A relational document retrieval system is developed that is primarily concerned with hierarchic tree structures and lateral links. The lateral linkages are relationships that cannot be represented within the tree structure representation. A number of annotated examples are given along with a detailed description of the relational data structure and the programs that manipulate and search this structure. The system was programmed in a string manipulating language designed for information retrieval purposes. (Author)
THE IMPLEMENTATION OF A RELATIONAL DOCUMENT RETRIEVAL SYSTEM

BY

YEAN-HSI CHANG

This work was supported by the Office of Education under Contract OE C-1-7-071213-4557; auxiliary support was provided by Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract DAAB-07-67-C-0199.

Reproduction in whole or in part is permitted for any purpose of the United States Government.

This document has been approved for public release and sale; its distribution is unlimited.
THE IMPLEMENTATION OF A RELATIONAL DOCUMENT RETRIEVAL SYSTEM

BY

YEAN-HSI CHANG
B.S., Chinese Naval College of Technology, 1953

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering, in the Graduate College of the University of Illinois at Urbana-Champaign, 1970

Urbana, Illinois
A relational document retrieval system is developed that is primarily concerned with hierarchic tree structures and lateral links. The lateral linkages are relationships that cannot be represented within the tree structure representation. A number of annotated examples are given along with a detailed description of the relational data structure and the programs that manipulate and search this structure. The system was programmed in ISL, a string manipulating language designed for information retrieval purposes.
ACKNOWLEDGMENT

The author wishes to express his gratitude to Professor R. T. Chien for his guidance and criticism in the preparation of this thesis, and especially to Mr. Fred Stahl, Ph.D. candidate of the Department of Computer Science, for teaching the operation of the CDC 1604 computer, the ILLAR and ISL systems, and his valuable suggestions. Thanks also to Mr. Nickolas Esser for the compiled material in his master's thesis, and to Mr. Karl Kelley for his help in debugging the program. Thanks are extended to Mrs. Sandy Bowles and Mrs. Sherry Kallembach for the typing of this manuscript.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II.</td>
<td>THE USE OF RELATIONAL INFORMATION IN DOCUMENT RETRIEVAL</td>
<td>3</td>
</tr>
<tr>
<td>III.</td>
<td>THE RELATIONAL STRUCTURE DESIGN</td>
<td>9</td>
</tr>
<tr>
<td>IV.</td>
<td>MAJOR FUNCTIONS NECESSARY FOR THE USE OF RELATIONAL STRUCTURES</td>
<td>14</td>
</tr>
<tr>
<td>V.</td>
<td>CONCLUSION</td>
<td>15</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>APPENDIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A.</td>
<td>DESCRIPTIONS OF PROGRAMS</td>
<td>17</td>
</tr>
<tr>
<td>B.</td>
<td>USER'S MANUAL</td>
<td>22</td>
</tr>
<tr>
<td>C.</td>
<td>SAMPLE OF THE HIERARCHIC TREE STRUCTURE USED TO TEST THE PROGRAM.</td>
<td>25</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

Document retrieval systems have been intensively studied in recent years. One of the more interesting aspects in this area is that of file structuring. One particular file structure may be found to be very effective in a particular system unacceptable in another system. A review of some of the well known document retrieval systems shows that there are many problematic aspects still remaining with respect to the file structure. The AESOP system [7] uses hierarchic tree structure to handle the category titles and tables to handle the items in the categories. The BOLD system [8] uses list subject category and matrices to show the relation between documents and index terms. The SMART system [9] uses an inverted file of clusters. But no one has considered relations between documents, such as: two documents could have the same descriptors and yet be discussing entirely different subjects, or conversely if two documents are concerned with the same topic but have entirely different descriptors; also the relationship between certain documents could be conditional upon the intent of the user. These relationships can not be expressed clearly by a simple list or a simple hierarchic tree structure. In view of this fact, Esser [1] has done some theoretical work on hierarchic tree structuring together with lateral links (LL) relating the terms used in the field of coding theory. The work reported here represents the development and implementation of the computational aspects of a relational data file for document retrieval employing the tree structuring a lateral-link concepts.

In order to give an overall picture of a system that employs these techniques described previously and shows the effectiveness of such an approach
a number of annotated examples will be given in Chapter II. Following these examples a detailed description of the relational data structure developed is given in Chapter III. In Chapter IV a formal description of all the features available in system is given.

