



5-1989

Retrieve: An Engineering Tool for Searching Remote Sensing and Environmental Engineering Databases

Kay F. Irby

Follow this and additional works at: https://trace.tennessee.edu/utk_gradthes

Recommended Citation

Irby, Kay F., "Retrieve: An Engineering Tool for Searching Remote Sensing and Environmental Engineering Databases. " Master's Thesis, University of Tennessee, 1989.
https://trace.tennessee.edu/utk_gradthes/6242

This Thesis is brought to you for free and open access by the Graduate School at TRACE: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Masters Theses by an authorized administrator of TRACE: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:

I am submitting herewith a thesis written by Kay F. Irby entitled "Retrieve: An Engineering Tool for Searching Remote Sensing and Environmental Engineering Databases." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Engineering Science.

Charles T.N. Paludan, Major Professor

We have read this thesis and recommend its acceptance:

Accepted for the Council:


Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

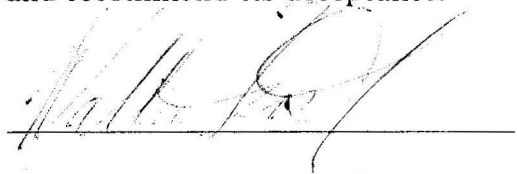
(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Kay F. Irby entitled "Retrieve: An Engineering Tool for Searching Remote Sensing and Environmental Engineering Databases". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Engineering Science and Mechanics.



Charles T.N. Paludan, Major Professor

We have read this thesis
and recommend its acceptance:





Accepted for the Council:


Lew Minkel
Vice Provost
and Dean of the Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee. Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Kay F. Slrly
Date 12/1/88

RETRIEVE: AN ENGINEERING TOOL FOR SEARCHING
REMOTE SENSING AND ENVIRONMENTAL ENGINEERING DATABASES

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Kay F. Irby

May 1989

DEDICATION

To My Parents,
Sylvester “Joe” Irby Jr., and Bernice Irby “Bunny”
I Love You
and to Leslie, Ray, Charles, Pat, and Geri
for believing in me

ACKNOWLEDGEMENTS

The author wishes to express her sincere appreciation to Mr. Wilbur Armstrong, who originally supported the idea of developing an experimental system, and provided invaluable advise on the information retrieval concepts used in this thesis. Many thanks are extended to Dr. Charles T. Paludan, major professor and thesis advisor, for the meticulous attention given to the reporting of the work described in the thesis. Also, the author wishes to express her gratitude to Dr. Frost for taking the time to give support as a committe member during untimely circumstances.

Special thanks are given to Larry Reynolds and Lisa Blanks in the graphics department for their support in the preparation of the thesis, David Modeste and Linda Hall for their patience and assistance in typesetting, and to several secretaries and students who have given support when needed.

ABSTRACT

The design and development of a semi-automatic information retrieval system which features manual indexing, and an inverted file structure is presented. The system requires manual indexing done by an expert in the subject field to ensure high-precision searching. High-recall is achieved through the implementation of the inverted file. The system provides an interactive environment, a thesaurus for normalization of the indexing language, ranking of retrieved documents, and flexible output specifications. The purpose of this thesis is to present the design and development of in-house search-aid software for small document collections intended for Remote Sensing and Environmental Engineering users.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION	1
1.1 Geographical Information Systems	2
1.2 Management Information Systems	4
1.3 Decision Support Information Systems	7
1.4 Database Management Information Systems	8
1.5 Question and Answering Information Systems	9
2. SURVEY OF LITERATURE	10
2.1 Problem Definition	10
2.2 Indexing	11
2.3 Dictionaries	16
2.4 Search Techniques and File Organization	19
2.5 Precision and Recall	21
3. REVIEW OF EXISTING SYSTEMS	27
3.1 Selection Criteria	28
3.2 MICRO-CDS/ISIS	30
3.3 PRO-SEARCH	35
3.4 DIALOG	37
4. IMPLEMENTATION OF DESIGN FEATURES	41
4.1 System Objectives and Methods	41
4.2 RETRIEVE Program Structure	44
5. CONCLUSION AND RECOMMENDATIONS	48
LIST OF REFERENCES	50

APPENDICES 53

 APPENDIX A FLOW CHARTS 54

 APPENDIX B RETRIEVE PROGRAM LISTING 62

 APPENDIX C THESAURUS 83

 APPENDIX D EXAMPLES OF PROGRAM EXECUTION . 97

VITA 109

LIST OF FIGURES

FIGURE	PAGE
1-1 Overlap among Types of Information Systems	3
1-2 GIS Hardware Components	5
1-3 GIS Software Components	6
2-1 Functional Overview of Information Retrieval	11
2-2 Resolving Power of Significant (Medium-Frequency) Words	13
2-3 Document Space Density Examples	15
2-4 Binary Search Example for Key C	20
2-5 Inverted and Direct File Examples	22
2-6 Typical Average Recall-Precision Graph	23
2-7 Graph of Precision Versus Recall for Sample Query of Table 2-3	26
3-1 Software Structure of MICRO/ISIS	32
4-1 Software Design of RETRIEVE	45

CHAPTER 1

INTRODUCTION

An information retrieval system can be defined as a system used to process, store, search, and retrieve information items. The information items may be processed in several forms. Normally, information retrieval systems handle textual data or bibliographic records, whereas data base management and management information systems process very structured data, and question and answering systems use complex knowledge bases and algorithms to evaluate user queries for various subject areas.

Increased technology and widespread popularity of the microcomputer have precipitated the processing of larger volumes of information. Abundant storage, increased power, and affordable prices provide in-house users with the flexibility of on-line application development and/or links to networked structures that provide access to remote data centers.

The objective of this study was to build a micro-computer based information retrieval system that would serve as a tool for engineers and students at the University of Tennessee Space Institute for searching Environmental Engineering and Remote Sensing documents. Since Engineering Science encompasses several related engineering disciplines, the system is not restricted to the two areas mentioned previously, but is able to accommodate various Engineering Science databases. This tool will decrease the connect time to larger databases accessed by the university by providing preliminary searches of the on-line databases.

Chapter 1 begins with an introductory discussion of types of information re-

trieval systems and compares the similarities and differences. Chapter 2 goes into the theoretical concepts of modern information retrieval, such as automatic indexing, weighting, searching, and precision and recall. Chapter 3 examines the pertinent literature and discusses features of several systems currently on the market. The systems discussed are microcomputer-based as well as larger networked systems. Chapter 4 presents the features actually implemented in the development of the system and the basis behind the selection of specific features, Chapter 5 covers the software design of the individual modules and their functions, and Chapter 6 contains a brief summary and recommendations for improving the system.

1.1 Geographical Information Systems

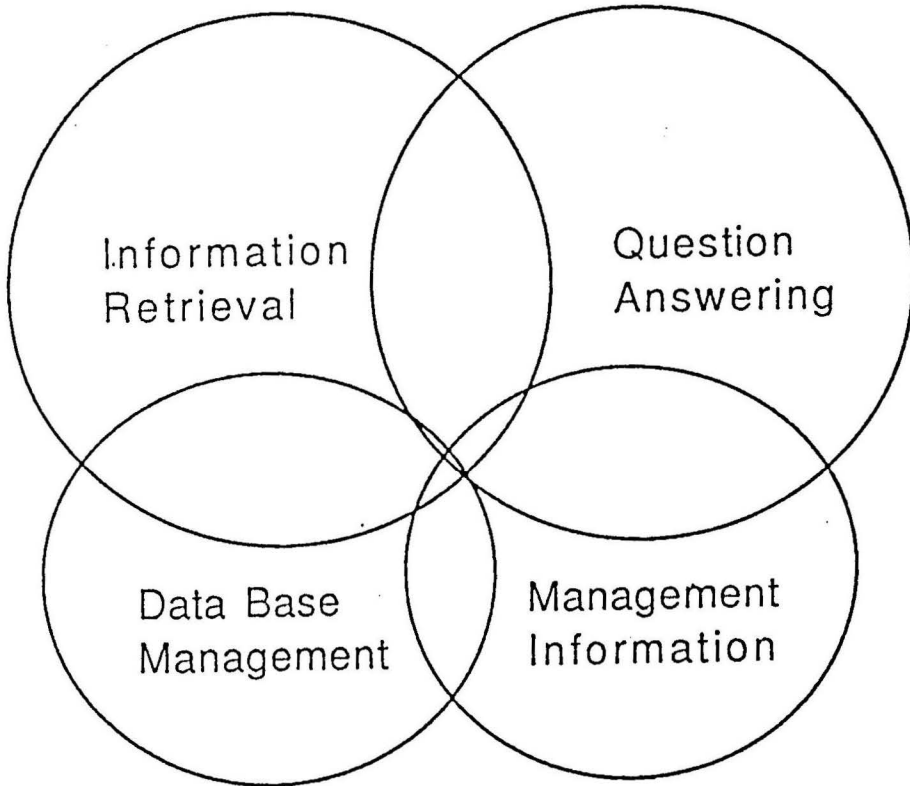
Several implementations have been employed to manage information and one usually finds that the methods tend to overlap as shown in Figure 1-1 [4].¹ Some of The most widely used methods include: Data Base Management Systems (DBMS), Management Information Systems (MIS), Geographical Information Systems (GIS), and Question and Answering Systems.

The type of information retrieval system currently in use in the Remote Sensing Division at The University of Tennessee Space Institute is a Geographical Information System. SCIPS, Small Computer Information Processing System, is the software package designed by the Remote Sensing Division to operate on an IBM-PC or AT compatible computer. Before elaboration on the details of SCIPS it will be necessary to define a Geographical Information System. A Geographical

¹ Numbers in brackets refer to similarly numbered references in the Bibliography

Retrieves documents and references
Stores natural language texts
Processes approximate queries

Retrieves specific facts
Stores facts about special discourse
areas and general knowledge
Processes unrestricted queries



Retrieves stored (numerical) data elements
Stores data elements in tabular form
Processes exact match queries

Adds to data base management
analyzing and synthesizing
procedures (summaries,
averages, projections)

Figure 1-1 Overlap among Types of Information Systems.

Information System is a computerized system that processes spatial data, usually in the form of statistical data, or maps. The GIS is an extension of a Database Management System because it has to have special ways of processing the spatial data before the data can be put into the database in record formats.

The hardware components of the GIS system currently in use are illustrated in Figure 1-2 [1]. The system consists of an IBM-PC computer, an Epson compatible printer, a harddisk unit with tape backup, a GTCO digitizing pad, a battery power supply system, and a Houston Instrument's pen plotter. The digitizing pad is used to convert maps or other images to digital data to be used in conjunction with various vendor and customized software.

The software programs used to input, store, and manipulate data are shown in Figure 1-3 [1]. Once the data is stored, DBASE III software is used to build and maintain databases. Customized software is used for conversion of coordinate systems, area calculations, printing map labels, and many other functions.

1.2 Management Information Systems

A management information system is a data base management system geared toward managerial needs. This system not only accesses structured records found in a data base, but also integrates several functions and various types of homogenous data needed to make accurate decisions. For example, aids such as scheduling, spread sheets, accounting, and projection/forecasting packages allow managers to use data to model real-time situations [4].

In order to provide management with timely, accurate, and pertinent information, it is necessary to identify the separate levels of management, and thus, potential system users. Typical levels of management include: (1) Senior Managers

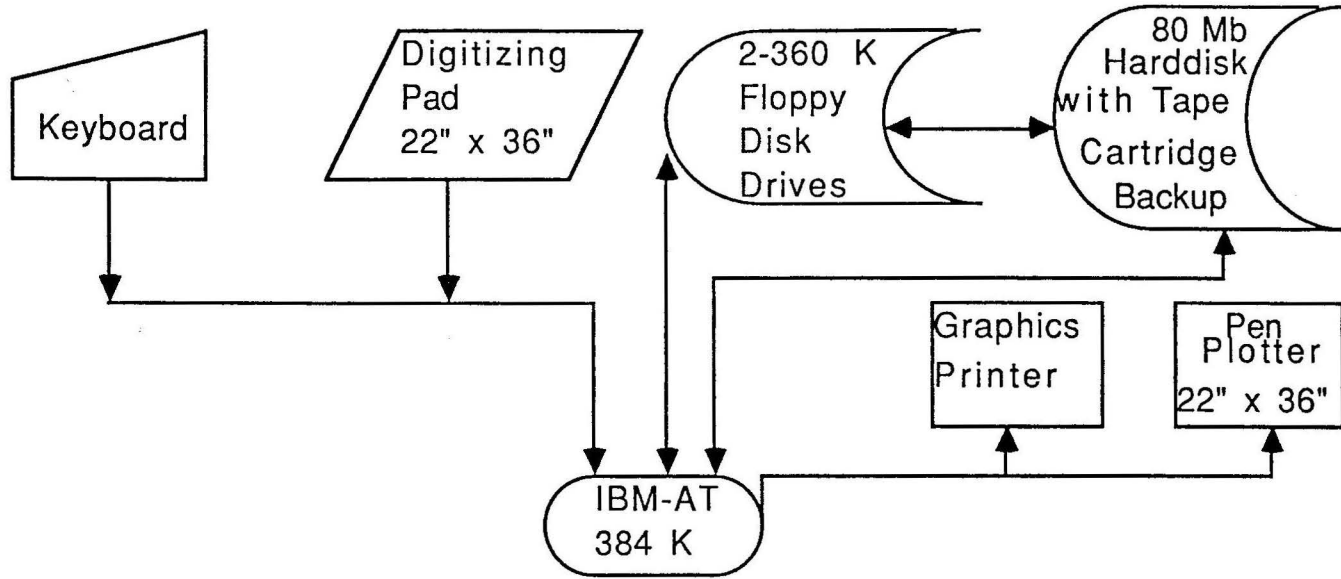


Figure 1-2 GIS Hardware Components.

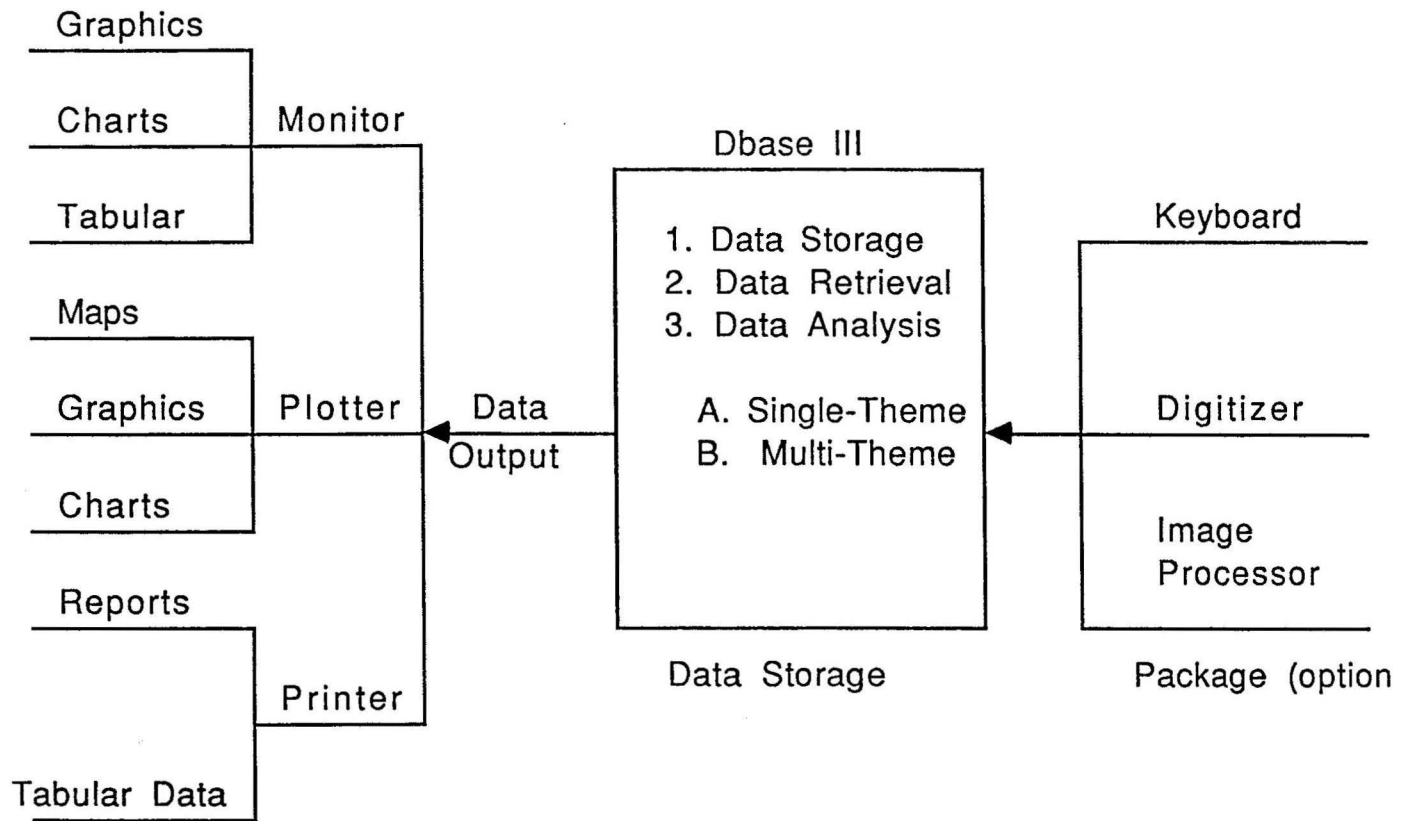


Figure 1-3 GIS Software Components.

and Planners, which include the organization's leaders and long-term planners, (2) Middle Management, which includes all leaders in charge of year to year planning, (3) Operational Management, whose leaders are responsible for month to month planning, and (4) Supervisors, who are concerned with day to day leadership and management [2].

The end result or overall purpose for using a management information system is to enable management to make better decisions in shorter reaction times. As was mentioned earlier, MIS systems integrate several functions and homogeneous data needed to make accurate decisions. These systems provide not only factual data, but also statistical extrapolations, decision evaluation functions, and even calculated risks or probabilities associated with alternative courses of action.

