEQUENTIAL MACHINES)<>

PRINT?  :=    YES<>

YOUR MESSAGE IS:

01] RETRIEVE ($A1 RUBINOFF & ($B AUTOMATA +
02] SEQUENTIAL MACHINES) + ($A1 PAULL & UNGRE & $B
03] STATE REDUCTION SEQUENTIAL MACHINES)

CORRECTIONS?  :=    YES<>

LINE NO.  :=    2<>
LINE(S):

    SEQUENTIAL MACHINES) + $A1 PAULL & UNBER←←GER & $B
    <>

MORE? :=    YE←←NO<>
PRINT?  :=    MO<>
ANSWER "YES" OR "NO".
PRINT?  :=    NO<>
THIS IS THE MESSAGE WHICH WOULD BE TRANSMITTED TO THE 7040:

RETRIEVE ($A1 RUBINOFF & ($B AUTOMATA +
SEQUENTIAL MACHINES) + $A1 PAULL & UNGER & $B
STATE REDUCTION SEQUENTIAL MACHINES)
```

EDITING EXAMPLES - PART 1

Figure 19

```
ENTER MESSAGE:

    TETRIEVE---------
RE-ENTER MESSAGE:

    RETRIEVE SA1 BAUER, WALTER F. & BROWN, J. HARVER
    & CARR, JOHN W. III & GARDNER, J. & HOFFMAN, ROSLY
    N. & PERKINS, ROBERT<>


PRINT?  :=    YES<>


YOUR MESSAGE IS:

01] RETRIEVE SA1 BAUER, WALTER F. & BROWN, J. HARVER
02] & CARR, JOHN W. III & GARDNER, J. & HOFFMAN, ROSLY
03] N. & PERKINS, ROBERT


CORRECTIONS?  :=    YES<>


LINE NO.  :=    1<>
LINE(S):

    RETRIEVE SA1 BAUER, WALTER F. & BROWN, J. HARVEY
    <>


MORE? :=    YES<>


LINE NO.  :=    5<>
ILLEGAL RESPONSE.
LINE NO.  :=    4<>
LINE(S):

    & POLLMAR, CARL & RAZGUNAS, L.
    & RICHARDSON, VIRGINIA & WRIGHT, J. B.<>
```

EDITING EXAMPLES - PART 2

Figure 20

```
MORE? :=     NO<>
PRINT?  :=     YES<>

YOUR MESSAGE IS:

01] RETRIEVE $A1 BAUER, WALTER F. & BROWN, J. HARVEY
02] & CARR, JOHN W. III & GARDNER, J. & HOFFMAN, ROSLY
03] N. & PERKINS, ROBERT
04] & POLLMAR, CARL & RAZGUNAS, L.
05] & RICHARDSON, VIRGINIA & WRIGHT, J. B.


CORRECTIONS?  :=     NO<>
THIS IS THE MESSAGE WHICH WOULD BE TRANSMITTED TO THE 7040:

RETRIEVE $A1 BAUER, WALTER F. & BROWN, J. HARVEY
& CARR, JOHN W. III & GARDNER, J. & HOFFMAN, ROSLY
N. & PERKINS, ROBERT
& POLLMAR, CARL & RAZGUNAS, L.
& RICHARDSON, VIRGINIA & WRIGHT, J. B.


  ENTER MESSAGE:

  YOU HAVE ONE MINUTE TO RESPOND.