The Appendix contains a description of the programs, a user's manual and a sample tree structure from the relational model used.

This document retrieval system was programmed in ISL, a string manipulating language designed for information retrieval [2,3,4,5,6].
II. THE USE OF RELATIONAL INFORMATION IN DOCUMENT RETRIEVAL

The relational document retrieval system implemented in this paper is one which uses all the available relations between items of the data base to give the user as much information as possible. It is also designed in such a way that the user needs no technical background in order to do the retrieval. The examples given in this chapter show the system's actual performance. The detail step-by-step description of how to operate the system is given in Appendix B.

The data base area used for this implementation is from the area of coding theory. The names are either the categories or the terms generally used in coding theory texts, e.g., cyclic codes, optimum codes, decoding algorithms, binary codes, etc. Each node in the tree structure represents a topic area.

If the user has in mind a topic in coding theory and he wishes to know more about it, then this program can be used. Suppose that a user has "OPTIMUM CODES" in mind. After the computer has been set up by following the procedure given in Appendix B, the statement "STATE YOUR REQUEST" will appear on the display console as in Figure 1 and will also be typed on the console typewriter. Now, he can type "OPTIMUM CODES" followed by a carriage return. At this point in the program "ABORT" may be typed instead and the program control will be passed to the monitor.

The topic that has been typed will be shown in the display console. The program can be restarted again by typing "ERASE" or the program will continue by typing "GO". After typing "GO", the display console will appear as in Figure 2. Pressing the right-most arrow on the screen with the lightpen,
STATE YOUR REQUEST

Figure 1.

1---+2---+ OPTIMUM CODES

Figure 2.
Figure 3.

Figure 4.
the display console will appear as shown in Figure 3. There are 15 commands, the functions of which are explained in detail in Appendix B. If "DWTREE" is selected with the light-pen, the display console will be as shown in Figure 4.

Suppose that the user wants to know the optimum codes under high rate transmission; he can press the right-most arrow with the light-pen to get Figure 3 shown on the display console again and select "ADD" with the light-pen. By typing "HIGH RATE" followed by a carriage return Figure 5 will be shown if "HIGH RATE" is available in the data file (in this case, it is).

By pressing the right-most arrow and selecting the "PTR2DWN" with the light-pen four times until the picture as shown in Figure 6 is shown on the display-console screen.

Pressing the right-most arrow and selecting "IATLINK" with the light-pen will then present the user the information shown in Figure 7.

If he presses the right-most arrow with the light-pen and selects "GO-ON", he will see the picture shown as in Figure 8. That means there is only one node whose name is "OPTIMUM CODES".

The user may start the retrieval process* if he feels that the topic names found are enough for his interests simply by pressing the right-most arrow and selecting "RETRIEVE" with the light-pen.

If he selects "LISTALL", he will have the whole list shown again. By selecting "PRINT", he will have the whole list printed. But, if he selects "RESTART", he will erase the whole list and start the program all over again.

*Program for retrieval process is not included in this implementation.
1 \quad + \quad 2 \quad - \quad + \quad \text{OPTIMUM CODES}
ITS SUBCATEGORIES IN THE TREE STRUCTURE ARE
TYPES OF OPTIMUM CODES
PROPERTIES OF OPTIMUM CODES
HIGH RATE

Figure 5.

1 \quad - \quad + \quad \text{OPTIMUM CODES}
ITS SUBCATEGORIES IN THE TREE STRUCTURE ARE
TYPES OF OPTIMUM CODES
PROPERTIES OF OPTIMUM CODES
2 \quad - \quad + \quad \text{HIGH RATE}

Figure 6.
OPTIMUM CODES
ITS SUBCATEGORIES IN THE TREE STRUCTURE ARE
TYPES OF OPTIMUM CODES
PROPERTIES OF OPTIMUM CODES

HIGH RATE
THEY MEET THE FOLLOWING CONSTRAINTS
HAMMING BOUND
MEET UPPER BOUND

Figure 7.

THERE IS NO MORE SUCH ITEM AVAILABLE

Figure 8.
III. THE RELATIONAL STRUCTURE DESIGN

In order to best illustrate how the relational information is stored, a representation of the subtree given in Figure 10 will be given. Three structure representations are described here.