1.3 Decision Support Systems

A decision support system can best be defined as a computer-based support for management decision-makers dealing with semi-structured problems. The previous discussion of the levels of management is analagous to the different levels of the support provided by a decision support system. The different levels of support are: (1) support that accesses facts or information retrieval, (2) support that involves filters and pattern recognition ability to information retrieval, (3) support that adds computational procedures for 1 and 2, and also enables the manager to ask for simple computations, comparisons, and projectives, and (4) support that provides a useful model for simulation of real-time environments [3].

Normally the problem is formulated using a natural language that is processed by the system, then a model is built from information collected, and finally the problem is evaluated. The main distinction between an MIS system and a decision

support system is that decision support systems perform operations on heterogeneous types of information. This type of system integrates information retrieval systems, data base management systems, computer graphics, and other capabilities which collectively provide powerful tools in support of the decision making process.

1.4 Database Management Systems

Database Management Systems are concerned with the storage, maintenance and retrieval of data facts versus textual data. The information is usually in tabular form implemented in a record structure that contains several fields of information. In order to extract information meaningful to the user, he/she needs to have previous knowledge of specific record identifiers.

In a database, each record contains several fields which are identifying characteristics for an information item. A typical example of a database is a set of records containing mailing addresses, telephone numbers, sex, salary, social security, and marital status of employees for a company. In order to search for specific information a display of the record structure should be provided to the user for accurate query formulation. The output may consist of all records, tables, or other customized forms [4].

1.5 Question and Answering Systems

Question and Answering Systems provide access to factual information in a natural language setting. The stored information consists of a large volume of related facts and general world knowledge. The user query is formed with a natural language and the responses may also contain natural language, although the problems associated with the redundancy of the English language and query formulation

make this system lean toward a more experimental state.

These systems consist of three parts: knowledge base, database, and control programs. The objective of the system is to analyze the user's request, compare the query with stored knowledge, and assemble an acceptable response from relevant facts [4].

CHAPTER 2

SURVEY OF LITERATURE

This chapter examines the basic principles applied in the theory of modern information retrieval systems. The concepts discussed encompass the design, development, testing, and analysis phases which include such topics as: problem definition and decision making, indexing and weighting, dictionaries, searching, and precision and recall issues. Because rudimentary principles of information retrieval systems cannot be explained without posing certain obvious questions, the chapter begins with problem definition.

2.1 Problem Definition

Problem definition can not be over-emphasized. Without proper direction and planning, the Information Systems Analyst may not adequately satisfy the needs of the end-user. First, one has to assess the targeted group for which the system is intended. For instance, will the information be used in a medical environment, judicial, or educational institution? Quite naturally one needs to know whether the user is knowledgeable about information retrieval concepts, or even about computers in general. These considerations will determine the level of user-friendliness needed.

If the system is to contain a large versus small document collection, there are questions concerning the amount of memory needed for primary and secondary storage of files, pointers, system procedures, and other software programs. Also, one has to forecast the projected growth of the stored database and be able to design the system accordingly so that current resources and future needs will be met with as little friction as possible.

The last point touches on the decision making process that gives rise to the most cost-effective way to meet the needs of the end-user. A useful motto is "keep things simple." Why provide "great" features that the inexperienced user will rarely use or need? On the average one will find that only a small percentage of the total targeted population will be knowledgeable about information retrieval theory to necessitate the frequent use of all features in a package.

2.2 Indexing

After the decisions are made in the problem definition phase, the next step is to consider the elements composing the functional phase of the system. Every information retrieval system consists of information items (DOCS), a set of requests (REQS), an indexing language (LANG), and some method of determining which, if any, of the information items meet the requests (SIMILAR). This functional overview is presented in Figure 2-1 [4].

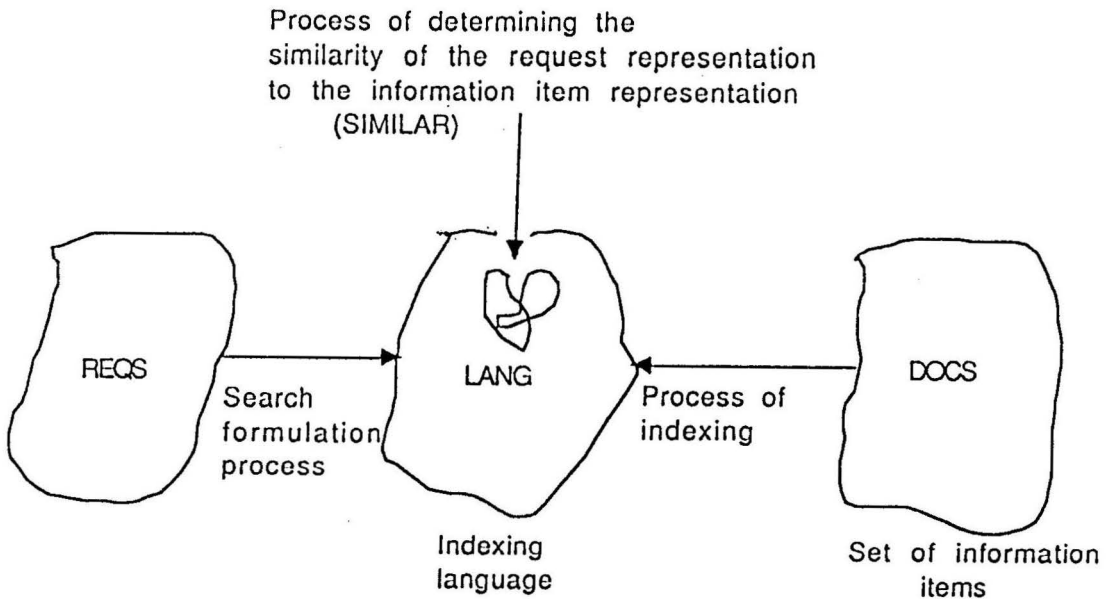


Figure 2-1. Functional overview of information retrieval.

Decisions that affect the mapping of the information items to the indexing language include: (1) manual indexing, (2) automatic indexing, or (3) a combination of the two processes. In addition, the indexing language can be prespecified (controlled) or uncontrolled. The overall purpose is to maximize the representation of information items using the indexing language. Several methods have been employed to maximize the assignment of index terms to information items [5].

Automatic indexing may involve deriving term significance factors based on frequency characteristics of each word in document texts. The first two steps involve calculating the frequency of each term k in document i ($freq_{ik}$), and determining the total collection frequency ($TOTFREQ_k$) for each word by summing the frequency of each term across all n documents given by the equation:

$$TOTFREQ_k = \sum_{i=1}^n (freq_{ik}) \quad (1)$$

After collecting the frequency characteristics one has to eliminate indiscriminate terms by setting limits or thresholds for high frequency and low frequency terms. The remaining medium frequency words will be used for assignment to documents as index terms. Figure 2-2 shows the curve representing these frequency characteristics [4].

Another approach to automatic indexing also uses term significance factors to compute the discrimination value of a term. This measures the degree to which a term distinguishes one document from another. In addition, it provides an adequate physical interpretation for the indexing process. In order to maximize the retrieval of relevant documents one must be able to measure the space density of a document collection. The expression for computing the discrimination value for each $term_k$ is:

$$DISCVALUE_k = AVGSIM_k - AVGSIM \quad (2)$$

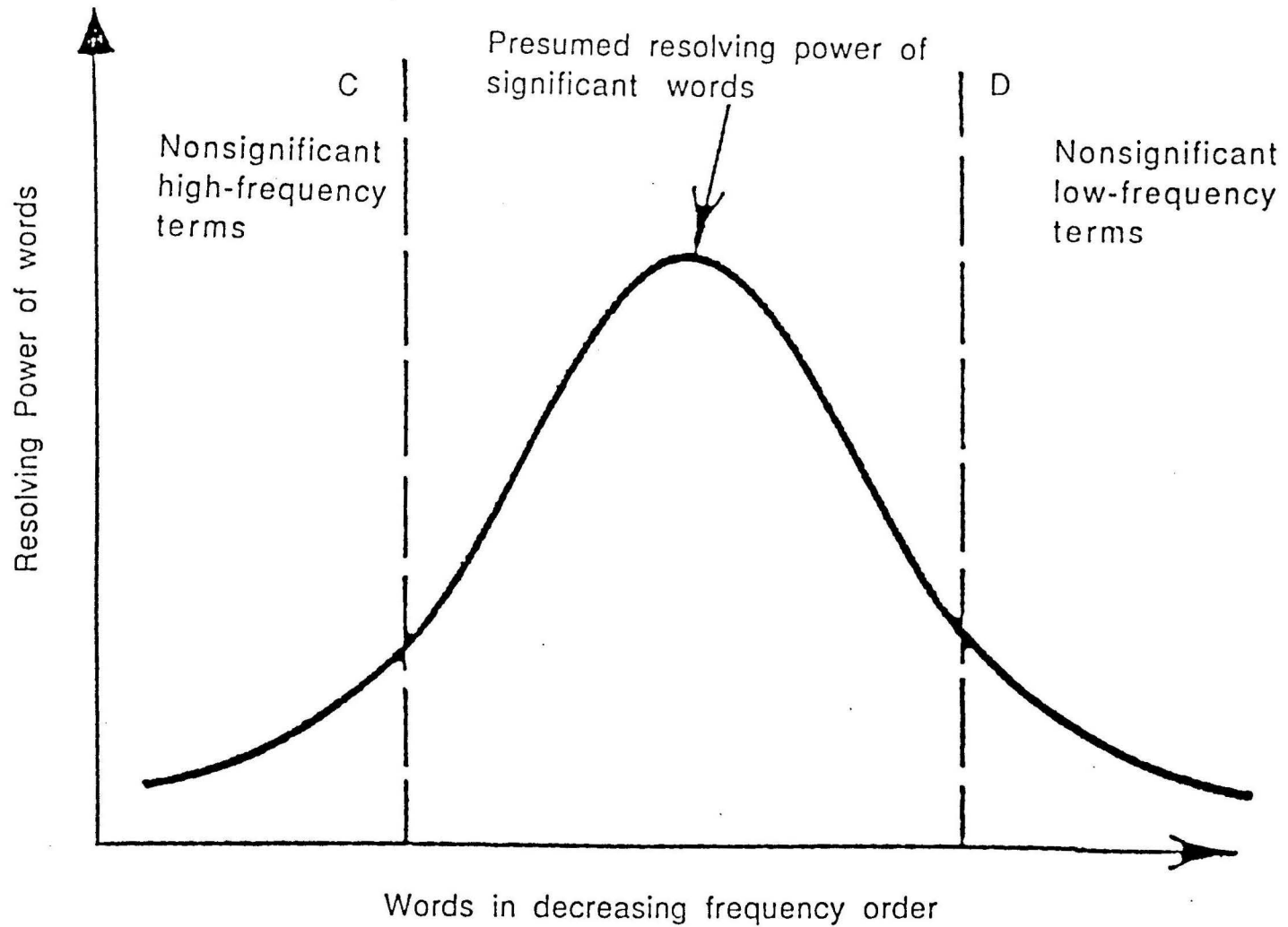


Figure 2-2 Resolving Power of Significant (Medium-Frequency) Words.

where $AVGSIM_k$ represents the space density after term k has been removed from the original document collection, and $AVGSIM$ represents the average document similarity for the entire collection [4].

The basic assumption is that a document space that is tightly grouped reflects somewhat similar index vectors, which is not useful for retrieval since one document can not be distinguished from another. The ideal situation is to expand the density of the document space so that relevant documents are separated from non-relevant documents. An example of a typical document environment, where the distance between two items is inversely related to the similarity of their index vectors, and an ideal document space collection is shown in Figure 2-3 [5].

After the discrimination values have been computed, the index terms can be placed into three categories: (1) good discriminators with a positive $DISCV ALUE_k$, (2) indifferent discriminators with a $DISCV ALUE_k$ close to zero, whose removal or addition leaves the similarity among documents unchanged, and (3) poor discriminators whose negative $DISCV ALUE_k$ renders the documents indistinguishable from each other. The term discrimination value can now be used in conjunction with the term frequency factors to assign an importance factor, or weight, for each term k in $document_i$. Equation 3 shows the relationship for weighting, using term discrimination values.

$$WEIGHT_k = freq_{ik} \cdot DISCV ALUE_k \quad (3)$$

Manual indexing usually has to be done by subject experts or someone experienced with Information Science. The use of dictionaries such as, Synonym and Thesaurus dictionaries, are widely use to normalize and control a language. This technique is one that can be easily implemented. More discussion on this topic will be given in Chapter 4.

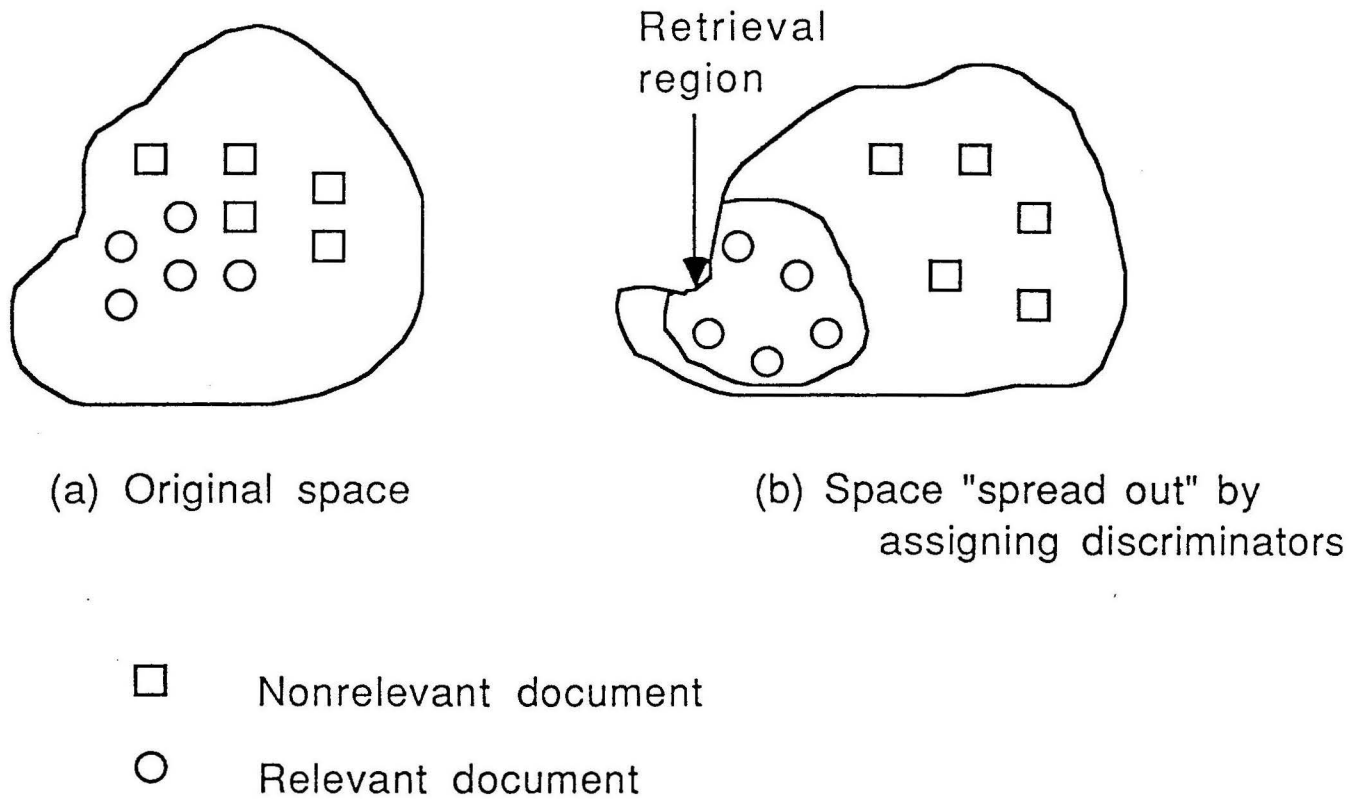


Figure 2-3 Document Space Density Examples.

2.3 Dictionaries

A dictionary is that which explains or defines: word usage, formal relationships, a basis for searching appropriate words, record of authoritative decisions on usage, and error control [6]. This section explains the role of dictionaries in the development of information retrieval systems. Specific examples include the Negative dictionary, Synonym dictionary and Thesaurus.

The first discussion deals with the one-for-one Synonym dictionary. This dictionary gets its name from the fact that for each entry there is one term definition whose meaning is construed to be synonymous to that of the entry. More than one entry can have the same definition, but some provision must be made for alternate spellings as well as different word forms. This relationship is shown in Table 2-1 [6].

Table 2-1 One-For-One Synonym Dictionary.

Entry	CODE
Book	1
Calculator	2
Catalog	3
Cataloger	4
Computer	5
Descriptor	6
Document	7
Index	8
Indexer	9
Processor	10
Query	11
Search	12
Term	13
Word	14

In addition to the Synonym dictionary, a more commonly used dictionary is the Thesaurus. Thesaurus groups relate words together according to subject class. Sometimes the term thesaurus is used to describe what is called a multiple-relation dictionary. To expand a vocabulary, sometimes index terms with questionable discrimination properties are used in conjunction with term association calculations. Whatever construction method is implemented, manual, semi-automatic, or fully-automatic, two problems must be addressed: (1) a decision must be made about what terms to include in the thesaurus, and (2) the terms specified for inclusion must be properly grouped.

One disadvantage of using a thesaurus is maintenance, and two kinds of maintenance problems arise. First, the thesaurus may require restructuring as a result of user interaction with the system. For example, new user populations may mean different levels of experience and interests which require new vocabulary terms. Secondly, if new documents are added to a collection, a thesaurus maintenance system will be needed to accommodate collection growth. Some possibilities to consider for updating the thesaurus are:

- (1) Leave the original thesaurus unchanged and use it for the expanded collection.
- (2) Place new terms into existing thesaurus categories only.
- (3) Place new terms into separate new classes.
- (4) Completely restructure the thesaurus by generating a term classification from updated vocabulary.

Table 2-2 shows an excerpt of a thesaurus used in an automatic indexing environment for documents in engineering [4].

Table 2-2 Typical Thesaurus Excerpt.