  EXCESSIVE DELAY.  CONNECTION TERMINATED.
```

EDITING EXAMPLES - PART 3

Figure 21

Figures 20 and 21 show several features of the editing routines. First shown is the effect of an attempt to delete more characters than exist in a message. Line replacement using the "CORRECTIONS? :=" option of full editing is shown, including the addition of line number N+1, replacement of one line by more than one line and an attempted replacement of a line not numbered between zero and N+1.

4.3. Interface with the Retrieval Programs

In the foregoing sections the communication routines were described from a user's point of view. A programmer writing for the 7040 sees the PDP-5 computer and the user of the system through a 7040 subroutine that controls the exchange of information between the two machines and the remote user.

This subroutine has four arguments and may be called from either FORTRAN or MAP programs. One argument is the starting address for an input buffer region. Information generated by the user and edited by the PDP-5 editing routines is placed in this buffer when it is transmitted from the PDP-5 to the 7040. The input buffer contains 450 words of 6 characters each, and the characters are represented in IBM six bit internal code. A second argument of the subroutine identifies the location of an output buffer of 900 words to be sent to the remote user, these words having the same code format as the input buffer words. In either buffer the first character of the first word of the buffer is the first character of the message. Since messages are of variable length (but no longer than the buffer capacity), a special symbol is used to mark the end of any message.

The timing mechanism that disconnects a station remaining silent for three minutes following an input request is mechanized in the PDP-5. It is necessary that the 7040 programs be able to test for this condition, in order to determine when for some reason the user has quit or been disconnected without signing off. A third argument of the communication routine conveys this information.

The final argument is set by all 7040 programs that call on the communication routine, and is used to indicate the action required. There are seven possible values of this argument, indicating seven possible required actions:

- Initialize the locations used for job termination, and enable the traps in the 7040 that respond to the PDP-5. This is done only once.
- Wait for a remote user to dial into the system.
- Send the contents of the output buifer to the user's terminal to be printed, and disconnect the remote terminal when the printing is completed.
- Send the contents of the output buffer to the user, and print them at the remote terminal. Then wait for the user to send back a message, using only short editing. Upon return from the subroutine, this message may be found in the input buffer.
- Perform the same functions as are described immediately above, except that the full editing options are made available to the user.

- Send the contents of the output buffer to the user, printing them at the remote conscle. No input from the remote terminal is received.
- Print a message at the remote console indicating that the 7040 is no longer available, disconnect the remote Teletype and restore the 7040 traps so that the PDP-5 cannot interrupt the 7040. The output buffer is not used, as the message printed is permanently stored in the PDP-5.

All functions except for the last one operate in the following manner: when the subroutine is called, control remains with the subroutine until the specified activity has been completed, insofar as the calling program is concerned. When the calling program receives control, the activity has been completed. The function described last is an exception to this, and is used only when the entire system is being shut down. It is also an exception in that it may be called at any time, even interrupting a previous call of the same subroutine with a different set of arguments.

In the case of all functions described except for the first and last, the subroutine call acts as a portal to the background program. The background work is performed while messages sent to the remote user are printed, while a user formulates and edits a message, or while the system waits for a user to dial in.

5.1. The Information System

The elements of the information system discussed in the preceeding chapters were the data file and its list retrieval mechanism, the editing and communications provisions, the list operations, and the linear and temporary list facilities. In this chapter the other components of the system will be considered, namely those that allow the system to accept a user's retrieval request (after processing by the editing routines), to transform that request into a set of list operations, to execute those operations, thereby producing a list of accession numbers for documents that satisfy the original request, and finally to allow the user to obtain selectively information concerning the documents so retrieved.

In the discussion below, an overall explanation of the retrieval request and the user's options in selecting the information returned to him will first be given. Following this is a discussion of the three routines that, together with the communications, data handling and list manipulation routines, make up the system.

5.2. The Retrieval Request

A retrieval request consists of the word "RETRIEVE", followed by some specification of documents to be retrieved. This request is processed by a subroutine named "retrieve," which is discussed in greater detail in Section 5.4. The subroutine generates a list of accession numbers of documents meeting the specifications of the request. The