A. Node Representation

Each model node consists of two parts as shown in Figure 9. In part one, there are seven fields. Field 1, "NEXT", contains the word address of the next node whose name starts with the same character as this node; field 2, "END", contains a pointer pointing to the byte address of the end of the character string of this node; field 3, "TYPE", contains the information which indicates the type of lateral link function that applies to this node; field 4, "I#", contains the item number of this node; field 5, "UPPER" contains the item number of the node which is at one level up in the hierarchic tree structure; field 6 and 7, "LOWER 1" and "LOWER 2", contain the item numbers of the first and last nodes, respectively, which are at one level down in the hierarchic tree structure. Part two of the node is a variable length string that contains the name of the topic represented by this node.

Figure 11 shows how this information would look for the node given in Figure 10.

B. Lateral Link Representation

These structures consist of three to eight fields depending on which type of lateral link function applies. Field 1 contains the ite
Figure 9. OPTIMUM CODES

Figure 10. Part of the tree structure about the node whose name is "TYPES OF OPTIMUM CODES".
Figure 11. Relations between nodes.

- - - - - related by pointer

- - - - - related by item number
number of the major node, and the rest of the fields contain item numbers of those nodes which required by the lateral link function.
For example, for the node "OPTIMUM CODES", the lateral link representation will be as shown in Figure 12, since the lateral link function LLB9 requires 7 item numbers.

C. Representation of Synonyms

These structures have the same form as the regular node representation. The differences are:

1) Field 3 in part one indicates the synonym property.
2) Field 5 in part one contains the item number of its synonym node.
3) Field 6 and 7 in part one contain zeros, and
4) Part two contains the name of the item which is not in the tree structure but is a synonym.

For example, "MAX. CODES" is not in the tree structure, but its synonym "MAX. DISTANCE SEPARABLE CODES" is. So its representation is as shown in Figure 13.
Figure 12. Lateral link representation for node "OPTIMUM CODES".

Figure 13. Synonym representation for "MAX. CODES".
IV. MAJOR FUNCTIONS NECESSARY FOR THE USE OF RELATIONAL STRUCTURES

The retrieval action of this program is initiated whenever the name of a topic or the leading portion of it is given. There are four major functions in this implementation.

A. The node is located by its name or the leading portion which is given. Nodes which have the same name or the leading portion, or which are related to the node found either by the hierarchy tree structure or by the lateral link functions will also be suggested, printed, and shown on the display console.

B. In any displayed list of nodes two movable pointers are provided which can move up and down independently for use in performing manipulative actions.

C. A lateral link function will be performed upon command on the nodes pointed to by the pointer 1 (PTR 1) if one record is required by that lateral link function, or on the two records pointed to by the pointers (PTR 1 and PTR 2) if two records are required by a lateral function. Lateral functions which require more than two records will be performed on records in the whole list.

D. Nodes which are not in the tree structure but have synonyms will be located. Its synonym will be printed and displayed as described in A.
V. CONCLUSION

The work presented in this paper is concerned with the hierarchy tree structure and lateral link functions in relational document retrieval systems. Fast access and processing have been carefully considered. The flexibility of this implementation is such that it is possible to apply these principles equally well to any other relational document retrieval data base which possesses hierarchy tree structures and lateral link function relations without any modification of the program.
REFERENCES


APPENDIX A

DESCRIPTIONS OF PROGRAMS
Appendix A contains the following information in the order listed below:

(1) MAIN PROGRAM
(2) SUBPROGRAMS
(3) FLOW CHARTS
1. Main Program

This implementation has been programmed on the Control Data 1604 computer at the Coordinated Science Laboratory of the University of Illinois. The main program and the subprograms comprising this implementation were written in the Information Search Language (ISL) [2,3,4,5,6].