CLASS	TERMS	CLASS	TERMS
408	DISLOCATION JUNCTION MINORITY-CARRIER N-P-N P-N-P POINT-CONTACT RECOMBINE TRANSITION UNIUNCTION	413	CAPACITANCE IMPEDANCE-MATCHING IMPEDANCE INDUCTANCE MUTUAL-IMPEDANCE MUTUAL-INDUCTANCE MUTUAL NEGATIVE-RESISTANCE POSITIVE-GAP
409	BLAST-COOLED HEAT-FLOW HEAT-TRANSFER		RESIST SELF-IMPEDANCE SELF-INDUCTANCE
410	ANEAL STRAIN	414	ANTENNA KLYSTRON PULSES-PER-BEAM RECEIVER SIGNAL-TO-RECEIVER
411	COERCIVE DEMAGNETIZE FLUX-LEAKAGE		

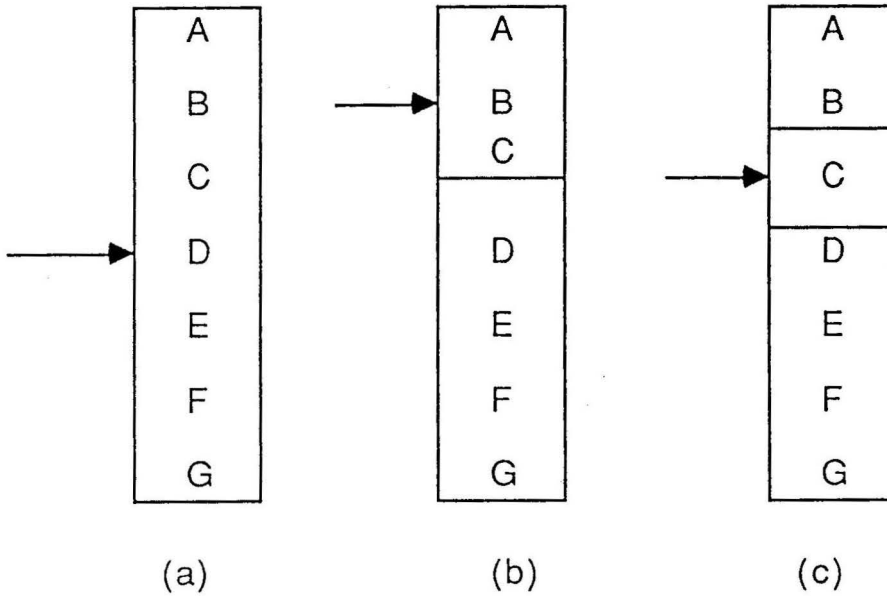
Last, but not least, is another commonly used dictionary called the Negative or Null dictionary. The Null dictionary contains high frequency and commonly used words like "a", "and", "the", and "this". The purpose of this dictionary is to minimize the search time used in searching for insignificant terms. Once a term is found to be in the Null dictionary, it is simply eliminated from the subsequent searches of the Synonym and/or Thesaurus dictionaries [7].

2.4 Search Techniques and File Organization

The type of search technique used in an information retrieval system depends on the size of the document collection, the data structures implemented for file organization, computer costs, and the efficiency of the search technique itself.

The simplest search is referred to as a Linear Search, or a Sequential Search. Since the collection is unordered, the file must be examined one element at a time. If the document collection is very large, it would be very inefficient to use this method, especially if the item were near the end of the collection. The main advantage is that insertions and deletions can be done easily without altering the order of already existing items.

The efficiency of a search is increased when the file is ordered, not strictly sequential, but by the key values used in the search. A Binary Search on the average requires $\log_2(n + 1)$ steps, where n is the number of total documents, compared to $(n + 1)/2$ steps required for the Linear Search. The process of the Binary Search is initiated by starting in the middle of the collection, and depending on whether the value of the search key is greater than or less than the value presently pointed to, one of the sections is discarded. The process continues halving until a match is found as shown in Figure 2-4 [4].



- (a) Initial comparison with middle element (D).
- (b) Next comparison with middle element of upper half (B).
- (c) Final comparison with remaining element (C); ignore upper half.

Figure 2-4 Binary Search Example for Key C.

Indexed files also require files to be ordered. The search for a particular item requires only a search of the index and the items specified by the index. On the average, it requires the number of steps needed to search the index, plus $(n + 1)/2$ additional steps for the specific sequential records of the file. For example, given a sequential file of one million records, a search to find a particular item beginning with a certain letter will require $(n + 1)/2$ or 500,000 steps on the average. If the indexed file method is used, then a separate one-dimensional array representing each letter of the alphabet, or 26 entries, is created. The search of the indexed array will require $(26 + 1)/2$ steps. If the number of records per specified letter equals 50,000, then this additional search requires requires $(50,000 + 1)/2$ or a total of 25,014 steps [4].

The big disadvantage with using indexed files is that the addition of new records requires updates in the index and the subfile. This leads into the topic of file organization. One of the most important decisions one can make concerns file organization, specifically, whether to use direct files or inverted files.

A direct file is a file in which the documents provide the main order of a file, whereas an inverted file is arranged by keyword, and each keyword includes a corresponding list of document numbers. Simply stated, the direct file identifies all terms associated with a document, and an inverted file identifies all documents associated with a term. An example of these relationships is shown in Figure 2-5 [4].

2.5 Precision and Recall

The discussion of theoretical concepts concludes with a brief examination of retrieval evaluation; namely, precision and recall. Certain decisions are made

Information Items

Topics	Document 1	Document 2	Document 3
Term 1	1	0	1
Term 2	1	1	0
Term 3	0	1	1
Term 4	1	1	1

(a) Inverted File

Information Items	Topics			
	Term 1	Term 2	Term 3	Term 4
Document 1	1	1	0	1
Document 2	0	1	0	1
Document 3	0	1	1	1
Document 4	1	0	1	1

(b) Direct File

Figure 2-5 Inverted and Direct File Examples.

regarding the degree of precision and recall desired by a user.

Recall is defined as the proportion of relevant material retrieved over the total relevant material in the collection.

$$R = \frac{\text{number of items retrieved and relevant}}{\text{total relevant in collection}} \quad (4)$$

Sometimes a user may want as much relevant material as possible, as in a literary search, so the user must submit broad queries to retrieve several documents. Usually recall increases as the number of documents increases; however, the precision is likely to decrease. This relationship is shown in Figure 2-6 [4].

On the other hand, some users may be more interested in high-precision; therefore, high-precision users will have to submit narrower and more specific queries. This involves rejecting those items that are less relevant.

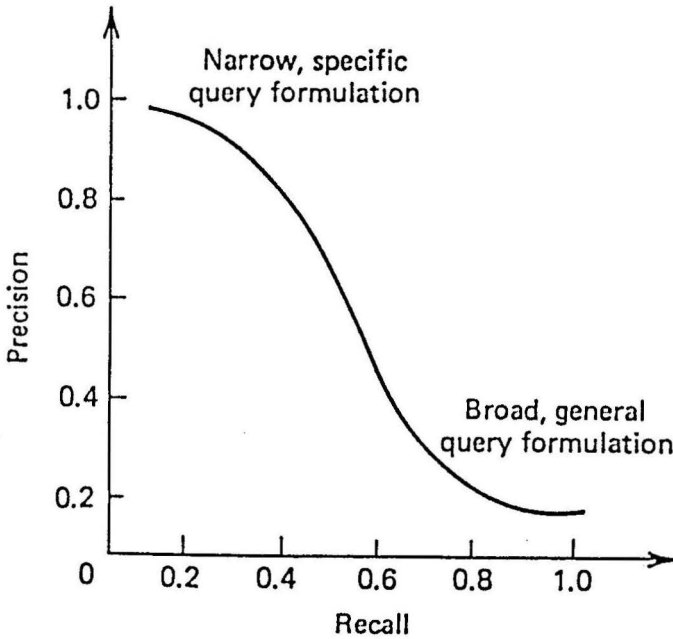


Figure 2-6 Typical Average Recall-Precision Graph.

Precision is then defined as the proportion of relevant material retrieved over the total retrieved.

$$P = \frac{\text{number of items retrieved and relevant}}{\text{total retrieved}} \quad (5)$$

Table 2-3 shows an example of the rankings of documents in decreasing query-document similarity order, and the calculations of the recall and precision values. Figure 2-7 is the corresponding graph of the recall and precision values for the sample query [4].

Recall-precision graphs have been criticized because several parameters have been obscured. The most noticeable problem in the graph of Figure 2-7 is the fact that the size of the retrieved document set and the collection size are not shown. Another problem is that one has to interpolate points in order to produce a continuous graph from the discrete set of points. Looking at Table 2-3, it is shown that the precision value is known exactly for a recall of 0.2, but not for 0.4, since the precision lies between 1.0 and 0.67 at 0.4. Also, the recall value is known exactly for a precision value of 0.5, but not when it is 1.0. The most important thing to remember is that the information obtained in Table 2-3 and Figure 2-7 are for specific queries and in order to obtain information about the average performance characteristics for several user queries, a number of curves must be processed [4].

Table 2-3 Output Ranking of Documents in Decreasing Query-Document Similarity Order and Computation of Recall and Precision Values for a Single Query.

Recall-Precision after retrieval of n documents				
n	Document Number (x = relevant)		Recall	Precision
1	588	x	.2	1.0
2	589	x	.4	1.0
3	576		.4	.67
4	590	x	.6	.75
5	986		.6	.60
6	592	x	.8	.67
7	984		.8	.57
8	988		.8	.50
9	578		.8	.44
10	985		.8	.40
11	103		.8	.36
12	591		.8	.33
13	772	x	1.0	.38
14	990		1.0	.36

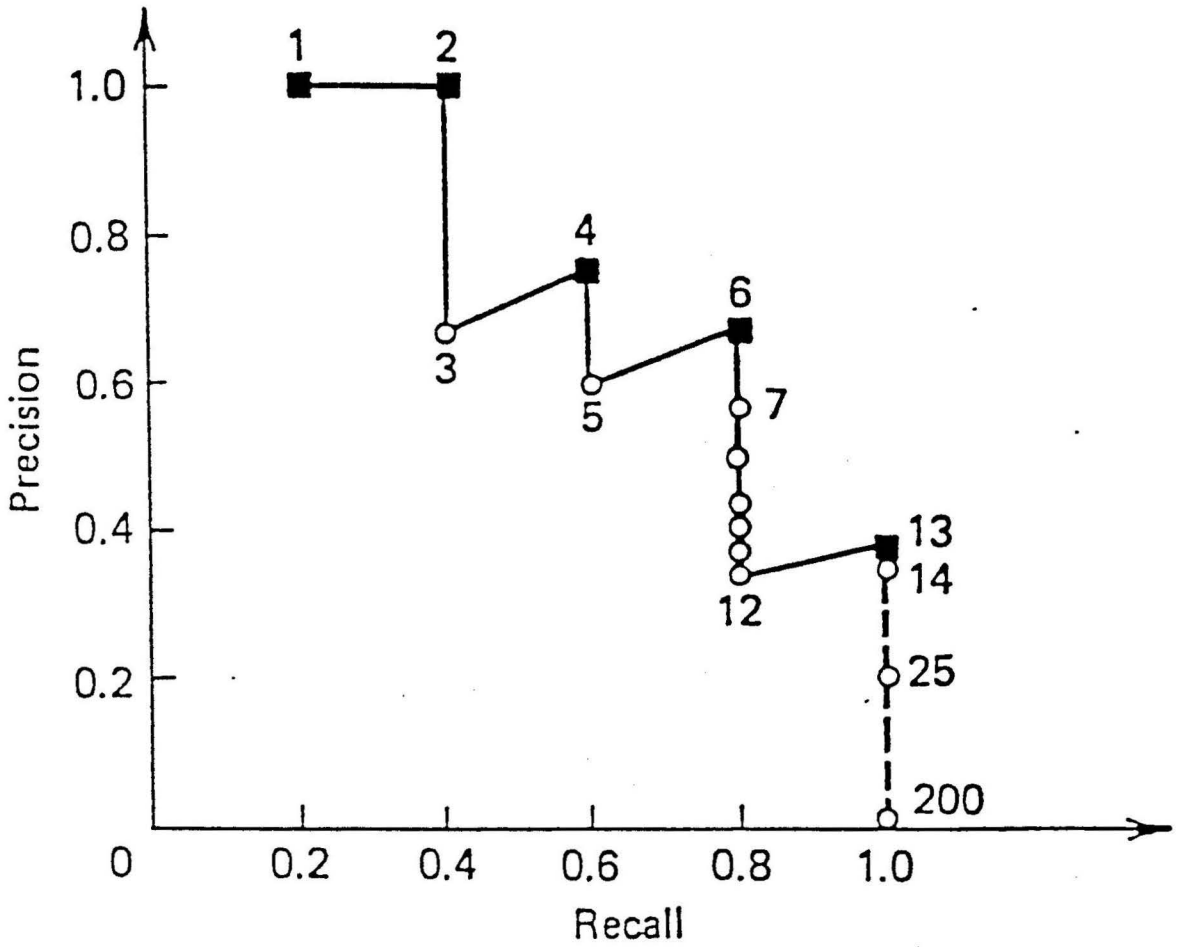


Figure 2-7 Graph of Precision Versus Recall for Sample Query of Table 2-3.

CHAPTER 3

REVIEW OF EXISTING SYSTEMS

The information age of the eighties has created a growing concern over the quality of software developed for database management systems and information retrieval systems. There are several systems commercially available on the market today, but not all systems contain the same features. Some are microcomputer-based, and others may be used for large multi-user minicomputer or mainframe systems. Nevertheless, there are several specific requirements and general considerations for selecting information retrieval software.

The decision to design and develop a system in-house rather than purchase an existing one was based on the fact that no one system delivers everything, and not all systems have the same features. RETRIEVE, although basic in structure and features, is able to provide both exhaustive and precise searches. The simplicity of the system is advantageous to the end user in that query formulation, and output specifications are user friendly, the manual indexing cuts down on total computer processing and provides control of the thesaurus for the indexer, the inverted file structure provides quick retrieval of the relevant documents, and the absence of the AND Boolean logic ensures retrieval of some documents as opposed to not retrieving any documents when searching for documents containing all terms specified.

This section first addresses a few selection criteria for software packages, and then examines several microcomputer-based and larger networked systems currently available on the market.

3.1 Selection Criteria

One of the most pressing considerations that needs to be addressed is storage requirement. This requirement is not always easy to estimate because the system developer not only has to take into account the current number and size of records contained in the database, but also has to consider the projected future growth of the system.

Disk storage will vary depending on how files are stored and processed. Since most packages use inverted indexes, which can take up large amounts of memory, storage must be adequate to handle the various pointers. In addition, added space for work files, associated files for processing the main file, and the software itself will require space on the disk system.

File organization and data structures implemented will also contribute to additional storage requirements. If fixed length records are employed, inevitably there will be some wasted storage because most records will be shorter than the longest record accommodated. Array utilization versus dynamically allocated structures, such as linked lists or stacks, will also cause some wasted storage.

Another selection criteria also of utmost importance is hardware requirements. Normally, floppy disk-based systems will not provide ample storage for large information retrieval systems, but for some smaller systems floppy disks are sufficient. Usually a harddisk is used for database software and storage of actual data to be searched because harddisk storage has greater storage capacity than floppy disks, is more flexible than using several floppy disks, and is inherently better in performance because of the rapid disk access and data transfer rates.

Search features vary considerably among the different retrieval packages. Some

of the differences have a lot to do with the end-user's abilities, but usually the complexity of the query language, the degree of help available on-line, and ease of query formulation play a large part. Someone more experienced with information retrieval concepts will obviously have a better grasp for searching strategies than the ever present ubiquitous end-user. Most systems include tutorials to help aid the searching process. Others may employ menu-driven interfaces. The menus help to present a choice of selections at each step, which eliminates the need to memorize a command language, but the disadvantage is that menus add significantly to connect time and thus increase search costs.

Boolean logic is commonplace in most information retrieval packages. Typical usage supports AND, OR, and NOT functions. The query formulation may require single entry of each term and the resulting sets combined, or multiple terms may be entered in a single query. Some packages may use parenthetical grouping and others may not. Some systems may not support the NOT logic at all. Ultimately the emphasis is not on the Boolean logic itself, but the degree of support and the ease and flexibility of the package overall.

One feature that is used in conjunction with Boolean searching is comparison searching. Comparison operators enable a user to specify values that are equal to, less than, greater than, less than or equal to, greater than or equal to, or not equal to another search argument. These types of operators are useful for numeric field searching.

Another major consideration is system output. The amount of flexibility and control provided by the software in formatting output to various devices such as the display screen, printer, or output file may be very important for several applications. Still the capabilities vary greatly from system to system . Some systems are not

very flexible when it comes to output and may provide only one fixed format, while others may provide the capability to design various formats that can be selected by the user.

Since a database contains information that is valuable to many users, it represents a considerable investment of time and money. Thus, the system should be protected from malicious or inadvertent tampering by incorporating a security system. Unfortunately, this is area is where most software packages are deficient. For larger systems, it may be necessary to develop a system of passwords and levels of access in order to ensure protection against any misuse of information.

One must realize that setting up, maintaining, and using a database system is complex; therefore, it is essential to provide adequate documentation for the system. In addition to a user manual and system manual, many systems provide on-line help screens, on-line tutorials, and telephone assistance to assist in the installation and use of the software. Even if the user is expected to be experienced with information retrieval systems, it is still good practice to have extensive documentation to ensure ease of use of the system [8].

Now that a few selection criteria have been discussed, the next few sections will examine software packages currently available for commercial use, namely, MICRO-CDS/ISIS, PRO-SEARCH, and DIALOG.

3.2 MICRO-CDS/ISIS

MICRO-CDS/ISIS is a bibliographic information management software package developed by UNESCO. It was originally intended for mainframe applications, but the current version released is geared toward the mini-micro market. First, general features of the software are discussed, and later the major features the package

has to offer are examined.

MICRO-CDS/ISIS is available free of charge to non-profit organizations of UNESCO member states. Hardware requirements include an IBM PC/XT with 512 Kb of RAM, a floppy drive, and a 10 Mb harddisk. The harddisk is capable of storing 10,000 to 12,000 records with an average record size of 500 bytes.

Presently, MICRO-ISIS is being used at the Library of the Computer Applications and Services Company in Budapest, Hungary, where it demonstrates the feasibility of microcomputer-based information retrieval systems for relatively small libraries.