list of accession numbers is then transmitted to a second subroutine, which allows the user to edit the actual document information to be given him. Both of these subroutines operate under the control of an executive; the second subroutine and the executive are discussed in Sections 5.6. and 5.7. and the effects of the second subroutine apparent to the user are covered in Section 5.3.

The specification part of a retrieval request consists of index terms, delimiters, sector designators and operators. Parentheses may be used for grouping. The allowable operators are &, ↑ and + , corresponding to the English AND , AND NOT, and inclusive OR. The first two are of equal rank higher than that of the third. Operators of equal rank are executed from left to right. (It will become apparent below that the space is sometimes also an operator in some contexts, but for the present it will be considered to occur either as part of a multiple word index term or as an ignored character added for ease in reading.)

Used to indicate the context of the index terms, the allowable sector designators are:

$A0	Date of indexing and indexer's code
$A1	Author
$A2	Date of issue
$A3	Title
$A4	Editors

$A9	Collection in which article appears
$B	Index terms
$C	Added information

In a retrieval request, the sector designators modify index terms. A designator's scope extends from the designator to the right, remaining in effect until the occurrence of another designator. The scope of designators is not influenced by parenthetical groupings.

Consider, for example, the request:

RETRIEVE $A3 RUNCIBLE I & $B (STATISTICAL FUNCTIONS

+ STANDARD DEVIATION) ↑ BUSINESS ORIENTED < > .

Figure 22 shows a Venn diagram illustrating the set of documents specified by the request.

5.3. Options on Information Returned

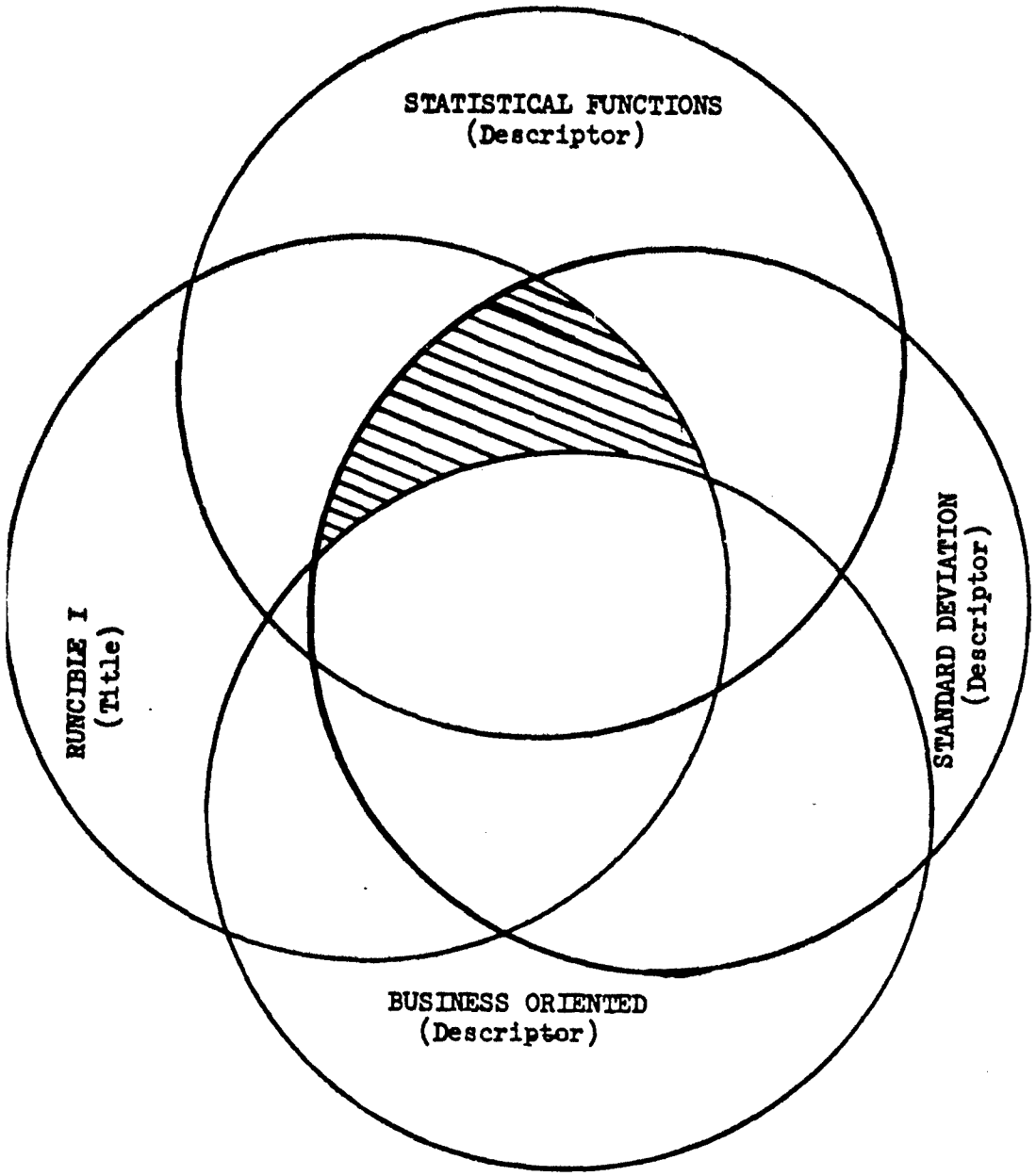
When the user has completed formulation of his retrieval request (including any editing necessary), the information system executes the request. The user is told the number of documents found. Assuming that the number is not zero, the user then selects the types of information that he wishes to have printed for each document (if, in fact, he desires any information returned). For the purposes of selecting the information categories to be returned, the sectors are divided into three broad categories as is done in the formulation of the retrieval request: A, B and C. Category A has ten subdivisions rather than the six allowed in retrieval. These are identified by the digits zero through nine and are, in order: date of indexing and

indexer's code, author, date of issue, title, issuers, page size, number of illustrations, and number of pages. Categories B and C refer, as before, to concept descriptors and added information codes.

After execution of a retrieval request and notification at the remote station of the number of documents retrieved, unless that number is zero, the remote operator is asked if he wants any information concerning the documents printed. If he answers "NO < >", the user is asked to enter another retrieval request. If he answers "YES < >", he is told to indicate the desired information categories. He is given appropriate cues (see examples in Section .4.), and in addition to answers of "YES < >" and "NO < >", the answers "ALL < >" and "FORGET < >" are allowed. The latter is employed when the system user feels that he has made an error, and wishes to enter a new search command. "ALL < >" is a synonym for "YES < >", except that it may be used for recovery from a certain error, as illustrated in Figure 25 of Section 5.4.

If no information is requested under any of the three categories, only accession numbers will be printed at the remote station.

If a user enters a series of search requests, he may have the same information returned for every request. Therefore, once having indicated the type of information to be printed, the user is always asked first if he wants the same information categories as before. If he answers "YES < >", he does not have to respecify information already given.



SCHEMATIC ILLUSTRATION OF THE RETRIEVAL REQUEST "RETRIEVE $A3 RUNCIBLE I & $B (STATISTICAL FUNCTIONS + STANDARD DEVIATION) ↑ BUSINESS ORIENTED < > .

Shaded area refers to documents specified

Figure 22

5.4. Examples of System Operation

In Figures 23 through 27, a complete "run" of the system is shown. The only part of the dialogue not covered in previous Sections is the command "END < >", which is used to signal the end of a run.

Figure 23 shows first the printing of "STANDBY" at the remote console until the 7040 computer becomes available, at which time the system obtains a valid user identification number, search mode is established, and a retrieval request is entered. This request calls for documents by specifying the intersection of three authors names, and one such document is found. The operator indicates that he wishes to have only authors' names and the title of the document printed, and this is done.

The system then accepts another retrieval request (shown in Figure 24). This one specifies documents by the intersection of a descriptor and an author's name. The same information as was returned previously is called for. Note that "CARR" alone specifies both "CARR, JOHN W. III" and "CARR, MARLENE".

Figure 25 shows a retrieval request used as an example in several parts of this chapter, as it includes the AND, AND NOT, and OR operators, as well as concatenation.

The operator indicates that he wishes to change the information categories to be returned. Although he first answers "NO < >" to "ALL $A? :=", he later specifies that all information in category A is to be printed by answering "ALL < >" to "ANY $A? :=". Subcategories A4 and A9 are not printed because the names of editors are not specified for the document retrieved, and it is not part of a larger collection.