There are three portions of the main program in the implementation which will be briefly described here.

a. The first portion of the main program serves to set up the files in the computer memory in the appropriate sequential form. For easy and fast access, directories are built in which the leading addresses of groups and the addresses with respect to item numbers are kept. Twenty-two lateral function directories are also built, each for one of the twenty-two categories of lateral link functions.

b. The second portion of the main program accepts commands from the user and makes a list of the required topic and synonyms and then suggests topics which may be of further interest to the user.

c. The remaining portion of the main program performs further manipulations of the information and executes the desired lateral link functions upon the user's demand.
2. SUBROUTINES

There are 13 subprograms:

a. **SETUP**: This subprogram arranges the records in the appropriate sequence and sets up the directories and the links.

b. **SEARCHA**: This subprogram analyzes a given character string and locates where in the relational file it is located.

c. **SEARCHB**: This subprogram locates a given character string in the portion of the relational file after a given address.

d. **COMPARE**: This subprogram compares two character strings of a given length to see if they are the same.

e. **LFTN**: This subprogram serves to perform the 22 lateral functions, to suggest new related nodes or to suggest the deleted of the node found.

f. **SETLL**: This subprogram puts all the item numbers which will be used in the particular lateral function into the corresponding directory.

g. **MOVPTR**: This subprogram moves the pointer specified from position 1 (pos1) to position 2 (pos2).

h. **CHKINO**: This subprogram checks for the item number specified to see if it is in the list DIRECT. The value 0 in accumulator A indicates success. Fail otherwise.

i. **GETINO**: This subprogram gets the item number of the topic in the record with leading address given and stores it at ITNO.
j. MOVTV1: This subprogram puts the statement with given leading address and ending address in the scope buffer and adjusts the list DIRECT.

k. MOVTV2: When the leading address of the record is given, this subprogram moves the topic title of that record to the scope buffer and sets both lists DIRECT and DIRTV.

l. GETRT1: This subprogram moves the statement between the two given addresses to the scope buffer. When, for the record with leading address at RCDADR, it finds all the related records and puts their topic titles in the scope buffer.

m. GETRT2: This subprogram gets the item numbers pointed to by both pointer 1 (1--) and pointer 2 (2--*), and checks to see if the condition of the lateral link function involved is satisfied. On success, it moves the statement between the two given addresses and the related topic names to the scope buffer.

3. FLOW CHARTS

In the flow charts, the following symbols are used:

a. b : Index designator,

b. B : Designated index register,

c. () : Contents of a register or storage location,

d. (a1,a2) : Contents between byte locations a1 and a2.
APPENDIX B

USER'S MANUAL
1. To start the program:
   (a) Mount tape G-140 (ISL MASTER) on logic unit 1, and tape F-137 (MAIN) on logic unit 2.
   (b) Mount tape F-138 (DATAMAIN) on logic unit 8.
   (c) Press "AUTO-LOAD" button to initiate the program.
   (d) Type "LOAD,2" followed by a carriage return.
   (e) Type "GO,MAIN" followed by a carriage return.

2. To start the retrieval process:
   The program will be ready when "STATE YOUR REQUEST" is typed on the typewriter, and at the same time shown on display console.

   Type a topic name or its leading portion which you are interested in. Follow this by a carriage return. What you have typed will also be shown on the display console.

   Another "carriage return" will give you the opportunity to terminate the program by typing "ABORT" or change the name by typing "ERASE"; and then a "carriage return".

   Your command will be accepted each time the display console is showing information. There are 15 simple commands available:

   (a) PTR1UP: To move pointed 1 (1--*) up the list one step.
   (b) PTR1DWN: To move the pointer 1 down the list one step.
   (c) PTR2UP: To move pointer 2 (2--*) up the list one step.
   (d) PTR2DWN: To move pointer 2 down the list one step.
   (e) DELETE: To delete the topic title pointed to by pointer 1.
(f) ADD: To add a new title at the end of the list.

(g) UPTREE: To find the title of the node one level higher than the one pointed to by Pointer 1 in the tree structure.

(h) DWNTREE: To find the title of the node one level lower than the one pointed to by Pointer 1 in the tree structure.

(i) RESTART: To erase the whole list and start the program over again.

(j) GO-ON: To keep the list in the memory and go on the retrieval process.

(k) LATLINK: To apply the lateral function on the topics pointed to by pointer 1 and also pointer 2 or of the whole list if function requires.

(l) LISTALL: To restore the part of the list kept in the memory and show them all on the TV screen.

(m) PRINT: To print out whatever shown in the TV screen.

(n) THATSALL: To terminate the program.