The structure of the software is composed of a set of six modules written in PASCAL language. These six modules illustrated in Figure 3-1 are divided into system functions and user functions [9]. The system programs are used to: (1) define the structure of the database, the master and index files, and the format and content of the worksheets and menus (ISISDEF), (2) to reorganize, back-up, and restore files (ISISUTL), and (3) to create message files, stopword list, etc (ISISXCH). The end-users (librarians and patrons) use the user programs to enter and modify data (ISIS), update the index file, search the database (ISISINV), and sort, display, and print results (ISISPRT).

MICRO-ISIS is a menu-driven, text-oriented software package. It is designed to accomodate fields and records of varying length, with a maximum record length of 4096 characters. MICRO-ISIS allows the use of subfield specifications which can be used with different format statements for printout and display. The subfield delimiter used is the roof sign followed by a one-character alphabetic or numeric subfield identifier.

Searchable elements are stored in an inverted file, which contains a reference

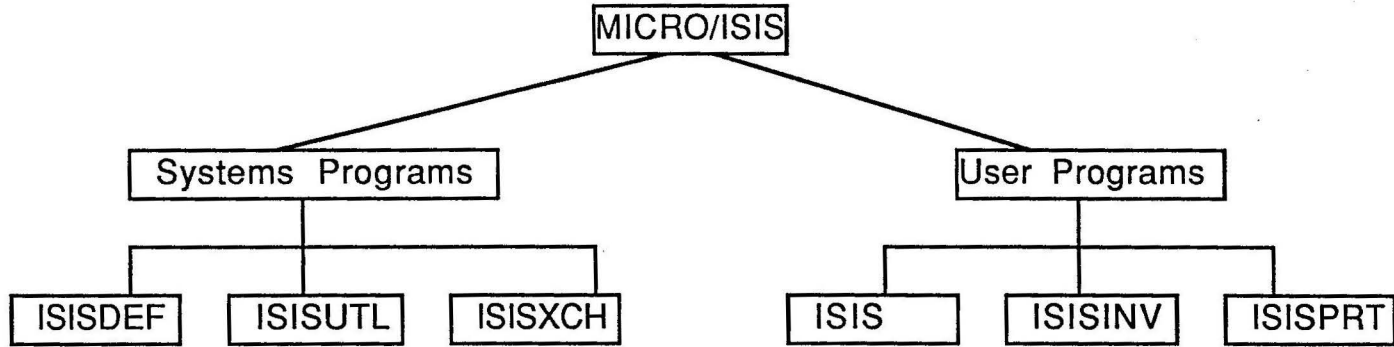


Figure 3-1. Software Structure of MICRO/ISIS.

to the master file records from which the searchable element was extracted. Table 3-1 gives details of the inverted file listing or search term dictionary where the various access points such as author names, subject descriptors, and title words are formed in alphabetical order [9]. Another important feature included in MICRO-ISIS is on-line data entry worksheets of the database. Data entry fields are arranged in the following groups: page 1, main entry and added entry heading data, page 2, data of bibliographic description, and page 3, descriptors, codes, and local data.

MICRO-ISIS ranks high in user-friendliness and reliability, but is lacking in data validation facilities. Its entry-time error detection features are modest in version 1.0, restricted to checking field type, field length, and character pattern. However, the most attractive feature of MICRO-ISIS is its ability for searching the database and printing results of the search. Query formulation uses Boolean logic that is used in conjunction with proximity operators, field qualification, and right truncation of search elements. Table 3-2 gives an example of possible search operations, where A and B are search terms [9]. In addition, MICRO-ISIS uses parenthetical grouping and sets, where previously formulated query questions can be referenced as single search elements by their sequence number.

Other features include multilingualism of menus, system worksheets, and messages. Creation of new language versions and changing the default language are well supported by the system. Data exchange facilities provide the option of moving data bases from one information management system into another. Databases operated mainly under MICRO-ISIS can be transferred to the CDS/ISIS mainframe environment for comprehensive validation and/or to print catalogs via photocomposition programs of CDS/ISIS. Sections of large mainframe databases, selected by range specification of by a search, can be downloaded to create a MICRO-ISIS

Table 3-1 Inverted File Listing.

SEQUENCE NUMBER	FREQUENCY	INDEX TERMS
989	1	CATER, JOHN P.
990	1	CAVINESS, BOB F. (ED.)
991	1	CHEN, CHING-CHIH (ED.)
992	1	CHICAGO (US)
993	1	CIBBARELLI, PAMELA (ED.)
994	1	CIKKGYUJTEMENY
995	1	CLIFFORD, MARTIN
996	1	CMOS TECHNOLOGIA
997	1	COBOL NYELV
998	1	COMMUNICATION
999	1	COMPUTACION
1000	3	COMPUTERIZED
1001	2	CONFERENCES
1002	4	CONSULTANCY
1003	4	CONSULTATIVE
1004	1	CONTROL CIRCUIT
1005	1	COOK, CURTIS R.
1006	3	CORPORATION
1007	1	COSTA, BETTY
1008	1	COSTA, MARIE

Table 3-2 Search Operations Example.

OPERATION	DESCRIPTION
A + B	logical OR connection
A * B	logical AND connection
A ^ B	logical AND NOT operation
A (G) B	occurrence in fields with identical tags
A (F) B	occurrence in the same field
A . . . B	occurrence in the same field, with a distance less than the number of dots

subset data base for more convenient and less expensive access [9]. The documentation includes both a reference manual and a tutorial. An Installation Guide and an Introduction to the System are also part of the documentation. In summary, one can see the attractiveness of this microcomputer-based library system is due to the many features discussed in this section, and to top it all off it is free of charge for qualified users.

3.3 PRO-SEARCH

PRO-SEARCH is a microcomputer-based search-aid software package designed to help on-line database searching. It was the second product developed by MENLO Corporation, IN-SEARCH being the first, and it concentrates on improving the search process itself rather than emphasizing telecommunications or search output packaging. PRO-SEARCH merges DIALOG and BRS files (on-line databases) and search protocols, suggests alternative data bases, provides other services to be searched in "native-mode", enhances output, and maintains detailed accounting

records for BRS and DIALOG sessions. This search-aid software is geared toward the experienced end-user who knows how to search, but may simply want to access a different system without having to learn or remember new protocols. In the "native-mode", PRO-SEARCH allows searchers to follow regular searching procedures, but they get to choose between either DIALOG or BRS protocols in accessing either service.

The basic structure of PRO-SEARCH includes two interface modes, high-level and native-mode. The high-level interface mode allows searching of DIALOG2 and BRS interchangeably. Worksheets are used for search strategy and up to 98 sets can be created. Database selection allows on-line as well as off-line searching using the full-screen editor.

Four main categories of information are used, which are equivalent to four floppy disks, Art, Education and Social Sciences; Biology and Medicine; Business, Government and News; and Engineering, Mathematics, and Physical Sciences. If the one megabyte of category disk is used on the harddisk this will allow more flexibility because if not, every time a category is switched, floppy disks must also be switched.

The native-mode allows creation of a custom on-line services directory with pre-configurations for telecommunicating up to 23 different services. However, the accounting reports, data sheet displays, and offline strategy only work for BRS, DIALOG, and DIALOG2.

After logging on, PRO-SEARCH provides for uploading of offline searches up to 50 lines and 240 characters. The buffer facility allows storage of search output on disk for later use. The search output on the screen can be examined by depressing the mark key, which sends a specified record to the buffer.

The Accounting Reports Facility calculates automatic cost information from BRS and DIALOG searches. There are two reports available, session and monthly. The session report is a session invoice that lists the subject, client, name, charge code, and a summary of the files checked with all costs attached including a system total. The monthly summary reports summarize on-line expenditures for all or part of a month, and can be specified by client, charge code, searcher, data base, or on-line service. Although the system establishes adequate telecommunication connections with data banks, it does lack some basic properties of most telecommunicating programs.

For example, it cannot upload a large file, because of a limited number of lines with a pre-set character limitation. Many bulletin boards and electronic mail systems will take full files of any length. With PRO-SEARCH, one would have to download the search using the PRO-SEARCH buffer and disk facilities, logoff the system, exit PRO-SEARCH, and re-connect and transmit using another telecommunications program.

PRO-SEARCH does improve the searching process itself with its text-editing of searches, phrase adjacency, cross-data base, cross system fluidity, on-line instructions, and outstanding data base selection tools. Still no one package offers everything, and PRO-SEARCH is no exception. It does not tag DIALOG records with field information needed to re-process easily in a data base management program, it does not allow for editing within PRO-SEARCH, and it does not utilize spreadsheets for calculating from numerical file downloads [10].

3.4 DIALOG

Lockheed Information Systems of Palo Alto, California developed the DIALOG

system. The DIALOG system is based on inverted files. By using a SELECT command the system creates sets of document reference numbers. For example, if a user is searching for relevant documents identified by "information retrieval", he would have to specify SELECT "information" (which would create one set of document reference numbers) first, and then SELECT "retrieval" (which creates another set of document reference numbers).

Boolean logic is also implemented in the system by employing the COMBINE statement. The COMBINE statement allows use of the Boolean operators AND, OR, and NOT. Using the above example, to form the query statement "information and retrieval", the user would have to specify COMBINE 1 and 2. The usual order of operation performs the NOT first, followed by the AND operator, and then the OR operation, but if parenthetical grouping is included then the sequence would be altered.

DIALOG also supports right truncation. For example, PSYCH? can be searched for PSYCHIATRIST, PSYCHIATRY, etc. Embedding of the ? can also be used to substitute for characters. Other highlighted features include several functions for word position information. One can search for pairs of adjacent words or terms located within a specified number of words of another term. The system also utilizes field identification for author (AU), classification code (CC), corporate source (CS), document type (DT), journal name (JN), language (LA), publication year (PY), and update (UP) [4].

Logging in to the system is not difficult, although it requires remembering phone numbers, telecommunications network protocols, service "addresses", and passwords. The pre-configured parameters for the one-step logon procedure include Knowledge Index, DIALMAIL and the communications networks DIALNET, Te-

lenet, TYMNET, and UNINET. For security measures, passwords are masked so they don't show up on the screen or printouts.

Other major features include offline creation of input (uploading), scrolling the screen, printing and/or saving to disk with edit capability, and accounting. The uploading feature allows the user to use the type-ahead buffer so strategies or messages can be created prior to connecting, and then the information can be sent automatically after connecting to save typing time. All information displayed on the DIALOGLINK screen passes into the retrieve buffer so that at any point, while connected or off-line, flexible movement of the cursor keys is available to move around the buffer. This feature is important if one doesn't print simultaneously. With DIALOGLINK, the last line sent to the type-ahead buffer, edited with the cursor, and sent again.

The information in the retrieve buffer can be directed to a printer or saved on disk during the on-line session or after logging off. Lines to be saved can be marked by using the on/off mark key, and editing capabilities are allowed on the saved files, which are straight ASCII files with the extension .REF. The last major feature to discuss, accounting, includes the capabilities to produce a title page and session invoice for a search session by specifying the accounting option from the session menu. To accumulate accounting information on a periodic basis for preparation of reports by client name, charge code, searcher, or data base, the Account Manager and the Communication Manager packages will be needed.

One major weakness of the system is that there is not a facility to include 2400 and 1200 baud rates in the same dialing directory. Even though DIALOGLINK has the ability to support other online services, it is not nearly as convenient. Another weakness is each time access to another service using another telecommunications

network is desired, it is necessary to enter repetitive entries in the Configure program. Also, the type-ahead and accounting features are not available for other services.

DIALOGLINK is geared for people who search primarily DIALOG services. The system operates on IBM-PC, PC/XT, PC/AT, Compac, or other IBM-compatible computers. At least 256K of RAM, DOS 2.0, or higher, and two double-sided disk drives or a hard disk system with at least one floppy are required. Hayes Smartmodems and ten other modems are also supported by the system. The Communications and Account Manager may be purchased separately, or an evaluation disk, which will operate for two hours on-line, including the Communications and Account Manager, is also available [11].

IMPLEMENTATION OF DESIGN FEATURES

Chapter 4 examines the decisions affecting the design implementation of the RETRIEVE system. Section 4.1 discusses the system objectives and methods used to implement various features. Section 4.2 discusses the overall structure and logic of the program, which will be elaborated upon with the flow charts and program listing in Appendix A and B respectively.

4.1 System Objectives and Methods

The purpose or objective of any information retrieval system is centered around satisfying the needs of the end user. As was stated in Chapter 2, the design of the system has to be in accordance with the needs of the targeted group(s) of people that will be using the system.

The Remote Sensing Division, Environmental Engineering department, and students of Remote Sensing/Environmental Engineering formed the targeted groups. The main objectives were (1) to combine specific query formulations to produce high precision results with exhaustive searching (high recall) for more generalized information, and (2) to provide interactive searching in a semi-automatic environment.

The database collection consisted of textual data in the form of abstracts to limit the amount of storage space required for full text documents. The abstracts were compiled from various Remote Sensing and Environmental Engineering studies. The representation of the database collection was implemented with a record structure to define the various fields of information for a particular entity. The fields of information chosen to represent each abstract included: a document number, a

counter for the number of lines in the title, a counter for number of lines for author information, author(s), a counter for lines of text in the abstract (lines were limited to 14), and the actual abstract (approximately 150 words). These counters were used to read the database information into the record structure.

The system was intended for use on a microcomputer since the document collection was intended to be relatively small. The harddisk for the GIS system provided adequate storage for external files, and the actual program. The software requirement was flexible because the IBM-PC provides several high-level language compilers. After writing the code for the program in PASCAL, several problems with debugging were encountered which made the process much too cumbersome. The program was rewritten on the VAX-11 because the system provided detailed documentation that made debugging quicker and easier.

PASCAL language was chosen for several reasons: (1) it is a higher-level language, (2) the author was very familiar with the language and features, (3) PASCAL language is known for its flexible user-defined data structures and string handling capabilities, and (4) PASCAL allows up to approximately 20 characters for variable names, which aids in choosing very descriptive names for debugging and documentation.

Arrays were used for direct access to the abstracts, and because the collection was small. If the database expansion was considered, then linked lists would be considered. The major drawback to using linked lists is the amount of storage needed for the files and pointers, and the major drawback to using array implementation is the fact that one must be aware of maximum bound specifications.

Other major decisions dealt with file structure and access methods. The main concern was to provide direct access for quick retrieval of the abstracts. The desired

result was achieved by using arrays for direct access. The file organization was sequential by system default. An inverted file structure was chosen for searching query terms. The choice of an inverted file structure was based on the fact that the physical structure of the inverted file identifies documents that contain terms. The array storing the abstracts is a two dimensional binary array where the rows represent the terms in the collection, and the columns represent the document numbers.

The binary array is marked with a one if a document contains a term, otherwise a zero is placed in the position. This structure provides for high-recall since all documents will be tagged when at least one query term is matched. Precision is hard to measure because it is contingent on the query formulation and the user's satisfaction: therefore, each query formulation will be different and each user will more than likely have different results to their specific query.

Another objective called for ranking the most relevant documents to a query (weighting). Each time a term was found to be in a document, a frequency counter was incremented by one for the particular term. After all terms were processed, the bubble sort algorithm sorted the frequencies in decreasing order, which produced the desired results.

Error checking was used sparingly in the program, so data entry was cumbersome. If an error was made in entering the database records, then one would have to complete entering the records, exit out of the program, and edit the database file, and if necessary, various counters. This method was also used when entering indexing terms per document. When an end-user enters query formulation terms from the thesaurus, the program checks for words that are misspelled by searching for a match between the terms entered and corresponding thesaurus terms which

are kept in an external file. If there is no match, then an error is flagged.

Thesaurus building was another design feature that was implemented . A thesaurus controls the indexing vocabulary. Ordinarily, this is an automated function that calculates term associations in order to group related terms in a class. Since documents were gathered from several remote sensing symposium articles, the papers were already grouped in the proper classes of information, which enabled bypassing this step. The classes of information used in the database collection were: UTSI Remote Sensing Theses, Land Cover/Land Use, Earth Science , Hydrospheric Science, Radar/Sonar. Image Processing, Work stations, and Photogrammetry. The complete thesaurus listing for the RETRIEVE document collection is found in Appendix C.

4.2 RETRIEVE Program Structure

The Retrieve program is an interactive PASCAL program that consists of a main program and five submodules. Figure 4-1 shows the structure of the program. The main menu allows the end-user to interactively select a function, but for clarity, the modules are described as one would logically use the software. The first step is to run the "Initialization" routine. This module should only be run once because it zeros out, or reinitializes counters and files. If the user runs into problems in another segment of the software he/she can choose which files are to be initialized again.

The next logical step is to enter records into the database collection that will be used for searching. The "Update" module prompts the user for the number of documents to process, and then the user enters the fields of information that corresponds to each record. After the number of documents specified have been

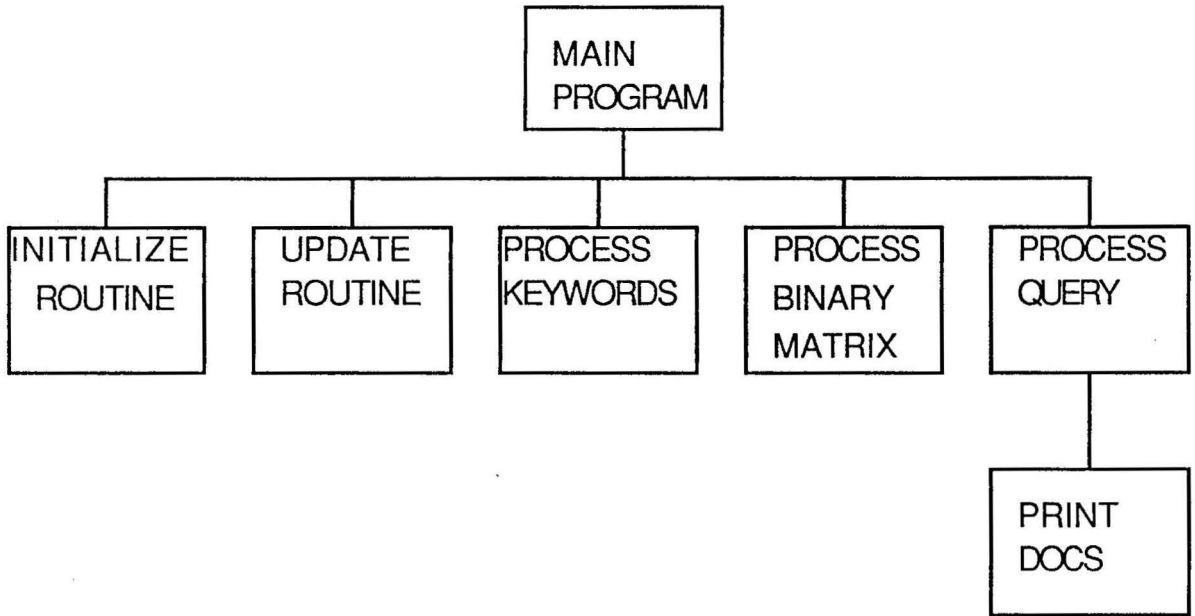


Figure 4-1 Software Design of RETRIEVE.

entered, the program puts the user back to the main menu.