```
STANDBY.
STANDBY.
STANDBY.
STANDBY.
STANDBY.

I AM :=                 003333<>

THE OPERATING MODE IS :=                    SEARCH<>

YOU MAY PROCEED.:=      RETRIEVE $A1 PATTERSON,
    G.W. & CARR, J.W. III & HOPPER, GRACE<>

PRINT?  :=    NO<>

000001 'REFERENCES' HAVE BEEN RETRIEVED.

PRINT SOME?  :=     YES<>

INDICATE SECTOR INFO. DESIRED. (ANSWER 'YES', 'NO', 'ALL', OR 'FORGET').

ALL $A?  :=     NO<>

ANY $A?  :=     YES<>

GIVE SECTOR DIGITS  :=    1,3<>

 $B?  :=    NO<>

 $C?  :=    NO<>

 ACC. NO.:  1
A3 FIRST GLOSSARY OF PROGRAMMING TERMINOLOGY- REPORT TO THE ASSOCIATION
A3  FOR COMPUTING MACHINERY
A1 ADAMS,C W+BACKUS,J W+CARR,J W III+OSBORN,R F+PATTERSON,G W+SVIGALS,J+
A1 WEGSTEIN,J+HOPPER,GRACE MURRAY

                    THAT'S ALL.
```

OPERATION OF INFORMATION SYSTEM - PART 1

Figure 23

```
YOU MAY PROCEED.:=      RETRIEVE $A1 CARR & DIAGNOSTIC
    PROGRAMMING-?
RE-ENTER MESSAGE:

    RETRIEVE $A1 CARR & $9 DIAGNOSTIC PROGRAMMING<>


PRINT?  :=    NO<>

000003 'REFERENCES' HAVE BEEN RETRIEVED.

PRINT SOME?  :=     YES<>

SAME INFORMATION CATEGORIES AS BEFORE?  :=    YES<>

 ACC. NO.:  123
A3 PROGRAMMER'S REFERENCE MANUAL FOR KOMPILER 3 AUTOMATIC CODING SYSTEM
A3 FOR THE IBM 704
A1 CAPPS,JOAN+CARR,MARLENE+ELSWORTH,A KENTON+HUDSON,JOHN W+HUGHES,ROBERT
A1  A+KREIDER,LEROY+KUHN,ROBERT+MONK,DOROTHY+TIEDE,KENNETH


 ACC. NO.:  40
A1 PERKINS,ROBERT+CARR,JOHN W III+BROWN,J HARVEY
A3 EASIAC, A PSEUDO COMPUTER-- A PAPER PRESENTED AT THE ANNUAL MEETING O
A3 F THE ACM, PHILADELPHIA, PA, 14 SEPT 1955


 ACC. NO.:  41
A3 ASSORTED NOTES FOR MATHEMATICS 173, 'METHODS IN HIGH-SPEED COMPUTATIO
A3 N'
A1 BAUER,WALTER F+BROWN,J HARVEY+CARR,JOHN W III+GARDNER,J+HOFFMAN,ROSLY
A1 N+PERKINS,ROBERT+POLLMAR,CARL+RAZGUNAS,L+RICHARDSON,VIRGINIA+WRIGHT,J
A1  B

                    THAT'S ALL.
```

OPERATION OF INFORMATION SYSTEM - PART 2

Figure 24

```
YOU MAY PROCEED.:=      RETRIEVE $A3 RUNCIBLE I 4 (SS STATISTICAL
    FUNCTIONS + STANDARD DEVIATION)! BUSINESS ORIENTED<>

PRINT?  :=    YES<>

YOUR MESSAGE IS:

01) RETRIEVE $A3 RUNCIBLE I 4 (SS STATISTICAL
02) FUNCTIONS + STANDARD DEVIATION)! BUSINESS ORIENTED


CORRECTIONS?  :=    NO<>

000001 'REFERENCES' HAVE BEEN RETRIEVED.

PRINT SOME?  :=     YES<>

SAME INFORMATION CATEGORIES AS BEFORE?  :=    NO<>

INDICATE SECTOR INFO. DESIRED. (ANSWER 'YES', 'NO', 'ALL', OR 'FORGET').