(o) RETRIEVAL: (Not included in this program).
APPENDIX C

SAMPLE OF THE HIERARCHIC TREE STRUCTURE USED TO TEST THE PROGRAM
DEFINE ALL STATEMENTS AND ARGUMENTS

SETUP (TAPE #8)

TVO = MID

TV1 = TV0 + 10B

PRINT (ASK1, ASK2)

SET PTR1 & PTR2

MOVE (ASK1, ASK2) TO (TV1, TV2)

ISLTV (TV1, TV2)

TSTRINT (Q1, Q2)

EXIT?

PRINT (Q1, Q2)

FIN
MOVE (Q1, Q2) TO TV BUFFER

ISLTV (Q1, Q2)

CR

TSTRING (CCM1, COM2)

EXIT?

ERASE?

TVI = TV0 + 20B
L = Q2 - Q1

CALL SEARCH1

FOUND?

MORE

GET THE RECORD TYPE

EQUIVALENT?

LATLINK

CALL LFTN

DISPLAY

SRRTTV (TV0, TV2)
SELECT WITH LIGHT-PEN

1. - PTR1UP
2. - PTR1DWN
3. - PTR2UP
4. - PTR2DWN
5. - DELETE
6. - ADD
7. - UPTREE
8. - DWNTREE
9. - RESTART
10. - GO-ON
11. - LATLINK
12. - LISTALL
13. - PRINT
14. - THATSALL
15. - RETRIEVE

GOTO ?

1. PTR1U
2. PTR1D
3. PTR2U
4. PTR2D
5. DELETE
6. ADD
7. UPTREE
8. DWNTREE
9. RESTART
10. STOR
11. TLLINK
12. LISTALL
13. PRT
14. THATSALL
15. THATSALL
FAIL \rightarrow ISLTV (FAIL1, FAIL2) \rightarrow CR. ORDER \rightarrow TVOFF

FAILED \rightarrow ISLTV (FAIL3, FAIL4) \rightarrow CR. FIN

THATSALL \rightarrow TVOFF \rightarrow FIN

RESTART \rightarrow TVOFF

TV\# \rightarrow 0
POS PTR1 = 0
POS PTR2 = 0

TVO = MID

TVO = TV1 - 10B

RCADR = (RCADR)

RCADR = 0?

CALL SEARCHB

(A) = 0?

ASK

MORE

FIN

YES

FAILED

NO

YES

NO
DELETE

TVOFF

POS = POSPTR1+1
(B2) = TV#→POS
(B3) = POSPTR1
(B4) = POS

DIRECTTV → DIRTV
+(B4))

(DIRECT+(B3)) = (DIRECT+(B4))

(A) = TV2(w)+3

MOVE (RCDBG, RCDND) TO TV BUFFER

(B5) = (# of words in statement)-1

BUF1 = adr(w) to Be deleted
BUF = adr(w) of Next Statement

TRANSFER WORD FROM BUF to BUF1

(B5) = 0 ?
Yes
(A) = BUF1+3
No
(B5) = (B5)-1

RCDBG, RCDND = adr(s) of title

(A) = 0?
Yes
No

ERI C

36
(DIRTV(B^4)) = (A)

RESET TV1

(B^3) = (B^3) + 1
(B^4) = (B^4) + 1

(B^2) = 0 ?

(B^2) = (B^2) - 1

TVNO = TVNO - 1

COUNT = POSPTR1 - POSPTR2

COUNT < 0?

MOVE PTR2
Up one step

DISPLAY
GET THE EQUIVALENT ITEM NUMBER (ITNO)

\[(B^2) = ITNO\]

DIRECTINO - DIRTNO + (B^2)

\[(DIRECT + B^2) = RCDADR\]

\[(DIRTV + (B^2)) = TR1(w)\]

TVNO = TVNO + 1

MOVE (EQ1, EQ2) To TV BUFFER

RESET TV1

LISTALL

TVOFF

TVNO = MID

PRT

TVC + 2

PRINT (TVO, TV2)

DISPLAY

FIN

END

REG

FINIS
START
SETUP

NEXT = ANT+500(w)
MID = NEXT+20B

READ from Tape (8)

End of file?
Yes
EOF

No

(Q) = MID+50B

CALL POP

CHAR = (A)

FIELD(2) = NXT

(B^2) = CHAR

(A) = (DIRT1+B^2)

(A) = 0 ?