Following the data entry of the database, the user can either decide to type all records in before running another module, or the user can choose to process the keywords for each document by specifying the routine "Process Keywords". The user is prompted for the number of documents to process, and then a prompt is shown asking for the number of terms to store for each document. The user then enters the keywords, keeping in mind that any errors made must be corrected through editing the appropriate files.

In this case, the corresponding files are the "Abstract File" which holds all unique terms entered in the database, "Aux File", which holds the keywords per document, and the "Ctrs Dat" file that keeps count of all database files processed, the number of unique terms used by the "Abstract File", and an array of indexes which hold the number of terms stored per document. The "Abstract File" discerns between duplicate terms entered by comparing each term with the cumulative set of unique terms entered. If there is not an existing entry, then the term is added to the file, otherwise it is discarded. The "Aux File" keeps all terms entered sequentially as they were entered. It does not discard duplicates because this will be the main file used to build the binary inverted file that looks to see if a term appears in a particular document.

The next logical step after the database records and keywords have been entered, is to process the inverted file called "Binary File". As with the initialization routine, this routine should be run once after any new entries have been entered. The routine "Process Search Array" uses the "Abstract File", and the "Aux File" to build the inverted file. The routine takes each unique term from the "Abstract File" and searches (sequentially) to see if the term is listed in the "Aux File". If it is

listed, then a "1" is entered in the position, otherwise a "0" is entered. The inverted file is a two-dimensional, 1000 x 150 array with the upper bounds representing the terms and documents respectively.

After the inverted file has been built, then the user can process queries using the "Process Query" routine. This routine allows the user to enter up to 20 keywords to search for specific documents. The module searches the "Abstract File" to make sure a match is found for the term, or prints an error message because an error was made in typing in the term. If a corresponding term was found, then the position is recorded. This position also corresponds to the term position in the inverted matrix. The searching process involves scanning across all columns representing the documents to see if a "1" or a "0" is present. If a "1" is present, then a frequency counter for the document number is incremented by one; therefore, after all terms have been processed, the "Freqctr File" is sorted in decreasing order to provide weighting or ranking of the documents. The documents with the highest query-document numbers are the more relevant.

The last procedure is the output routine "Print Docs". This procedure is automatically called by the "Process Query" routine and provides the user with a menu to specify viewing parameters. Either the user can view the documents on the CRT screen, or obtain a printout of the documents, or both. An option is also made to specify the number of documents printed. Because the program does an exhaustive search automatically, even the documents with only one match will be printed if the user does not specify otherwise. After the search is finished, the program returns to the main menu to continue processing. Examples of the sample runs of the program are found in Appendix D.

CONCLUSION AND RECOMMENDATIONS

The RETRIEVE system is capable of being a valuable and easy to use search aid for small document collections for users of Remote Sensing and Environmental Engineering studies. The system is intended to produce high precision and high recall to the users. The high recall is achieved through the structure of the inverted file, but the precision of the system is dependent on the satisfaction of the user. In order to ensure a relatively high level of success, the user should try to pinpoint very specific terms for higher precision, and more general terms for high recall. The best results should include a mixture of both combinations.

In retrospect, it is apparent where certain system features can be improved. Provisions for automatic thesaurus building should be incorporated, which would require several computations for term-term associations that would group the terms in the same class.

Another major feature would be to implement linked lists so that the system could accommodate a growing database. In conjunction with this feature, it would also be necessary to incorporate clustering techniques that will help to make the searching and retrieval process faster. With a large collection, document-document calculations would have to be made to group related documents together in clusters. Usually with large collections, a feedback option and set building feature are needed to help in the retrieval process. With the feedback option, the user can submit another query based on the more relevant set of documents combined perhaps with Boolean logic and previous data sets retrieved. Feedback helps to increase the

precision of a search by selecting or feeding back the more relevant documents with a new query formulation.

As was mentioned previously, Boolean logic would enhance any retrieval system, but would also require more expertise on the user's part. However, if enough documentation is presented, or the main operator of the system is knowledgeable, then this feature should be implemented since most retrieval systems use Boolean logic.

No software would be complete without ample error checking capabilities. The main concern though, would be to implement error handling routines in such a way that after a message is given, the program doesn't terminate, but will continue processing if the error was not very severe. Because data entry usually involves mistakes being made, it is necessary to include the majority of these error handling routines in the data entry segments of the program.

Finally, it would be a good idea to include more fields of information so that the user can have access to more information for searching. For example, information concerning date of publication, additional references, cost of the document, etc. would be useful.

In conclusion, the RETRIEVE information retrieval system is an experimental system, but it does offer an interactive environment, controlled vocabulary, and ease of use in a semi-automatic environment that is very conducive for high precision and high recall searching for a general population of users.

LIST OF REFERENCES

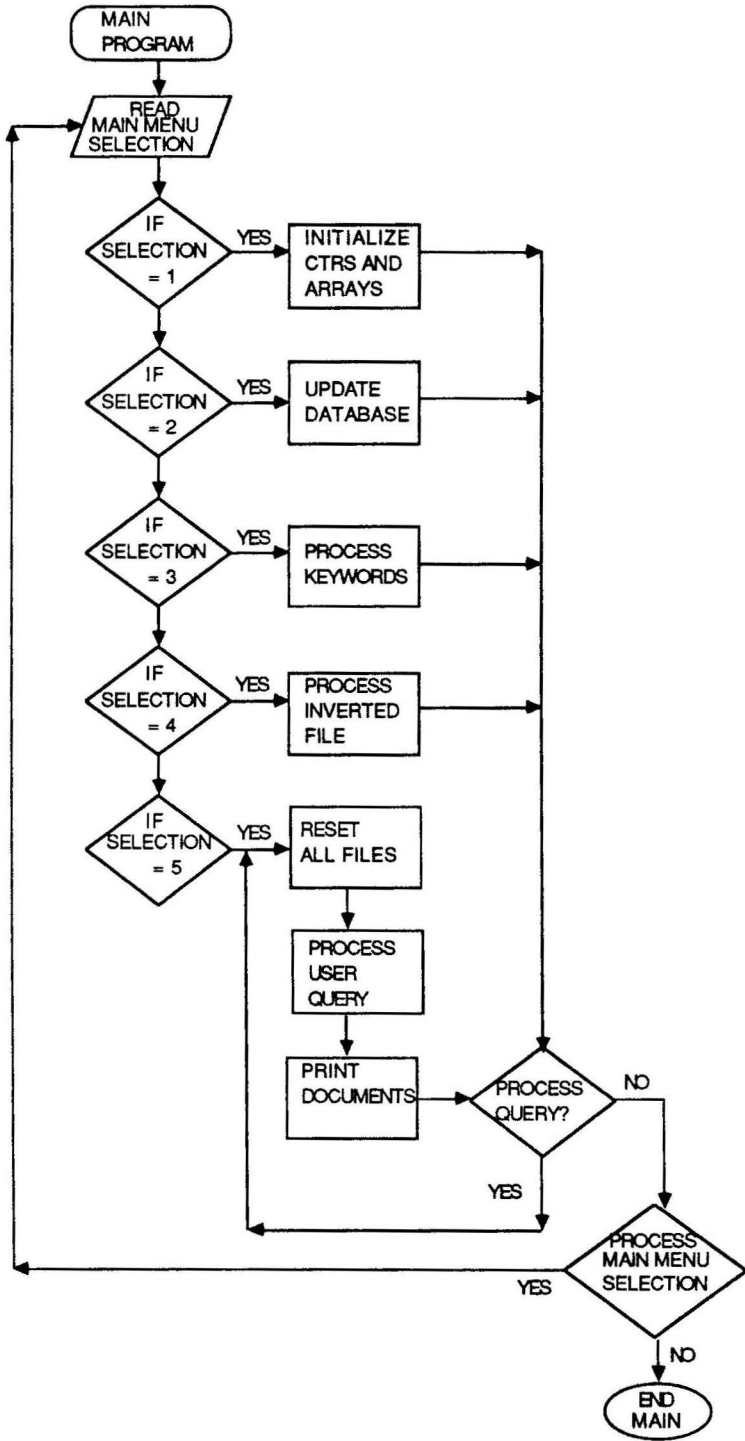
LIST OF REFERENCES

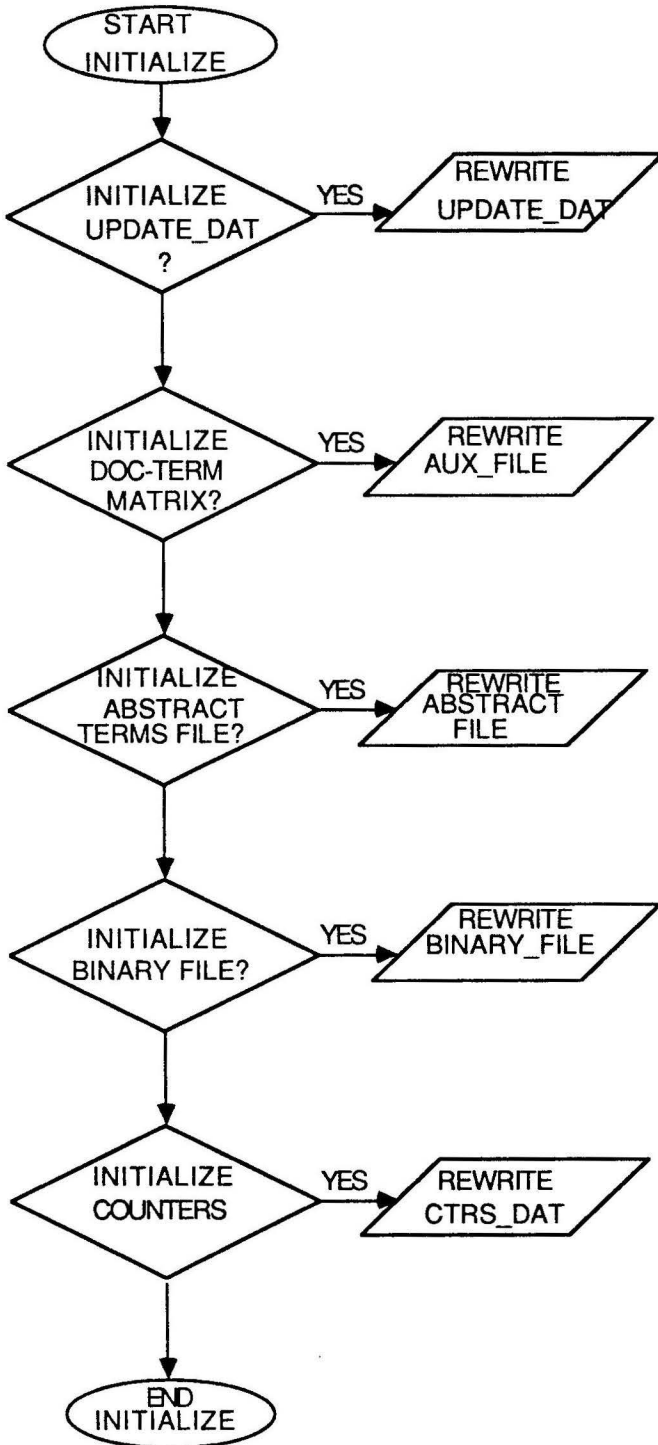
1. Luke, Gregory M., SCIPS Small Computer Information Processing System, The University of Tennessee Space Institute Remote Sensing Division, Unpublished.
2. Hutt, A.T.F., A Relational Data Base Management System, John Wiley and Sons, 1979.
3. Pulkkinen, Kyosti, "The Phases of Development of an Organization and Knowledge Representation within DSS", Proceedings of the IFIP WG 8.3 Working Conference on Knowledge Representation for Decision Support Systems. Elsevier Science Publishers B.V. (North-Holland), p.43, 1984.
4. McGill, Michael J. and Salton Gerard, Introduction to Modern Information Retrieval, McGraw-Hill Inc., 1983.
5. Salton, Gerard, A Theory of Indexing, Society for Industrial and Applied Mathematics, 1975.
6. Meadow, Charles T., The Analysis of Information Systems. Melville Publishing Co., 1973.
7. Bryant, Denise M., An Automatic Information Retrieval System for the Scientific and Engineering Computer Program Abstracts at Arnold Engineering Development Center, The University of Tennessee Space Institute M.S. Thesis, 1978.
8. Lundeen, Gerald and Tenopir, Carol, "Microcomputer Software for In-House Databases...Four Top Packages Under 2000 dollars", Online, September 1985, v9 n5, p.33.
9. Jasco, Peter, Szucs, Andras, and Verga, Sander, "MICRO-CDS/ISIS: A Bibliographic Information Management Software from UNESCO", Microcomputers for Information Management, September 1986, v3 n3, p.194.
10. Quinta, Barbara "Menlo Corporation's Pro-Search: Review of a Software Search-Aid". Online. January 1986, v10 n1, pp. 17-25.

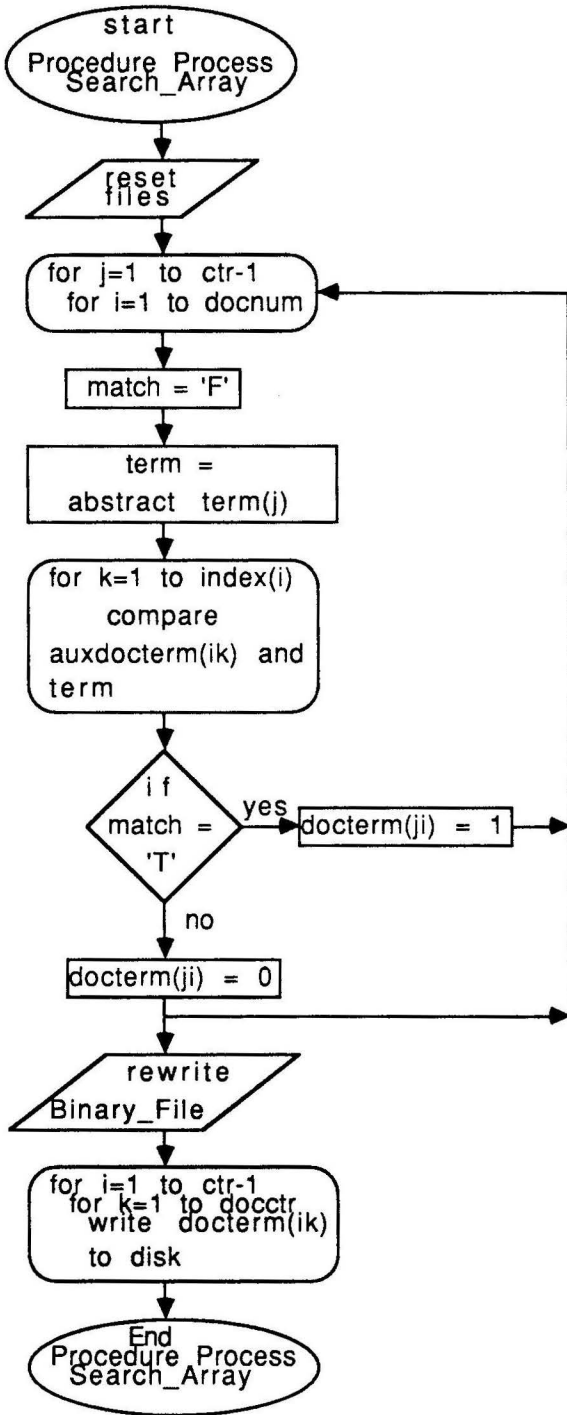
11. Witiak, Joanne "Dialoglink: A Review of Dialog's Search Assistance Software", Online, November 1986, v10 n6, pp. 39-42.
12. Keen, Peter G.W., Scott, Morton and Michael S. Decision Support Systems: An Organizational Perspective, Reading etc., 1978.