ALL $A?  :=     NO<>

ANY $A?  :=     ALL<>

 $B?  :=    YES<>

 $C?  :=    NO<>

 ACC. NO.:  60
A0 8/6/65+JB
A2 SEPT 1959
A3 MANUAL- STATISTICAL EXTENSIONS FOR RUNCIBLE I (FOR BASIC IBM 650 MULT
A3 IPASS MODE ONLY
A1 ALANEN,J D+HAYNAM,GEORGE E+LEONE, FRED C+LYNCH,W C+MONCK,D+OPASKAR,C+
A1 PETZNICK,G W+SPERONI,J
A5 CASE INSTITUTE OF TECHNOLOGY, COMPUTING CENTER AND STATISTICAL LAB (I
A5 N COLLABORATION), CLEVELAND, OHIO
A6 22X28 CM
A7 0
A8 PP NC 110
B  *SOFTWARE+*IBM 650 (C.M.)+*RUNCIBLE I (P.L.)+GENERAL DESCRIPTION+*PRO
B  GRAMMING+*LIBRARY OF ROUTINES+SPECIFICATIONS+*STATISTICAL FUNCTIONS+S
B  TATISTICAL DISTRIBUTIONS+STATISTICAL ANALYSIS+NUMERICAL ANALYSIS+POLY
B  NOMIALS+SIMULTANEOUS EQUATIONS+MULTIPLE PRECISION+FLOATING POINT+MATR
B  IX MANIPULATIONS AND CALCULATIONS+PUNCHED CARDS+MACHINE DEPENDENT+SCI
B  ENTIFIC ORIENTED+(IBM 533 CONTROL PANEL)+(CHEBYSHEV APPROXIMATIONS)+(
B  ANALYSIS OF VARIANCE)+(BINOMIAL COEFFICIENTS)+(CHOLESKY ALGORITHM)+(R
B  ANDOM NUMBERS)+(GAUSSIAN ELIMINATION)+(REGULA FALSI METHOD)+(HASTING
B  APPROXIMATION)+(TAYLOR'S SERIES)+(T-TEST)+(F-TEST)+(CHI-SQUARE TEST)+
B  (STANDARD MOMENT (STATISTICAL))+(CORRELATION COEFFICIENTS)+(GAMMA FUN
B  CTION)+(POISSON DISTRIBUTION)+(NORMAL DISTRIBUTION)+(BINOMIAL DISTRIB
B  UTION)+(REGRESSION ANALYSIS)

                    THAT'S ALL.
```

OPERATION OF INFORMATION SYSTEM - PART 3

Figure 25

Figure 26 illustrates the system response when information is to be printed, but not information in categories A or B or C. The only information remaining is the set of accession numbers. This example also illustrates the query "MORE? :=" after every fifteen lines. Had the user answered "NO < >", the system would have returned to "YOU MAY PROCEED. :=".

Finally, Figure 27 shows the retrieval of an empty set of documents, and the sign-off command.

The present section completes the description of the information system from a user's viewpoint; the remaining three sections of this chapter describe the mechanization of the retrieval process, the information returning routine, and the executive under which the system operates.

5.5. The Retrieval Process

The retrieval subroutine, using the data file described in Chapter 2 and the list routines described in Sections 3.3. and 3.4., operates on an input string specifying a set of documents and generates a list of accession numbers of documents meeting the specifications. A preliminary discussion of the retrieval routine is given in reference 20.

The input string upon which the retrieval routine operates is the original retrieval request, with the first word ("RETRIEVE") deleted, and a special symbol (77_8) added to indicate the end of the string. The executive routine supplies this input string.

In the processing of this string, the space acts as an operator (concatenation) only when it falls between two adjacent words

```
YOU MAY PROCEED.:=      RETRIEVE $B LANGUAGE<>

PRINT?  :=    NO<>

000257 'REFERENCES' HAVE BEEN RETRIEVED.

PRINT SOME?  :=     YES<>

SAME INFORMATION CATEGORIES AS BEFORE?  :=    NO<>

INDICATE SECTOR INFO. DESIRED. (ANSWER 'YES', 'NO', 'ALL', OR 'FORGET').

ALL $A?  :=     NO<>

ANY $A?  :=     NO<>

 $B?  :=    NO<>

 $C?  :=    NO<>

ACCESSION NUMBERS FOUND:

100    101    102    103    104    105    106    107    108    109    110-1
110-2  110    111    112    113    114-1  114-2  114-3  114    115-1  115-2
115-3  115    116    117    118    119    120    121    122    123    124
125    126    127    128    129    130    131    132    133    134-1  134-2
134    135    136    137    138    139    140    141    150    151    1522
152    153    154    155    156    157    15     163    164    165    166
168    169    16     170    171    172    173    176    17     180    182
183    186    187    188    190    191    192    195    197    199    19
200    201    202    203    204    206    208    209    20     212    213
215    216    219    21     220    221    222    224    227    229    22
230    231    232    233    234    235    236    237    238    23     245
246    247    248    249    24     250    252    253    254    255    256
257    258    259    260    261    262    263    264    265    266    267
268    26     270    271    272    273    274    276    277    278    279
281    282    283    284    285    286    287    288    289    28     290