YES

A

NO

BUF = (A)

(A) = (BUF)

(A) = 0 ?

YES

B

NO
(DIRT1 + (B^2)) = NEXT

(BUF) = NEXT

(NEXT) = 0

(B^2) = Item no.

(DIRTN0 + (B^2)) = NEXT

LTYPE = Type of lateral function

LTYPE = LLN?

LTYPE = EQU1?

READIN

Reset NEXT and MID
(B^3) = POSPTR1
RCDADR =
(DIRT+(B^3))
TVO = TV1-10B

START
SUEPROGRAM
LFTN

MOVTV2
(RCDADR)

BUF = RCDADR+2
LTYPE = (BUF)

LTYPE = LLN?
YES
MOVTV1
(LN1, LN2)

LTYPE = LL*?
YES
GETRT1
(S1, S2, DIR)*

NO

NO

L

* |
---|---|---|
LL | N | (S1, S2, DIR) |
---|---|----------------|
LA1 | 2 | (L1, L2, DIRT A1) |
LA2 | 2 | (L3, L4, DIRT A2) |
LA3 | 5 | (L3, L4, DIRT A3) |
LA4 | 3 | (L3, L4, DIRT A4) |
LB1 | 2 | (L5, L6, DIRT B1) |
LB2 | 3 | (L5, L6, DIRT B2) |
LB3 | 2 | (L7, L8, DIRT B3) |
LB4 | 3 | (L7, L8, DIRT B4) |
LB6 | 2 | (L9, L10, DIRT B6) |
LB7 | 3 | (L9, L10, DIRT B7) |
LB8 | 3 | (L11, L12, DIRT B8) |
LB10 | 3 | (L15, L16, DIRT C0) |
LB11 | 7 | (L17, L18, DIRT C1) |
LB12 | 8 | (L17, L18, DIRT C2) |
LB14 | 2 | (L21, L22, DIRT C4) |
SUBPROGRAM LFTN CONTINUE

**

<table>
<thead>
<tr>
<th>LL</th>
<th>N</th>
<th>(S1,S2,DIR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB13</td>
<td>3</td>
<td>(L19, L20, DIRTC3)</td>
</tr>
<tr>
<td>LB15</td>
<td>3</td>
<td>(L23, L24, DIRTC5)</td>
</tr>
<tr>
<td>LB16</td>
<td>3</td>
<td>(L25, L26, DIRTC6)</td>
</tr>
</tbody>
</table>
SUBPROGRAM LFTN CONTINUE

LLB9

GETINO (RCDADR)

N = 8
(B^2) = 0
(B^3) = POSPTR2

A) = (DIRTB9 direction (B^2))

(A) = ITNO?

YES

NO

(B^2) = (B^2) + N

(B^1) = (B^1) - 1

BUF1 = (B^2)
(B^3) = POSPTR2

GETINO (DIRECT + (B^3))

(B^1) = 2

(B^2) = (B^2) + 1

(A) = (DIRTB9 direction (B^2))

(A) = ITNO?

YES

NO

1

(B) = j?

YES

NO

(B^2) = (B^2) + 1
(B^3) = (DIRTB9 direction (B^2))
RCDADR = (DIRTN + B^3)

MOVT V2 (RCDADR)

(B^2) = BUF1 + 7
(B^2) = (DIRECTB9 - DIRTB9)
RCDADR = (DIRECT# - DIRTN)

MOVT V2 (RCDADR)

MOVT V1 (LN3, LN4)

DONE

EXIT
START
SUBPROGRAM
SEARCHA

CALL
POP(Q)

GET ADR(CHAR)

ADR(CHAR) = 0?

YES

FAIL
(A) = 0

NO

CALL
COMPARE

CHAR=(CHAR)

(A) = 0?

NO

YES

(A) = ADDR. FOUND

EXIT

Input: Q, LL
Return: Fail: (A) = 0
Succ: (A) = addr. found
START
SUBPROGRAM SEARCHB

BEG = leading adr(B) of topic

COMPARE (Q, RCDADR, Y-L)

FOUND?

ADR = Adr(B) of next record

(A) = 0?