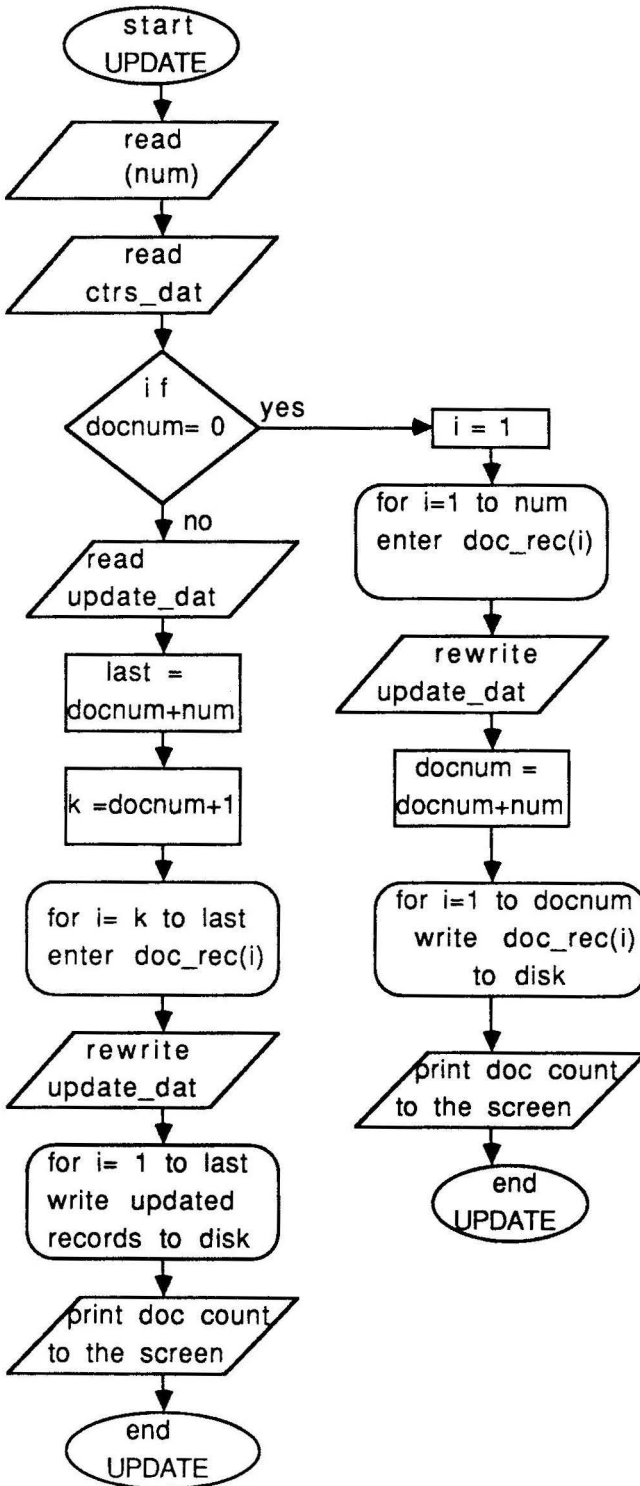
APPENDICES

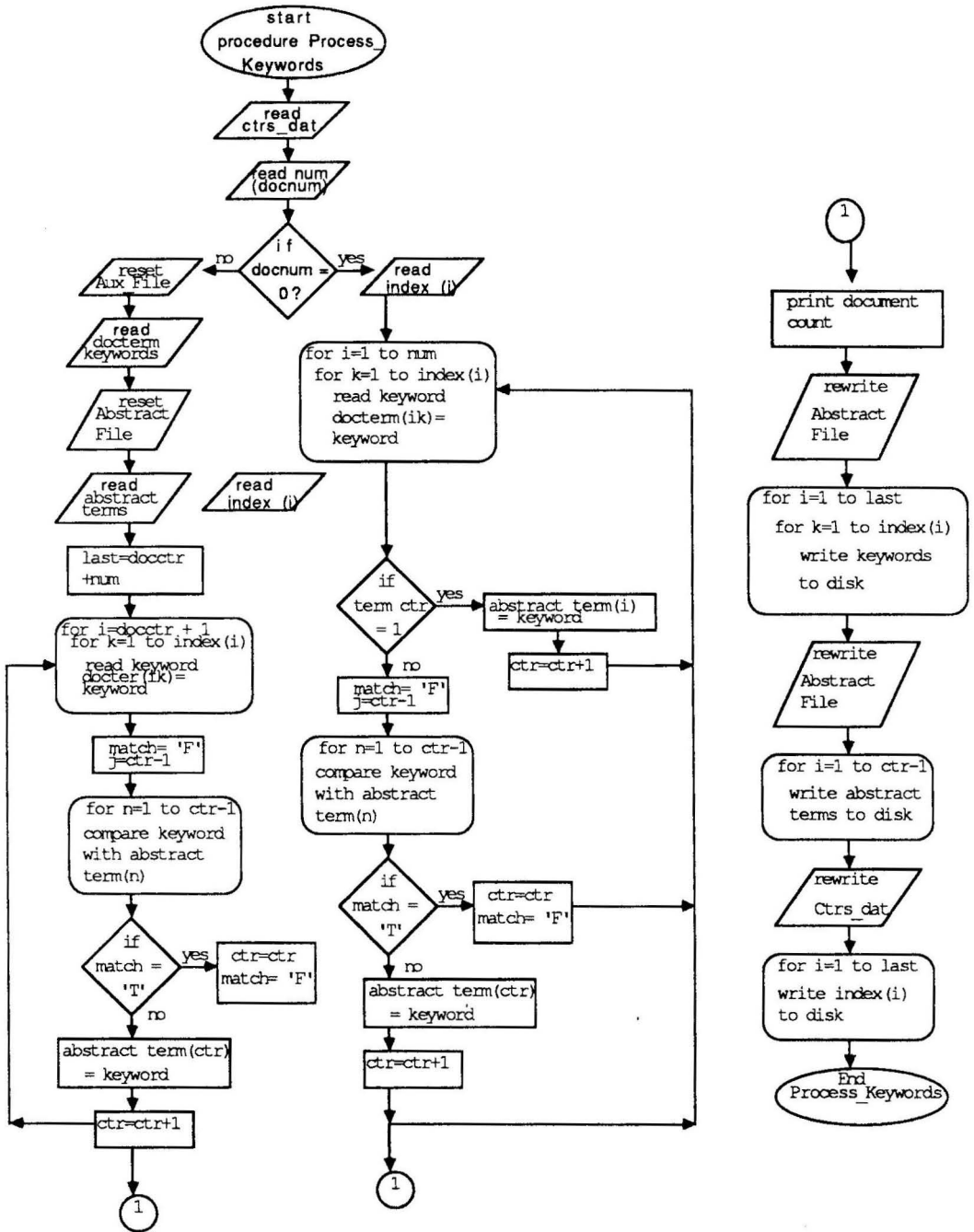
APPENDIX A
FLOW CHARTS

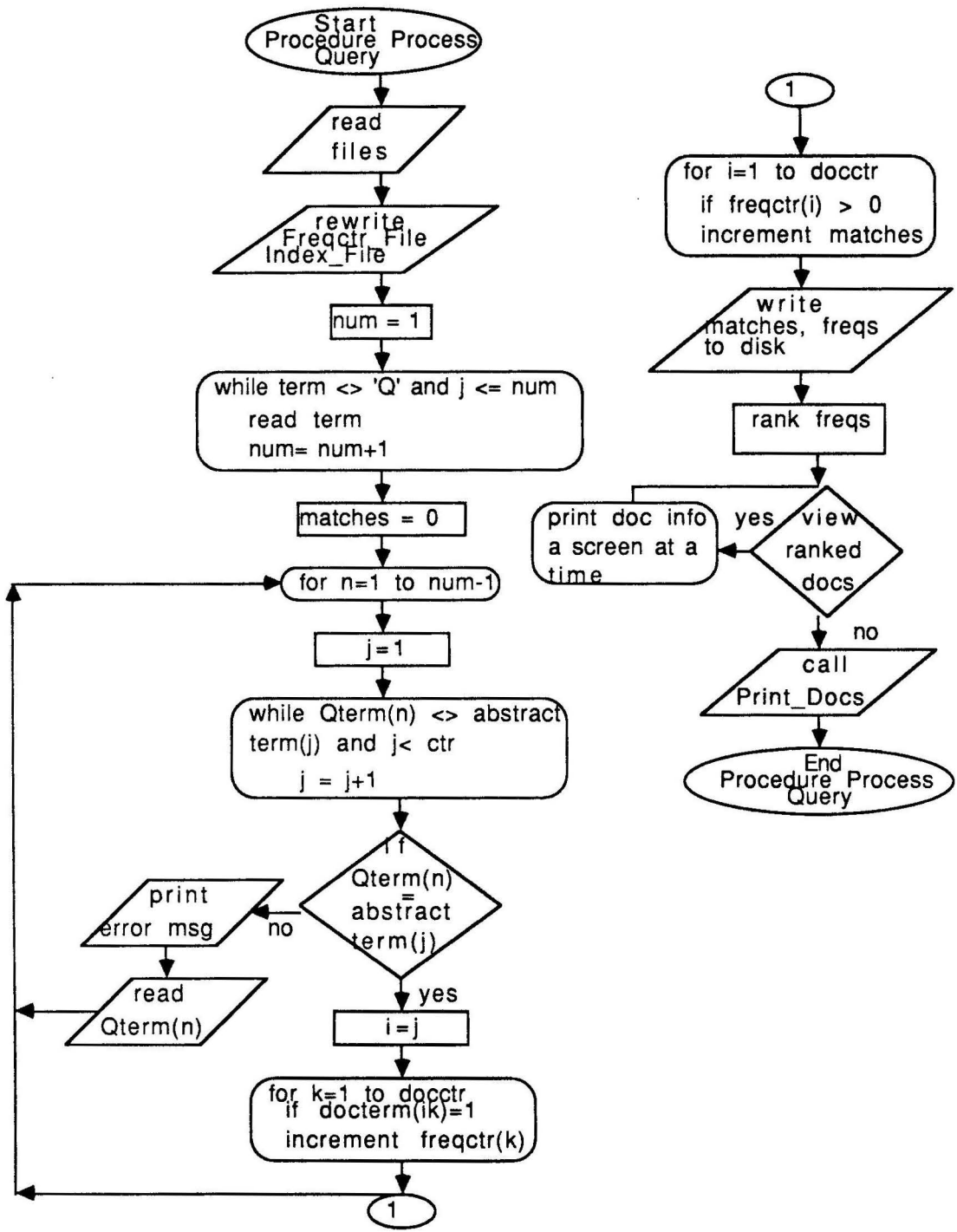


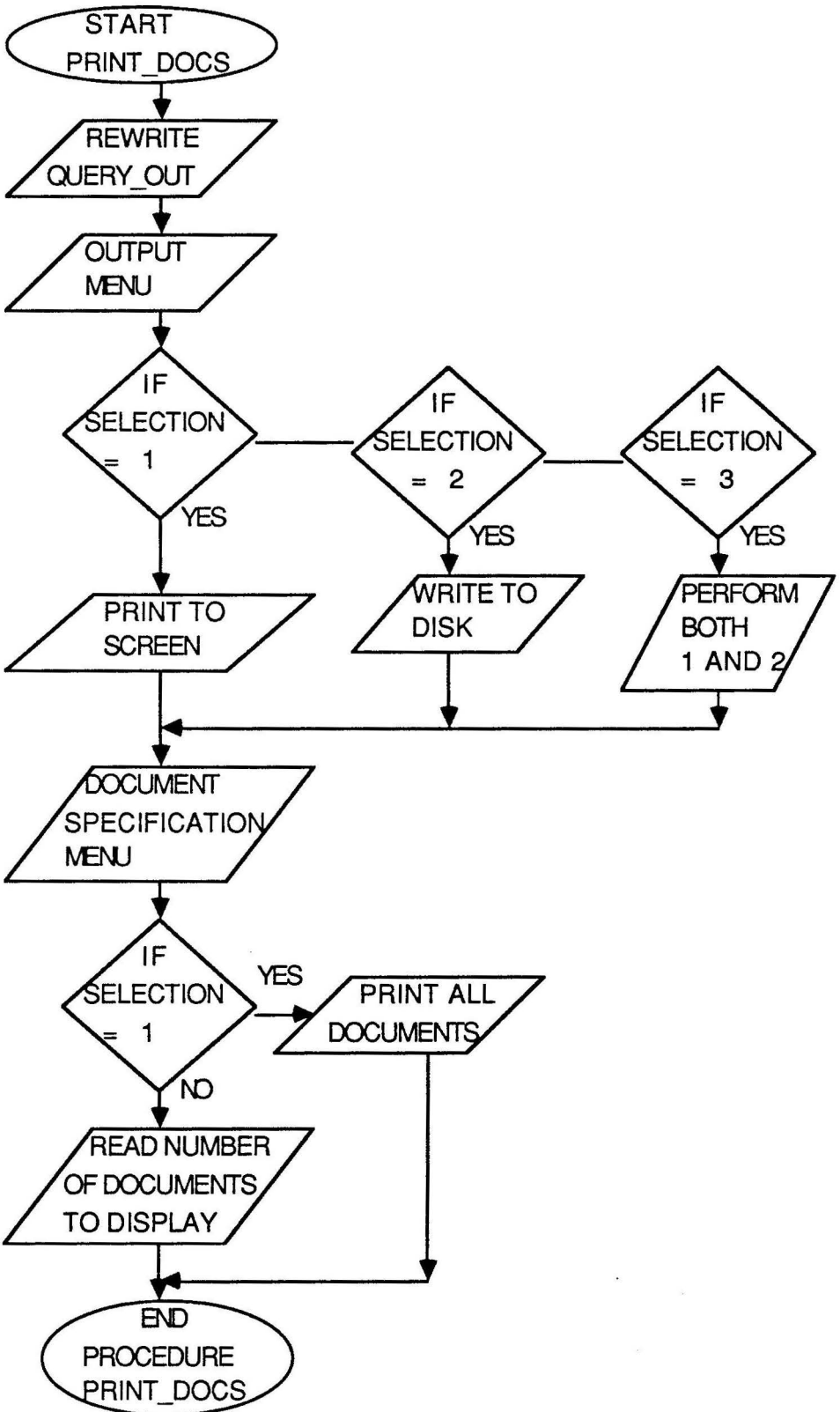












APPENDIX B
RETRIEVE PROGRAM LISTING

```

PROGRAM RETRIEVE (INPUT, OUTPUT,
    ABSTRACT_FILE, AUX_FILE, CTRS_DAT, BINARY_FILE,
    UPDATE_DAT, QUERY_OUT, TEMP_FILE, INDEX_FILE, FREQCTR_FILE);
CONST BLANKS = ' ';
BLANK = ' ';
TYPE
    INDEX_TERMS = VARYING[50] OF CHAR;
    INDEX_ARRAY = ARRAY[1..150] OF INTEGER;
    SEC_ARR = ARRAY[1..150, 1..20] OF INDEX_TERMS;
    BIN_ARR = ARRAY[1..1000, 1..150] OF CHAR;
    TERM_ARR = ARRAY[1..1000] OF INDEX_TERMS;
    {}
    STRINGS = VARYING[90] OF CHAR;
    CTR_DATA = RECORD
        DOCNUM: INTEGER;
        CTR: INTEGER;
        DOCCTR: INTEGER;
        INDEX: INDEX_ARRAY;
    END;
    DOC_DATA = RECORD
        DOCNO: VARYING[3] OF CHAR;
        COUNT: INTEGER;
        TITLE: ARRAY[1..3] OF STRINGS;
        ROWS: INTEGER;
        AUTHOR: ARRAY[1..15] OF STRINGS;
        LINES: INTEGER;
        ABSTRACT: ARRAY[1..15] OF STRINGS;
    END;
    DOC_ARRAY = ARRAY[1..150] OF DOC_DATA;
    {}
{***** DEFINITION OF VARIABLES IN MAIN PROCEDURE *****}
{}
{AUX_FILE, ABSTRACT_FILE, CTRS_FILE AND BINARY_FILE- EXTERNAL FILES}
{STORED IN SECONDARY MEMORY AND READ INTO PRIMARY MEMORY BY PROGRAM}
{RETRIEVE. THESE FILES FUNCTION AS BOTH INPUT AND OUTPUT FILES }
{KEYWORDS: A STRING ARRAY THAT HOLDS THE KEYWORDS OF EA. DOCUMENT}
{DOCNUM: REFERS TO THE DOCUMENT NUMBER TO PROCESS }
{CTR: COUNTS TOTAL NUMBER OF TERMS IN DOCUMENT COLLECTION }
{AUX_DOCTERM: A SECONDARY MATRIX CONTAINING KEYWORDS/DOCUMENT }
{USED TO BUILD ABSTRACT TERMS ARRAY AND BINARY MATRIX }
{DOC_TERM: THE BINARY DOCUMENT-TERM MATRIX FOR ENTIRE COLLECTION}
{ABSTRACT_TERMS: A ONE-DIMENSIONAL ARRAY CONTAINING ALL TERMS }
{STRINGS: TYPE OF INPUT OF ABSTRACTS }
{DOC_DATA: RECORD STRUCTURE OF DOCUMENT INFORMATION }
{DOC_ARRAY: ARRAY THAT STORES RECORD INFORMATION }
{CTR_DATA: RECORD STRUCTURE THAT STORES VARIOUS INDEXES }
{DOC_RECS: HOLDS DOCUMENT INFORMATION }
{RESPONSE: CHARACTER RESPONSE TO INTERACTIVE PROMPTS }
{*****}
{}
VAR
    CTR_REC: CTR_DATA;
    QUERY_OUT, AUX_FILE, ABSTRACT_FILE, BINARY_FILE: TEXT;
    CTRS_DAT, TEMP_FILE, INDEX_FILE, FREQCTR_FILE: TEXT;
    UPDATE_DAT: TEXT;
    KEYWORDS: INDEX_TERMS;
    AUX_DOCTERM: SEC_ARR;
    DOC_TERM: BIN_ARR;
    ABSTRACT_TERMS: TERM_ARR;
    DOC_RECS: DOC_ARRAY;

```

```

DOCCTR, INDEX, I, N, K: INTEGER;
RESPONSE, REQUEST, CHARACTR: CHAR;
{}
{}
{*****          PROCEDURE UPDATE          *****}
{
{PROCEDURE UPDATE PROVIDES AN INTERACTIVE ENVIRONMENT FOR THE }
{USER TO ENTER UP TO 150 DOCUMENT RECORDS                       }
{
{VARIABLES: I IS AN INDEX TO THE ARRAY OF RECORDS DOC_RECS     }
{      K, N, J ARE VARIOUS LOOP INDEXES                         }
{      NUM HOLDS THE CURRENT NUMBER OF RECORDS TO PROCESS     }
{      LINES IS AN INDEX TO THE ABSTRACT LINES ARRAY          }
{GLOBAL VARIABLES: DOC_RECS                                     }
{
{*****}
{}
PROCEDURE UPDATE (VAR CTR_REC: CTR_DATA; VAR DOC_RECS: DOC_ARRAY; VAR
      UPDATE_DAT: TEXT; VAR CTRS_DAT: TEXT);
VAR
      I, K, N, NUM, J, LINES, LAST: INTEGER;
      REQUEST: CHAR;
BEGIN
      WRITE ('ENTER NUMBER OF DOCUMENTS TO PROCESS :');
      READLN (NUM);
      WRITELN (' ');
      RESET (CTRS_DAT);
      WITH CTR_REC DO
      BEGIN
          READLN (CTRS_DAT, DOCNUM);
          READLN (CTRS_DAT, CTR);
          READLN (CTRS_DAT, DOCCTR);
          FOR I:= 1 TO DOCCTR DO
              READLN (CTRS_DAT, INDEX[I]);
          END;
          IF (CTR_REC.DOCNUM = 0) THEN
              BEGIN
                  FOR I:=1 TO NUM DO
                      BEGIN
                          WRITELN ('ENTER DOC NUMBER');
                          READLN (DOC_RECS [I].DOCNO);
                          WRITELN (UPDATE_DAT, DOC_RECS [I].DOCNO);
                          {}
                          WRITELN ('ENTER TITLE');
                          WITH DOC_RECS [I] DO
                              BEGIN
                                  COUNT:=1;
                                  READLN (TITLE [COUNT]);
                                  WHILE (TITLE [COUNT] <> '/') DO
                                      BEGIN
                                          COUNT:=COUNT+1;
                                          READLN (TITLE [COUNT]);
                                      END;
                                  WRITELN (UPDATE_DAT, COUNT);
                                  FOR N:= 1 TO COUNT DO
                                      WRITELN (UPDATE_DAT, TITLE [N]);
                                  END;
                                  {}
                                  WRITELN ('ENTER AUTHOR');
                                  WITH DOC_RECS [I] DO