MORE?  :=     YES<>

ACCESSION NUMBERS FOUND:

293    294    295    296    297    298    299    29     300    301    302
303    304    306    308    309    33     310    311    312    313    314
315    31     33     34     351    35     36     37     38     39     41-5
41-6   41-7   41     43     44     45     46     47     48     50     52
53     54     55     56     57     58     59     60     61     62     63
64     65     66     67     68     69     70     71     72     73     74
75     76     77     78     79     80     81     82     83     84     85
86     87     88     89     90     91     92     93     94     95     96
97     98     99

                    THAT'S ALL.
```

OPERATION OF INFORMATION SYSTEM - PART 4

Figure 26

```
YOU MAY PROCEED.:=      RETRIEVE $A1 RUBINOFF, MORRIS<>

PRINT?  :=    NO<>

NO 'REFERENCES' HAVE BEEN RETRIEVED.

YOU MAY PROCEED.:=      END<>

PRINT?  :=    NO<>

YOU HAVE GIVEN THE END SIGNAL.

CONNECTION TERMINATED.
```

OPERATION OF INFORMATION SYSTEM - PART 5

Figure 27

forming an index item; otherwise it is ignored. Periods, commas and carriage returns are syntactically equivalent to spaces. The space is first in the hierarchy when it acts as an operator.

Multiple-word index terms are deconcatenated into single index items, separated by space operators. Every index item is prefixed with a single character that designates the sector (or context) of the index term from which the index item was formed:

Input Sector Designator	Index Item Prefix
$A0	0
$A1	1
.	.
.	.
$A5	5
$A9	9
$B	T
$C	I

Consider the command:

RETRIEVE $A3 RUNCIBLE I & $B STATISTICAL FUNCTIONS

+ STANDARD DEVIATION) ↑ BUSINESS ORIENTED > .

The executive routine, after replacing the end of message symbol (< >) by a special single character and deleting the word "RETRIEVE" from the string, calls the retrieval subroutine and transmits to it the slightly modified retrieval request. The retrieval subroutine takes the input string and transforms it to a list (or "stack") consisting of an end symbol, parentheses, operators and prefixed index items. This stack is in early operator Polish suffix form, and the

transformation is a standard one[9] except that the index items must be prefixed according to the sector designators and the space is considered to be an operator only when it occurs within a multiple-word index item. An example, the stack corresponding to the retrieval request above, is shown in Figure 28.

After the stack has been formed, it is processed from the top down. The occurrence of any two list names immediately above an operator results in a call to the subroutine that mechanizes that operator. The two lists act as arguments, and the two list names and the operator are replaced in the stack by the name of a temporary list, which list contains the results of the operation. (Since the decoding mechanism will return the name of a null list if an index item is not in the system, no special steps need be taken for this case.)

As is shown in Figures 28 and 29, this process continues until all operations have been performed.

If it is impossible to form the stack, owing to a syntactical error in the input string (the user's request), an error message is printed and the user is allowed to re-enter the search request.

5.6. Data Return

The retrieval routine described in the previous section generates a list of accession numbers of documents satisfying the criteria specified in the search request. The user has control over the actual data returned, as described in Section 5.3. If he elects to receive accession numbers only, they are immediately obtainable in the last temporary list generated by the retrieval routine.

When more than the accession numbers is desired, the linear

Initial Stack

1RUNCIBLE

1I

Δ

TSTATISTICAL

TFUNCTIONS

Δ

TSTANDARD

TDEVIATION

Δ

+

&

TBUSINESS

TORIENTED

Δ

↑

1. Concatenation forms temporary list 1 from inverted lists 1RUNCIBLE and 1I.

2. Concatenation forms temporary list 2 from inverted lists TSTATISTICAL and TFUNCTIONS.

3. Concatenation forms temporary list 3 from inverted lists TSTANDARD and TDEVIATION.

4. Union forms temporary list 4 from temporary lists 2 and 3.

EXAMPLE OF FUNCTION OF RETRIEVE SUBROUTINE
First Four Operations

Figure 28

Results after first four operations

Temporary list 1

Temporary list 4

&

TBUSINESS

TORIENTED

Δ

↑

5. Intersection forms temporary list 5 from temporary lists 1 and 4.

6. Concatenation forms temporary list 6 from inverted lists TBUSINESS and TORIENTED.

7. Intersection with complement forms temporary list 7 from temporary lists 5 and 6. This is the final output of subroutine retrieve

EXAMPLE OF FUNCTION OF RETRIEVE SUBROUTINE

Final Three Operations

Figure 29

file is used. This file, described in Section 3.5., contains images of the original document data in the format described in the Appendix*. Every list in this file corresponds to a document (preceded by a dollar sign, in order to avoid the existence of inverted and linear lists with identical names). The data in the list are the original data characterizing the document.

Thus, in order to obtain the information in various categories (and subcategories), the accession numbers generated by the retrieval process are used. When prefixed by a six-bit constant code, accession numbers become names of lists. The linear lists so named give the data required, and every linear list entry is tagged to indicate its sector. The information to be printed at the remote station is selected by a simple scan of the appropriate linear list data. As these data are in image format, only character positions corresponding to the sector designation position (column 1) in the card need be checked.

5.7. The Executive

It is the function of the executive routine to control identifying numbers, and to call and execute retrieval and communication routines in the required sequence. A complete description of this routine may be found in reference 6 . A short explanation is included here for completeness.

The processing of background work is governed by executive

---