Return: on SUCCESS: (A) = Adr(w)
FAIL: (A) = 0
Input : AA, BB, LNGTH
Return: (A) = 0

START
SUBPROGRAM
COMPARE

AA = AA + 1
BB = BB + 1

(Q) = AA

CALL
POP
(GET CHAR1)

(Q) = BB

CALL
POP
(GET CHAR2)

CHAR1 = CHAR2?

NO

LNGTH = LNGTH - 1

YES

LNGTH = 0?

YES

(A) = LNGTH

EXIT

NO

SUCCESS
START
SUBPROGRAM SETLL

READ (8)

End of file?

YES

OEF

NO

Store into DIRL

EXIT

START
SUBPROGRAM MOV PTR

(A)=((DIR TV +POS1)-1)

Erase PTR

(A)=((DIR TV +POS2)-1)

Set PTR

EXIT
START SUBPROGRAM CHKINO

RCDADR = (DIRTINO → DIRTNO)

(B^3) = TVNO - 1
(B^2) = 0

(A) = (DIRT + B^2)

YES

(A) = RCDADR?

NO

(B^3) = 0?

NO

(B^3) = (B^3) - 1
(B^2) = (B^2) + 1

YES

EXIT

START SUBPROGRAM MOVT7L

(B^2) = TVNO

(DIRTTV + (B^2)) = TV1

Set blank and carriage return

MOVE (M1, M2) to (TV1, TV2)

TV1 = TV2 + 14B
TV#'s = TVNO#'s

EXIT
SUBPROGRAM MOVTV2

(M3,M4)=Title
(B^2) = TVNO

(DIRTTV+(B^2))=TV1
(DIRECT+(B^2))=M5

Set blank and carriage return

MOVE
(M3,M4) TO (TV1,TV2)

TVNO = TVNO+1

EXIT

15 = leading address of record.

START SUBPROGRAM GETIN

Get content at (M5)+3 in A

ITNO = (A)

EXIT
START
SUBPROGRAM GETRT1

ITNO = ITEM NO.

(B^3) = N - 2
(B^2) = 0

(A)(DIR1+(B^2))

(B^2) = (B^2)+N

YES

(A) = ITNO?

NO

(B^2) = (B^2)+N

YES

EXIT

NO

(B^3) = (B^3)-1

NO

RCDDR = (DIRTNO+(B^4))

MOVTV2 (RCDDR)

(B^3) = 0?

YES

NO

(S1,S2) = Statement
DIR1 = Directory assigned

(DIR1+(B^2))

(B^4) = (DIR1+(B^2))

(B^2) = (B^2)+1
START
SUBPROGRAM GETRT2

ITNO = Item number

(B^2) = 0
(B^3) = N - 3

(A) = (DIR2 + (B^2))

(RCDADR) = (DIRECT + (B^1))

ITNO = Item no.
of (RCDADR)

(B^2) = (B^2) + 1

(DIR2 + (B^2)) = ITNO?

YES

(DIR2 (B^2)) = ITNO

NO

(B^2) = (B^2) + N

(B^1) = POS1 TR2

(RCDADR) = (DIRTNO + (B^1))

MOVT V1
(KS3, KS4)

MOVT V2
(RCDADR)

(B^3) = 0?

YES

NO

(B^3) = (B^3) - 1

MOVT V1
(LN3, LN4)

EXIT

48

53
Distribution List (Con’d.)

Commanding Officer (ARMED-KNT)
U. S. Army Ballistics Research Laboratory
Aberdeen Proving Ground
Aberdeen, Maryland 21005

Technical Director
U. S. Army Aircraft System Command
Aberdeen Proving Ground
Aberdeen, Maryland 21005

Electromagnetic Compatibility Analysis Center (ECMC)
Attn: (AGCST)
North Severn
Annapolis, Maryland 21402

Commanding Officer
U. S. Army Photographer Topographic Lake
Port Salve, Virginia 22606

Director: (EV-5)
Night Vision Laboratory, USAEM
Port Salve, Virginia 22606

Electro-optic Impact Laboratory
Arm Rockwell
Port Salve, Virginia 22606

Dr. John W. H. Snelthede
Institute for Defense Analyses
100 Army-Navy Drive
Arlington, Virginia 22202