```

```

BEGIN
    ROWS:=1;
    READLN(AUTHOR[ROWS]);
WHILE (AUTHOR[ROWS] <> '/') DO
    BEGIN
        ROWS:=ROWS+1;
        READLN(AUTHOR[ROWS]);
    END;
    WRITELN(UPDATE_DAT,ROWS);
    FOR N:= 1 TO ROWS DO
        WRITELN(UPDATE_DAT,AUTHOR[N]);
    END;
{}
WRITELN('ENTER ABSTRACT');
WITH DOC_RECS[I] DO
    BEGIN
        LINES:=1;
        READLN(ABSTRACT[LINES]);
        WHILE(ABSTRACT[LINES] <> '/') DO
            BEGIN
                LINES:=LINES+1;
                READLN(ABSTRACT[LINES]);
            END;
        WRITELN(UPDATE_DAT,LINES);
        FOR N:= 1 TO LINES DO
            WRITELN(UPDATE_DAT,ABSTRACT[N]);
            WRITELN(UPDATE_DAT);
        END;
    END;
    CTR_REC.DOCNUM:=CTR_REC.DOCNUM+NUM;
    WRITELN(' ');
    WRITELN('NUMBER OF DOCUMENTS PROCESSED :',CTR_REC.DOCNUM:2);
    REWRITE(CTRS_DAT);
    WITH CTR_REC DO
        BEGIN
            WRITELN(CTRS_DAT,DOCNUM);
            WRITELN(CTRS_DAT,CTR);
            WRITELN(CTRS_DAT,DOCCTR);
            FOR I:= 1 TO DOCCTR DO
                WRITELN(CTRS_DAT,INDEX[I]);
                WRITELN(' ');
                WRITELN(' ');
            END;
        END
    ELSE
        BEGIN
            WRITELN(' ');
            RESET(UPDATE_DAT);
            FOR I:= 1 TO CTR_REC.DOCNUM DO
                BEGIN
                    WITH DOC_RECS[I] DO
                        BEGIN
                            {}
                            {}
                            READLN(UPDATE_DAT,DOCNO);
                            READLN(UPDATE_DAT,COUNT);
                            FOR N:= 1 TO COUNT DO
                                READLN(UPDATE_DAT,TITLE[N]);
                                READLN(UPDATE_DAT,ROWS);

```



```

        FOR N:= 1 TO ROWS DO
            READLN(UPDATE_DAT,AUTHOR[N]);
            READLN(UPDATE_DAT,LINES);
        FOR N:= 1 TO LINES DO
            READLN(UPDATE_DAT,ABSTRACT[N]);
            READLN(UPDATE_DAT);
        END;
    END;
LAST:=CTR_REC.DOCNUM + NUM;
FOR I:=CTR_REC.DOCNUM+1 TO LAST DO
    BEGIN
    {}
    { READ IN NEW RECORDS INTO STRUCTURE      }
    {}
    WITH DOC_RECS[I] DO
        BEGIN
            WRITELN('ENTER DOC NUMBER');
            READLN(DOCNO);
        {}
            WRITELN('ENTER TITLE');
            COUNT:=1;
            READLN(TITLE[COUNT]);
            WHILE (TITLE[COUNT] <> '/') DO
                BEGIN
                    COUNT:=COUNT+1;
                    READLN(TITLE[COUNT]);
                END;
        {}
            WRITELN('ENTER AUTHOR');
            ROWS:=1;
            READLN(AUTHOR[ROWS]);
            WHILE (AUTHOR[ROWS] <> '/') DO
                BEGIN
                    ROWS:=ROWS+1;
                    READLN(AUTHOR[ROWS]);
                END;
        {}
            WRITELN('ENTER ABSTRACT');
            LINES:=1;
            READLN(ABSTRACT[LINES]);
            WHILE (ABSTRACT[LINES] <> '/') DO
                BEGIN
                    LINES:=LINES+1;
                    READLN(ABSTRACT[LINES]);
                END;
            END;
        END;
    END;
    {}
    { WRITE RECORDS TO UPDATED FILE }
    {}
    REWRITE(UPDATE_DAT);
    FOR I:= 1 TO LAST DO
        BEGIN
            WITH DOC_RECS[I] DO
                BEGIN
                    WRITELN(UPDATE_DAT,DOCNO);
                    WRITELN(UPDATE_DAT,COUNT);
                    FOR N:= 1 TO COUNT DO
                        WRITELN(UPDATE_DAT,TITLE[N]);
                    WRITELN(UPDATE_DAT,ROWS);
                END;
            END;
        END;
    END;

```

```

FOR N:= 1 TO ROWS DO
  WRITELN(UPDATE_DAT,AUTHOR[N]);
  WRITELN(UPDATE_DAT,LINES);
FOR N:= 1 TO LINES DO
  WRITELN(UPDATE_DAT,ABSTRACT[N]);
  WRITELN(UPDATE_DAT);
END;
END;
WRITELN('NUMBER OF DOCUMENTS PROCESSED :',NUM:2);
WRITELN(' ');
CTR_REC.DOCNUM:=LAST;
REWRITE(CTRS_DAT);
WITH CTR_REC DO
  BEGIN
    WRITELN(CTRS_DAT,CTR_REC.DOCNUM);
    WRITELN(CTRS_DAT,CTR);
    WRITELN(CTRS_DAT,DOCCTR);
    FOR I:= 1 TO DOCCTR DO
      WRITELN(CTRS_DAT,INDEX[I]);
    END;
  WRITELN(' ');
  WRITELN('TOTAL NUMBER OF DOCUMENTS PROCESSED :',CTR_REC.DOCNUM:2);
  WRITELN(' ');
  WRITELN(' ');
END;
END;
{}
{}
{*****          PROCEDURE INITLIZE          ***** }
{
{PROCEDURE INITLIZE INITIALIZES MAIN AND SECONDARY ARRAYS }
{VARIABLES: I,K REPRESENT DOCUMENT AND TERM COUNTERS, RESPECTIVELY}
{GLOBAL VARIABLES: AUX_DOCTERM ,ABSTRACT_TERMS,DOC_TERM }
{
{*****          }
{}
PROCEDURE INITLIZE(VAR AUX_DOCTERM:SEC ARR;VAR ABSTRACT_TERMS:TERM_ARR;
  VAR DOC_TERM:BIN_ARR;VAR CTR_REC:CTR_DATA;
  VAR AUX_FILE:TEXT;VAR ABSTRACT_FILE:TEXT;VAR CTRS_DAT:
  TEXT;VAR BINARY_FILE:TEXT;VAR UPDATE_DAT:TEXT;
  VAR KEYWORDS:INDEX_TERMS);
CONST BLANKS = '
ZERO = '0';
VAR I,K,J:INTEGER;
REQUEST:CHAR;
BEGIN
  WRITELN('INITIALIZE UPDATE FILE? Y/N');
  READLN(REQUEST);
  IF(REQUEST = 'Y') THEN
    REWRITE(UPDATE_DAT);
  {}
  WRITELN('INITIALIZE AUX_DOCTERM MATRIX? Y/N');
  READLN(REQUEST);
  IF(REQUEST = 'Y') THEN
    BEGIN
      RESET(AUX_FILE);
      REWRITE(AUX_FILE);
      FOR I:=1 TO 150 DO
        BEGIN
          FOR K:=1 TO 20 DO

```

```

        WRITELN(AUX_FILE, BLANKS);
        WRITELN(AUX_FILE);
    END;
END;
{}
WRITELN;
WRITELN;
WRITELN(' INITIALIZE ABSTRACT_TERMS MATRIX AND KEYWORDS? Y/N');
READLN(REQUEST);
IF (RESPONSE = 'Y') THEN
    BEGIN
        RESET(ABSTRACT_FILE);
        REWRITE(ABSTRACT_FILE);
        KEYWORDS:=BLANKS;
        FOR I:= 1 TO 1000 DO
            BEGIN
                ABSTRACT_TERMS[I]:=BLANKS;
                WRITELN(ABSTRACT_FILE, ABSTRACT_TERMS[I]);
            END;
        END;
    END;
{}
WRITELN;
WRITELN;
WRITELN(' INITIALIZE DOC-TERM MATRIX? Y/N');
READLN(REQUEST);
IF (REQUEST = 'Y') THEN
    BEGIN
        RESET(BINARY_FILE);
        REWRITE(BINARY_FILE);
        FOR I:= 1 TO 1000 DO
            BEGIN
                FOR K:= 1 TO 150 DO
                    WRITE(BINARY_FILE, ZERO);
                    WRITELN(BINARY_FILE);
                END;
            END;
        END;
    END;
{}
WRITELN;
WRITELN;
WRITELN(' INITIALIZE DOCNUM, TERM CTR, AND DOCCTR? Y/N');
READLN(REQUEST);
IF (REQUEST = 'Y') THEN
    BEGIN
        RESET(CTRS_DAT);
        REWRITE(CTRS_DAT);
        WITH CTR_REC DO
            BEGIN
                DOCNUM:=0;
                CTR:=1;
                DOCCTR:=0;
                WRITELN(CTRS_DAT, DOCNUM);
                WRITELN(CTRS_DAT, CTR);
                WRITELN(CTRS_DAT, DOCCTR);
                FOR I:= 1 TO 150 DO
                    BEGIN
                        INDEX[I]:=0;
                        WRITELN(CTRS_DAT, INDEX[I]);
                    END;
                END;
            END;
        END;
    END;
END;

```

```

{}
END;
{}
{}
{ *****      PROCEDURE PROCESS_KEYWORDS      ***** }
{
{PROCEDURE PROCESS_KEYWORDS ALLOWS THE USER TO SPECIFY THE }
{DOCUMENT NUMBER TO PROCESS. THE USER ENTERS UP TO 10 KEY- }
{WORDS/DOCUMENT AND THE PROGRAM ENTERS THE KEYWORDS IN THE }
{AUX DOCTERM MATRIX. THE PROGRAM UPDATES THE ABSTRACT TERMS }
{MATRIX BY FIRST SEARCHING FOR DUPLICATE ENTRIES, AND ENTERS }
{ONLY UNIQUE TERMS IN THE MATRIX. }
{
{VARIABLES: NUM, DOCNUM ARE COUNTERS FOR DOCUMENT NUMBER }
{      K, INDEX, N ARE LOOP COUNTERS }
{GLOBAL VARIABLES: DOCNUM HOLDS NUMBER OF DOCUMENTS PROCESSED }
{      CTR HOLDS THE TOTAL NUMBER OF UNIQUE TERMS }
{      AUX_DOCTERM HOLDS THE DOC-TERM MATRIX OF TERMS }
{      ABSTRCT_TERMS HOLDS THE UNIQUE TERMS }
{ ***** }
{}
PROCEDURE PROCESS_KEYWORDS (VAR CTR_REC:CTR DATA;VAR AUX DOCTERM:SEC_ARR;
      VAR ABSTRACT_TERMS:TERM_ARR;VAR KEYWORDS:
      INDEX_TERMS;VAR CTRS_DAT:TEXT;VAR AUX_FILE:TEXT;
      VAR ABSTRACT_FILE:TEXT);
CONST BLANK = ' ';
VAR
      LAST, I, NUM, J, K, N, COUNT:INTEGER;
      CHARACTR, REQUEST, MATCH:CHAR;
BEGIN
      RESET (CTRS_DAT);
      WITH CTR_REC DO
      BEGIN
            READLN (CTRS_DAT, DOCNUM);
            READLN (CTRS_DAT, CTR);
            READLN (CTRS_DAT, DOCCTR);
            FOR I:= 1 TO DOCCTR DO
                  READLN (CTRS_DAT, INDEX[I]);
            {}
      END;
      WRITELN;
      WITH CTR_REC DO
      BEGIN
            WRITELN ('ENTER NUMBER OF DOCUMENTS TO PROCESS');
            READLN (NUM);
            IF (DOCCTR = 0) THEN
                  BEGIN
                        WRITELN;
                        FOR I:=1 TO NUM DO
                              BEGIN
                                    WRITELN ('ENTER NUMBER OF TERMS TO PROCESS (FOR DOCUMENT', I:2);
                                    READLN (INDEX[I]);
                                    FOR K:=1 TO INDEX[I] DO
                                          BEGIN
                                                WRITE ('TERM NO', K:3);
                                                WRITE (' ');
                                                READLN (KEYWORDS);
                                                WRITELN;
                                                AUX_DOCTERM [I, K] :=KEYWORDS;
                                                IF (CTR = 1) THEN

```

```

        BEGIN
            ABSTRACT_TERMS [CTR] :=KEYWORDS;
        CTR:=CTR+1;
        END
    ELSE
        BEGIN
            MATCH:='F';
            J:=CTR-1;
        { SEARCH FOR DUPLICATE TERMS IN ABSTRACT_TERMS ARRAY}
            FOR N:=1 TO J DO
                BEGIN
                    IF (KEYWORDS = ABSTRACT_TERMS [N] ) THEN
                        MATCH:='T';
                    END;
                    IF (MATCH = 'T') THEN
                        BEGIN
                            CTR:=CTR;
                            MATCH:='F';
                        END
                    ELSE
                        BEGIN
                            ABSTRACT_TERMS [CTR] :=KEYWORDS;
                            CTR:=CTR+1;
                        END;
                    END;
                END;
            END;
        WRITELN;
        WRITELN ('NUMBER OF DOCUMENTS PROCESSED:', NUM:2);
        WRITELN;
        WRITELN ('TOTAL NO. OF UNIQUE KEYWORDS:', CTR-1:2);
    {}
    REWRITE (CTRS_DAT);
    WRITELN (CTRS_DAT, DOCNUM);
    WRITELN (CTRS_DAT, CTR);
    WRITELN (CTRS_DAT, NUM);
    FOR I:= 1 TO NUM DO
        WRITELN (CTRS_DAT, INDEX[I]);
    {}
    REWRITE (ABSTRACT_FILE);
    FOR I:= 1 TO CTR-1 DO
        WRITELN (ABSTRACT_FILE, ABSTRACT_TERMS [I]);
    {}
    REWRITE (AUX_FILE);
    FOR I:=1 TO NUM DO
        BEGIN
            FOR K:= 1 TO INDEX[I] DO
                WRITELN (AUX_FILE, AUX_DOCTERM [I, K]);
                WRITELN (AUX_FILE, BLANKS);
            END;
        END
    ELSE
        BEGIN
            RESET (AUX_FILE);
            FOR I:= 1 TO DOCCTR DO
                BEGIN
                    FOR K:= 1 TO INDEX[I] DO
                        READLN (AUX_FILE, AUX_DOCTERM [I, K]);
                        READLN (AUX_FILE);
                    END;
                END;
            END;
        END;
    END;

```

```

{}
RESET(ABSTRACT_FILE);
FOR I:= 1 TO CTR-1 DO
    READLN(ABSTRACT_FILE,ABSTRACT_TERMS[I]);
{}
LAST:=DOCCTR+NUM;
FOR I:= DOCCTR+1 TO LAST DO
    BEGIN
        WRITELN('ENTER NUMBER OF TERMS FOR DOCUMENT ',I:2);
        READLN(INDEX[I]);
        FOR K:= 1 TO INDEX[I] DO
            BEGIN
                WRITE('TERM NO. ',K:2);
                WRITE(' ');
                READLN(KEYWORDS);
                WRITELN;
                AUX_DOCTERM[I,K]:=KEYWORDS;
                MATCH:='F';
                J:=CTR-1;
                { SEARCH FOR DUPLICATE TERMS IN ABSTRACT_TERMS ARRAY}
                FOR N:=1 TO J DO
                    BEGIN
                        IF (KEYWORDS = ABSTRACT_TERMS[N] ) THEN
                            MATCH:='T';
                        END;
                    IF (MATCH = 'T') THEN
                        BEGIN
                            CTR:=CTR;
                            MATCH:='F';
                        END
                    ELSE
                        BEGIN
                            ABSTRACT_TERMS[CTR]:=KEYWORDS;
                            CTR:=CTR+1;
                        END;
                    END;
                END;
            END;
        WRITELN;
        WRITELN('NUMBER OF DOCUMENTS PROCESSED: ',NUM:2);
        WRITELN;
        WRITELN('TOTAL NO. OF UNIQUE KEYWORDS: ',CTR-1:2);
    {}
        REWRITE (CTRS_DAT);
        WRITELN(CTRS_DAT,DOCNUM);
        WRITELN(CTRS_DAT,CTR);
        WRITELN(CTRS_DAT, LAST);
        FOR I:= 1 TO LAST DO
            WRITELN(CTRS_DAT, INDEX[I]);
        {}
        REWRITE(ABSTRACT_FILE);
    FOR I:= 1 TO CTR-1 DO
        WRITELN(ABSTRACT_FILE,ABSTRACT_TERMS[I]);

REWRITE(AUX_FILE);
FOR I:=1 TO LAST DO
    BEGIN
        FOR K:= 1 TO INDEX[I] DO
            WRITELN(AUX_FILE,AUX_DOCTERM[I,K]);
            WRITELN(AUX_FILE,BLANKS);
        END;