* Except that columns 73 through 80 do not appear, since this information is present by virtue of the list names and need not be duplicated.

calls to the communications routine, described in Section 4.3. When a remote user calls into the system and types a space character, the PDP-5 traps the 7040, interrupting the background processing. Control then passes from the communications routine to the executive, which requests that the user enter an identifying number and compares this number with a table of authorized user numbers. An unsuccessful comparison will result in disconnection of the user. If the user number is found in the table, the user is asked to specify a mode of operation* and then to enter some retrieval request.

The user's retrieval request is scanned and the first word of the message, which must specify the retrieval routine, is isolated. A comparison scheme, which allows for the correction of minor spelling errors, is employed when the isolated first word of the request is tested. If it is found to be the name of a specified process, the routines that execute that process are called. If there is a close but not exact match on the comparison, the user is asked to verify the first word. If there is no close match the user is asked to re-enter the first word of the request.

When a properly identified request has been found, the appropriate routine is called. Currently available are the retrieval routine and provisions for changing modes and signing off. Many routines will receive arguments in the same way as the retrieval routine. When the retrieval routine is called, an array of alpha-numeric data--the original input request with the first words deleted--is made available to the retrieval routine.

* This feature has been included for future system expansion; currently only the search mode exists.

The executive is notified by the communications routine whenever a user disconnection not ordered by the executive has occurred. If this happens, the executive prepares for a new user. Also, when 7040 time runs out, the computer operator may signal the executive by manual transfer to a specified location in 7040 memory. The executive then informs the remote user that he has run out of time and disconnects the remote station.

6. Conclusions and Suggestions for Future Work

The primitive information system described in this dissertation has been mechanized and found to be a relatively simple system for many persons to use. The characteristics established in Chapter 1 have been achieved and have proved to be desirable for technically oriented users. The system serves as an excellent medium for technical documents, and is currently being used for further experimentation.

The file organization scheme for the mechanized system was based upon results of the study in Chapter 2 relating internal file characteristics and file usage characteristics to memory utilization and response time. Both the linear and the temporary lists, needed as working area in the retrieval process, are embraced within this general file structure.

The inverted and temporary files are processed by the list operation routines, the latter being called by a program not unlike an algebraic language compiler. Thus, a user's retrieval request is stated in terms of simple logical connectives acting on index terms, and is automatically transformed into a series of list operations before execution. The result is a single list that is transmitted to routines that consider it as a list of names of linear lists and selectively transmit data from the named linear lists to the user.

By careful attention to the interfaces between routines, the system has been constructed in a modular manner, so that it may easily be modified and expanded for future experiments.

Particular attention has been paid to features accommodating on-line console operation. For this reason the system has been designed to provide the user with cues during his search, and extensive input editing features have been made available to him. For the same reason, the user

has been given the option of choosing any selection of classes of information displayed at the remote console; to force him to wait for typewriter printout of material that he does not want would destroy the usefulness of real time operation.

Certain improvements are warranted for more advanced systems. Nontechnical personnel have found the retrieval command somewhat more difficult to master. A possible solution to this problem is the incorporation of a simpler retrieval command, perhaps more limited in scope but easier to use. The present command would be retained for the class of users who find that it meets their needs.

In a production system, most applications would require that multiple consoles be provided. This would involve either more buffer space in the smaller computer or more frequent interruption of the larger computer if the present equipment were to be used. Re-entrant coding of the communications and editing routines (or replication of them, less efficient from a memory utilization standpoint) would also be required.

Most production systems would also require some form of mechanization of the conventional "broader term", "narrower term", "used for" and "use", although this could be incorporated into the present file structure if desired.

The basic data file organization does not require modifications for the class of applications considered, even allowing those requirements of advanced systems mentioned above. This does not, however, indicate that the organization scheme is appropriate to applications such as those requiring real time file maintenance, as for example would exist in some inventory applications. The criteria developed in Chapter 2 are valuable in the consideration of different file organizations for various applications.

APPENDIX

INDEXING INFORMATION FORMAT

1. Data Base Input Format

This appendix discusses the requirements on the format of indexing information. The indexing scheme generally follows the procedure described in reference 18 . Document indexing information is recorded on standard forms, and from these forms cards are punched for computer input. The input data described here form the system's data base.