Director, Physical & Eng. Sciences Div.
3095 Columbia Pike
Arlington, Virginia 22204

Commanding General
U. S. Army Security Agency
Attn: JABC-TEL
Arlington, Virginia 22209

Dr. John E. Tello, Code 637
Information Systems Branch Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

Commanding General
LeCCDC Institute of Land Combat
800 North Quincy Street
Alexandria, Virginia 22314

NASA Lewis Research Center
21000 Brookpark Road
Cleveland, Ohio 44135

Commanding Director
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

AFAL (AVL), R. W. Lockart
Wright-Patterson AFB, Ohio 45433

Dr. Robert L. Fontana
East, Dept of Electrical Engineering
Purdue University
Lafayette, Ind 47907

Professor Joseph E. Enoe
Chairman, Dept of Electrical Engineering
The University of Michigan
Ann Arbor, Michigan 48104

Dr. G. J. Murphy
The Technological Institute
Rose-Hulman Institute of Technology
P.O. Box 26001

Commanding Officer
Naval Research Facility
Indianapolis, Indiana 46264

Dr. Jay C. Hennon, Head
School of Electrical Engineering
Purdue University
Lafayette, Ind 47907

Professor Joseph E. Enoe
Chairman, Dept of Electrical Engineering
The University of Michigan
Ann Arbor, Michigan 48104

Dean of Electrical Engineering
University of Illinois
Urbana, Illinois 61801

Commandant
U. S. Army Command & General Staff
College
Attn: Acquisitions, Library Division
Fort Leavenworth, Kansas 66027

Department of Electrical Engineering
Rice University
Houston, Texas 77001

HQ AND (AMC)
Brooks AFB, TX 78235

USAFA/SHRM
Brooks AFB, Texas 78235

Mr. R. B. LoVeri
Technical Adviser, Requirements
USAF Security Service
Kelly Air Force Base, Texas 78234

Director Electronics Research Center
The University of Texas at Austin
Eng-Science Bldg 110
Austin, Texas 78712

Department of Electrical Engineering
Texas Technological University
Lubbock, Texas 79409

Commandant
U. S. Army Air Defense School
Attn: Missile Sciences Div., C&G Dept
P. O. Box 9390
Fort Bliss, Texas 79916

Director
Aerospace Flight Sciences
Frank J. Soller Research Laboratory (OMD)
IARF Academy
Colorado Springs, Colorado 80940

Director of Faculty Research
Department of the Air Force
U. S. Air Force Academy
Colorado Springs, Colorado 80940

Major Richard J. Owen
Tenure Associate Professor
Dept of Electrical Engineering
U. S. Air Force Academy
Colorado Springs, Colorado 80940
Academy Library (DSFLB)
U. S. Air Force Academy
Colorado Springs, Colorado 80840

M. A. Rothenberg (STEP-D-A(S))
Scientific Director
Desert Test Center
Bldg. 100, Soldiers Circle
Fort Douglas, Utah 84113

Utah State University
Dept of Electrical Engineering
Logan, Utah 84321

School of Engineering Sciences
Arizona State University
Tempe, Ariz. 85281

Commanding General
U. S. Navy Strategic Communications Command
Attn: SCC-CG-SAE
Fort Huachuca, Arizona 85613

The University of Arizona
Dept of Electrical Engineering
Tucson, Arizona 85721

Cpt C. E. Baum
AM/LG
Los Alamos National Laboratory
Los Alamos, N.M. 87544

Dr. Leo Young
Stanford Research Institute
Menlo Park, Calif. 94025

ADDENDUM:

Dr. H. K. Ziegler, Chief Scientist
Army Member TAC/JSEP (AMSEL-SC)
U. S. Army Electronics Command
Fort Monmouth, New Jersey 07703
A relational document retrieval system is developed that is primarily concerned with hierarchic tree structures and lateral links. The lateral linkages are relationships that cannot be represented within the tree structure representation. A number of annotated examples are given along with a detailed description of the relational data structure and the programs that manipulate and search this structure. The system was programmed in ISL, a string manipulating language designed for information retrieval purposes.
Relational Structure
Lateral Links
Hierarchic Tree Structure
Coding Theory
Information Searching Language
Structure Representation