2. General Card Format

Information in all fields is left justified. Cards are divided into four fields as follows:

Column 1 - this indicates the type of information (author, title, etc.) on the card. The possible entries in this column are individually discussed below, and are 0,1,2,3,4,5,6,7,8,9,A,B,C,I, and T.

Columns 2-3 - there are cases when one card is not sufficient to record the information corresponding to one column 1 category. Then columns 2 and 3 are used to indicate continuation to a second, third,..., card. These columns are left blank except when used for continuation.

Columns 4-72 - the bulk of the indexing data (such as authors, index terms, etc.) is recorded here, with appropriate control symbols.

Columns 73-80 - the accession number of the indexed document is punched in columns 73-80 of <u>every</u> card associated

with that document. (This is not difficult since the keypunch operator simply has to set up a control card on the keypunch to allow automatic duplication of these characters after the first card has been punched.

3. Conventions in Processing

- Two or more adjacent spaces are equivalent to a single space.
- The last card of the data deck must have a Z in column 1 and columns 2-80 must be blank.
- The dollar sign must appear only as a control character, and must terminate every data section of every non-continued card, and must also terminate the last card of a group of continued cards.
- Cards in a deck must be grouped by accession number, but within one accession number group, cards may be out of order.
- The plus sign and the slash are discussed below; they are in some cases data-level symbols and in other cases control-level symbols.

4. Column One Codes

Column 1 is used to indicate the sector of the information appearing on the card. Note that in the present system, codes A, B, C and T are treated identically.

Column-1 codes:

Code	Card Information
0	Date of indexing and indexer's code

Code	Card Information
1	Author
2	Date of issue
3	Title
4	Editors, etc.
5	Issuers
6	Page size
7	Number of illustrations
8	Number of pages
9	Collection in which article appears
A	Index terms
B	Index terms
C	Index terms
T	Index terms
I	Added information

5. Specific Sector Formats

Following are descriptions of required formats for the data in various sectors. The formats are listed by the column one, or sector, code.

Code 0. All blanks will be ignored, and code zero cards may not be continued. In the data field of the card must be the date of indexing, written in the form "$\alpha\alpha/\beta\beta/\gamma\gamma$", where $\alpha\alpha$, $\beta\beta$, and $\gamma\gamma$ are one or two digit numbers for the month, day, and year. This must be followed by a plus sign, the indexer's code (no more than six characters), and a dollar sign. The indexer's code may not contain the + or $ symbols.

Codes 1, 3, 4, 5, 9, A, B, C. In these cases, the names of authors, editors, etc., should be in the form last, first. In the case that there is more than one author, editor, agency, etc., these are to be separated by plus signs. Therefore, the plus sign cannot be used on the data level. Multiple spaces are interpreted as single spaces; spaces on either side of a plus sign or comma are syntactically ignored. Periods are treated as spaces. For example, consider the two lines below, which result in identical contributions to the inverted lists:

BEATTY, ROBERT W. + ISHARD, F. + OGUCHI,DR.B$

BEATTY,ROBERT W+ISHARD,F+OGUCHI,DR B$

Code 2. Date of issue must be justified left in column 4. The formats allowed are: αα βββ..β γγγγ, where αα is the day (one or two digits) and may be omitted, βββ..β is the month (three or more letters) and may be omitted, and γγγγ is a four digit year.

Codes 6,7,8. For these codes, all blanks in the data field will be ignored. Cards with these codes should not be continued. The plus sign may be used on the data level, since the only recognized control character is the dollar sign.

Code I. All blanks will be ignored. The plus sign is at the control level. In addition, the slash becomes a control character indicating a repetition of the last alphabetic character, preceded by a plus sign, except in the case of Z-codes.

University of Pennsylvania
The Moore School of Electrical Engineering
Philadelphia, Pennsylvania 19104

UNCLASSIFIED

DESIGN PRINCIPLES FOR AN ON-LINE INFORMATION RETRIEVAL SYSTEM

Scientific Interim

Thomas C. Lowe

December 1966	121	26

AF 49(638)-1421

9769-01
61445014
681304

67-14

AFOSR 67-0423

Distribution of this document is unlimited.

TECH, OTHER

Air Force Office of Scientific Research(SRI)
1400 Wilson Boulevard
Arlington, Virginia 22209

Areas investigated include slow memory data storage, the problem of decoding from an index to a slow memory address, the structure of data lists and data list operators, communications between the human user and the system, processing of retrieval requests, and the user's control over the return of information retrieved.

Linear, linked and inverted file structures are considered. Empirical data from the Repository of the Association for Computing Machinery are used for illustrative purposes. These data are also used in the portion of the decoding mechanism study which deals with the effects of truncation of index terms.

Following the file organization study, the necessary list structures and list operators are designed. An editing language for use by the human operator in communicating with the system is specified, as are requirements for the execution of "background" programs when a user's information retrieval request is not being processed. Finally, a simple sequence of man-machine communications which allow the user of the system to specify what classes of data are to be returned to him is outlined.

DD FORM 1473

UNCLASSIFIED

Unclassified
Security Classification

14. KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
Linear File						
Balanced Trees						
Symbol Trees						
Data List Structure						
Executive						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.