```

```

        END;
    END;
END;
{}
{ ***** PROCEDURE PROCESS_SEARCH_ARRAY ***** }
{ PROCEDURE PROCESS_SEARCH_ARRAY COMPARES THE AUX_DOC-ERM ARRAY CONTAIN- }
{ THE ORIGINAL KEYWORDS FOR EACH DOCUMENT WITH THE ABSTRACT TERMS ARRAY }
{ THAT HOLDS ALL UNIQUE TERMS IN THE COLLECTION. IF FOR EACH DOCUMENT(i) }
{ TERM(k) IS PRESENT, A 1 IS ENTERED IN THE CORRESPONDING POSITION IN THE }
{ THE BINARY DOC-TERM MATRIX. }
{ }
{ VARIABLES:K,J ARE LOOP INDEXES AND ARE INDEXES FOR AUX_DOCTERM ARRAY }
{ I IS A LOOP INDEX AND IS THE INDEX FOR ABSTRACT TERMS ARRAY }
{ MATCH TELLS WHETHER A MATCH WAS FOUND IN THE ORIGINAL DOC-TERM }
{ MATRIX AND THE ABSTRACT_TERMS MATRIX }
{ }
{ GLOBAL VARIABLES:AUX_DOCTERM,ABSTRACT_TERMS,CTR,DOC_TERM,DOCNUM, }
{ }
{ ***** }
{}
PROCEDURE PROCESS_SEARCH_ARRAY(VAR AUX_DOCTERM:SEC_ARR;VAR ABSTRACT_TERMS:
TERM_ARR;VAR DOC_TERM:BIN_ARR;VAR CTR_REC:CTR_DATA;VAR AUX_FILE:TEXT;
VAR ABSTRACT_FILE:TEXT;VAR CTRS_DAT:TEXT;VAR BINARY_FILE:TEXT);
    VAR
        I,K,J,N:INTEGER;
        TERM:VARYING[50] OF CHAR;
        MATCH:CHAR;
BEGIN
    RESET(CTRS_DAT);
    RESET(AUX_FILE);
    RESET(ABSTRACT_FILE);
    RESET(BINARY_FILE);
    WITH CTR_REC DO
        BEGIN
            READLN(CTRS_DAT,DOCNUM);
            READLN(CTRS_DAT,CTR);
            READLN(CTRS_DAT,DOCCTR);
            FOR I:= 1 TO DOCCTR DO
                READLN(CTRS_DAT,INDEX[I]);
            {}
            FOR I:= 1 TO DOCCTR DO
                BEGIN
                    FOR K:= 1 TO INDEX[I] DO
                        READLN(AUX_FILE,AUX_DOCTERM[I,K]);
                        READLN(AUX_FILE);
                    END;
                {}
                FOR I:=1 TO CTR-1 DO
                    READLN(ABSTRACT_FILE,ABSTRACT_TERMS[I]);
                END;
            {}
            WITH CTR_REC DO
                BEGIN
                    FOR J:=1 TO CTR-1 DO {TOTAL TERMS}
                        BEGIN
                            FOR I:=1 TO DOCNUM DO {TOTAL DOCUMENTS}
                                BEGIN
                                    TERM:=ABSTRACT_TERMS[J];
                                    MATCH:='F';
                                    FOR K:=1 TO INDEX[I] DO

```

```

        BEGIN
            IF (AUX_DOCTERM[I,K] = TERM) THEN
                MATCH:='T';
            END;
        IF (MATCH = 'T') THEN
            DOC_TERM[J,I]:='1'
        ELSE
            DOC_TERM[J,I]:='0';
        END;
    END;
END;
WRITELN;
WRITELN;
REWRITE (BINARY FILE);
    FOR I:=1 TO CTR-1 DO
        BEGIN
            FOR K:= 1 TO DOCCTR DO
                WRITE (BINARY_FILE,DOC_TERM[I,K]:1);
                WRITELN (BINARY_FILE);
            END;
        END;
END;
END;
{}
{ ***** PROCEDURE PROCESS_QUERY ***** }
{PROCEDURE PROCESS_QUERY ALLOWS THE USER TO ENTER UP TO 20 WORDS FOR }
{RETRIEVAL PURPOSES. THE QUERY VECTOR, QUERY TERMS, IS SIMILAR IN }
{STRUCTURE TO ABSTRACT_TERMS (WHICH HOLDS ALL UNIQUE TERMS). THE }
{ABSTRACT_TERM ARRAY IS SEARCHED AGAINST EACH WORD ENTERED BY THE USER }
{AND A COUNTER,COUNT,IS SET UP SO THAT IF THERE IS A MATCH THEN THE }
{POSITION IS RECORDED FOR THE Nth(COUNT) TERM IN THE QUERY VECTOR }
{ }
{VARIABLES: TEMP IS THE TEMPORARY HOLDING ARRAY FOR THE USER'S ENTRIES }
{ Q_TERMS IS THE STRING STRUCTURE FOR THE INPUT }
{ I IS THE INDEX TO THE QUERY VECTOR }
{ N IS A COUNTER FOR THE NUMBER OF USER ENTRIES }
{ COUNT IS A COUNTER FOR THE TERM POSITION IN THE QUERY VECTOR }
{ MATCH TELLS WHETHER THE USER TERM MATCHES A STORED TERM }
{ }
{GLOBAL VARIABLES: DOC_TERM, ABSTRACT_TERMS, CTR, DOCNUM }
{*****}
{}
PROCEDURE PROCESS_QUERY (VAR DOC_TERM:BIN_ARR;VAR BINARY_FILE:TEXT;
    VAR CTR_REC:CTR_DATA;VAR CTRS_DAT:TEXT;
    VAR ABSTRACT_TERMS:TERM_ARR;VAR ABSTRACT_FILE:TEXT;
    VAR DOC_RECS:DOC_ARRAY;VAR UPDATE_DAT:TEXT;
    VAR QUERY_OUT:TEXT;VAR TEMP_FILE:TEXT;
    VAR INDEX_FILE:TEXT;VAR FREQCTR_FILE:TEXT);
CONST BLANK=' ';
LABEL 1;
TYPE
    WORDS = VARYING[50] OF CHAR;
    DOCFREQ = ARRAY[1..150] OF INTEGER;
    TEMP_ARR = ARRAY[1..20] OF WORDS;
    INDEX_ARR = ARRAY[1..150] OF INTEGER;
VAR
    Q_TERMS:TEMP_ARR;
    I, J, K, NUM, N, TEMP, CURRENT, LAST, MATCHES, COUNT: INTEGER;
    FREQCTR:DOCFREQ;
    INDEX_VECTOR:INDEX_ARR;
    REQUEST:CHAR;
{}

```



```

{*****}
{      PROCEDURE PRINT_DOCS                                     }
{  PROCEDURE PRINT_DOCS PRINTS PROVIDES A MENU FOR THE USER  }
{  TO SELECT DIFFERENT OPTIONS FOR PRINTING THE RESULTS OF THE }
{  SEARCH. THE PROCEDURE ACCEPTS PARAMATERS MATCHES AND      }
{  INDEX_VECTOR FROM PROCEDURE PROCESS_QUERY                  }
{                                                              }
{  GLOBAL VARIABLES: DOC_RECS                                }
{                                                              }
{*****}
PROCEDURE PRINT_DOCS (VAR MATCHES:INTEGER;VAR INDEX_VECTOR:INDEX_ARR;
                     VAR DOC_RECS:DOC_ARRAY;VAR QUERY_OUT:TEXT);
VAR REQUEST,OPTION:CHAR;
    NUM,I,N, LAST, ITEMS:INTEGER;
BEGIN
{      PRINT OUT RESULTS OF QUERY                             }
{}

WRITELN;
WRITELN;
REWRITE(QUERY_OUT);
WRITELN('*****');
WRITELN('*');
WRITELN('          DISPLAY MENU          *');
WRITELN('*');
WRITELN('*   OPTION          FUNCTION   *');
WRITELN('*   _____          _____ *');
WRITELN('*');
WRITELN('*  1.....DISPLAY TO SCREEN *');
WRITELN('*');
WRITELN('*  2.....PRINT TO OUTPUT FILE*');
WRITELN('*');
WRITELN('*  3.....BOTH (1) AND (2) *');
WRITELN('*');
WRITELN('*****');
WRITE('ENTER OPTION NUMBER ');
READLN(REQUEST);
PAGE(OUTPUT);
WRITELN;
WRITELN;
IF (REQUEST = '1') THEN
BEGIN
WRITELN('*****');
WRITELN('*');
WRITELN('*          DOCUMENT SELECTION MENU          *');
WRITELN('*');
WRITELN('*   OPTION          FUNCTION   *');
WRITELN('*   _____          _____ *');
WRITELN('*');
WRITELN('*  1.....SELECT ALL DOCUMENTS *');
WRITELN('*');
WRITELN('*  2.....SPECIFY NUMBER TO DISPLAY*');
WRITELN('*');
WRITELN('*****');
READLN(OPTION);
PAGE(OUTPUT);
IF (OPTION = '1') THEN
BEGIN
WRITELN;
WRITELN;
WRITELN('RANKINGS IN DECREASING ORDER OF QUERY/DOCUMENT SIMILARAITY');

```

WRITELN;

```
FOR NUM:= 1 TO MATCHES DO
  BEGIN
    I:= INDEX_VECTOR[NUM];
    WITH DOC_RECS[I] DO
      BEGIN
        WRITELN(DOCNO);
        WRITELN;
        FOR N:= 1 TO COUNT-1 DO
          WRITELN(TITLE[N]);
        WRITELN;
        FOR N:= 1 TO ROWS-1 DO
          WRITELN(AUTHOR[N]);
        WRITELN;
        FOR N:= 1 TO LINES-1 DO
          WRITELN(ABSTRACT[N]);
        WRITELN;
        WRITELN;
      END;
    END;
  END
ELSE
  BEGIN
    WRITELN('ENTER NUMBER OF DOCUMENTS TO DISPLAY');
    READLN(ITEMS);
    WRITELN;
    WRITELN;
    FOR NUM:= 1 TO ITEMS DO
      BEGIN
        I:= INDEX_VECTOR[NUM];
        WITH DOC_RECS[I] DO
          BEGIN
            WRITELN(DOCNO);
            WRITELN;
            FOR N:= 1 TO COUNT-1 DO
              WRITELN(TITLE[N]);
            WRITELN;
            FOR N:= 1 TO ROWS-1 DO
              WRITELN(AUTHOR[N]);
            WRITELN;
            FOR N:= 1 TO LINES-1 DO
              WRITELN(ABSTRACT[N]);
            WRITELN;
            WRITELN;
          END;
        END;
      END;
    END;
  END
ELSE
  IF (REQUEST = '2') THEN
    BEGIN
      WRITELN('*****');
      WRITELN('*                                     *');
      WRITELN('*          DOCUMENT SELECTION MENU          *');
      WRITELN('*                                     *');
      WRITELN('*          OPTION          FUNCTION          *');
      WRITELN('*          _____          _____          *');
      WRITELN('*                                     *');
      WRITELN('*          1.....SELECT ALL DOCUMENTS          *');
      WRITELN('*                                     *');
```

```

WRITELN(' * 2.....SPECIFY NUMBER TO DISPLAY*');
WRITELN(' * *');
WRITELN('*****');
READLN(OPTION);
PAGE(OUTPUT);
IF (OPTION = '1') THEN
  BEGIN
WRITELN(QUERY_OUT, 'RANKINGS IN DECREASING QUERY/DOCUMENT SIMILARITY');
WRITELN(QUERY_OUT);
  FOR NUM:= 1 TO MATCHES DO
    BEGIN
      I:= INDEX_VECTOR[NUM];
      WITH DOC_RECS[I] DO
        BEGIN
          WRITELN(QUERY_OUT, DOCNO);
          WRITELN(QUERY_OUT);
          FOR N:= 1 TO COUNT-1 DO
            WRITELN(QUERY_OUT, TITLE[N]);
            WRITELN(QUERY_OUT);
          FOR N:= 1 TO ROWS-1 DO
            WRITELN(QUERY_OUT, AUTHOR[N]);
            WRITELN(QUERY_OUT);
          FOR N:= 1 TO LINES-1 DO
            WRITELN(QUERY_OUT, ABSTRACT[N]);
            WRITELN(QUERY_OUT);
            WRITELN(QUERY_OUT);
          END;
        END;
      END;
    END
  ELSE
  BEGIN
WRITELN('ENTER NUMBER OF DOCUMENTS TO DISPLAY');
READLN(ITEMS);
  FOR NUM:= 1 TO ITEMS DO
    BEGIN
      I:= INDEX_VECTOR[NUM];
      WITH DOC_RECS[I] DO
        BEGIN
          WRITELN(QUERY_OUT, DOCNO);
          WRITELN(QUERY_OUT);
          FOR N:= 1 TO COUNT-1 DO
            WRITELN(QUERY_OUT, TITLE[N]);
            WRITELN(QUERY_OUT);
          FOR N:= 1 TO ROWS-1 DO
            WRITELN(QUERY_OUT, AUTHOR[N]);
            WRITELN(QUERY_OUT);
          FOR N:= 1 TO LINES-1 DO
            WRITELN(QUERY_OUT, ABSTRACT[N]);
            WRITELN(QUERY_OUT);
            WRITELN(QUERY_OUT);
          END;
        END;
      END;
    END;
  END
  ELSE
  IF (REQUEST = '3') THEN
  BEGIN
WRITELN('*****');
WRITELN(' * *');
WRITELN(' * DOCUMENT SELECTION MENU *');

```

```

BEGIN
  Writeln(DOCNO);
  Writeln;
  Writeln(QUERY_OUT, DOCNO);
  Writeln(QUERY_OUT);
  FOR N:= 1 TO COUNT-1 DO
    BEGIN
      Writeln(TITLE[N]);
      Writeln(QUERY_OUT, TITLE[N]);
    END;
  Writeln;
  Writeln(QUERY_OUT);
  FOR N:= 1 TO ROWS-1 DO
    BEGIN
      Writeln(AUTHOR[N]);
      Writeln(QUERY_OUT, AUTHOR[N]);
    END;
  Writeln;
  Writeln(QUERY_OUT);
  FOR N:= 1 TO LINES-1 DO
    BEGIN
      Writeln(ABSTRACT[N]);
      Writeln(QUERY_OUT, ABSTRACT[N]);
    END;
  Writeln;
  Writeln;
  Writeln(QUERY_OUT);
  Writeln(QUERY_OUT);
END;
END;
END;
END;
{}
BEGIN
  WITH CTR_REC DO
    BEGIN
      {}
      {}
      INITIALIZE FREQCTR AND INDEX_VECTOR ARRAYS
      {}
      REWRITE(FREQCTR_FILE);
      Writeln(FREQCTR_FILE, 'TERM FREQUENCY DATA');
      FOR I:= 1 TO DOCCTR DO
        Writeln(FREQCTR_FILE);
        Writeln(FREQCTR_FILE);
        Writeln(FREQCTR_FILE);
        REWRITE(INDEX_FILE);
        Writeln(INDEX_FILE, 'INDEX VECTOR');
        FOR I:= 1 TO DOCCTR DO
          Writeln(INDEX_VECTOR[I]:=I);
        Writeln;
      {}
      {}
      Writeln('ENTER UP TO 20 KEYWORDS AT THE PROMPT >');
      Writeln('ENTER (Q) TO QUIT PROCESSING KEYWORDS');
      Writeln;
      WRITE('> ');
      NUM:=1;
      READLN(Q_TERMS[ NUM ]);
      Writeln;

```

```

WHILE (Q_TERMS[NUM] <> 'Q') DO
  BEGIN
    WRITE ('> ');
    NUM:=NUM+1;
    READLN(Q_TERMS[NUM]);
    WRITELN;
  END;
{}
MATCHES:=0;
FOR N:= 1 TO NUM-1 DO
  BEGIN
    1: J:=1;
    WHILE (Q_TERMS[N] <> ABSTRACT_TERMS[J]) AND (J < CTR) DO
      J:=J+1;
    {}
    IF (Q_TERMS[N] = ABSTRACT_TERMS[J]) THEN
      BEGIN
        I:=J;
        FOR K:= 1 TO DOCCTR DO
          BEGIN
            IF (DOC_TERM[I,K] = '1') THEN
              FREQCTR[K]:=FREQCTR[K]+1;
            END;
          END
        ELSE
          BEGIN
            WRITELN('-----ERROR, NO MATCH FOUND FOR KEYWORD ',N:2);
            WRITELN('CHECK SPELLING AND REENTER KEYWORD ');
            WRITE ('> ');
            READLN(Q_TERMS[N]);
            WRITELN;
            GOTO 1;
          END;
        END;
      END;
    FOR I:= 1 TO DOCCTR DO
      BEGIN
        IF (FREQCTR[I] <> 0) THEN
          MATCHES:=MATCHES+1;
        END;
      WRITELN(FREQCTR_FILE,'QUERY TERMS');
      WRITELN(FREQCTR_FILE);
      FOR N:= 1 TO NUM-1 DO
        WRITELN(FREQCTR_FILE,Q_TERMS[N]);
      WRITELN(FREQCTR_FILE);
      WRITELN(FREQCTR_FILE);
    {}
    WRITELN(FREQCTR_FILE,'DOCUMENT-QUERY MATCHES = ',MATCHES:3);
    WRITELN(FREQCTR_FILE);
    WRITELN(FREQCTR_FILE);
    FOR I:= 1 TO DOCCTR DO
      WRITELN(FREQCTR_FILE,'TERM FREQUENCY FOR DOCUMENT ',I:3,'=',FREQCTR[I]);
      WRITELN(FREQCTR_FILE);
      WRITELN(FREQCTR_FILE);
    {}
    WRITELN;
  { BUBBLE SORT AND RANK THE MOST RELEVANT DOCUMENTS IN DESCENDING ORDER }
  {}
  FOR LAST:=DOCCTR DOWNT0 2 DO
    BEGIN

```

```

FOR CURRENT:=1 TO LAST-1 DO
  BEGIN
    IF (FREQCTR[CURRENT] < FREQCTR[CURRENT+1]) THEN
      BEGIN
{}
      SWAP FREQCTR ELEMENTS
      TEMP:=FREQCTR[CURRENT];
      FREQCTR[CURRENT]:=FREQCTR[CURRENT+1];
      FREQCTR[CURRENT+1]:=TEMP;
{}
      SWAP CORRESPONDING INDEX VECTOR ELEMENTS
      TEMP:=INDEX_VECTOR[CURRENT];
      INDEX_VECTOR[CURRENT]:=INDEX_VECTOR[CURRENT+1];
      INDEX_VECTOR[CURRENT+1]:=TEMP;
      END;
    END;
  END;
END;
WRITELN;
WRITELN(FREQCTR_FILE);
WRITELN('DOCUMENT-QUERY MATCHES = ',MATCHES:3);
WRITELN(FREQCTR_FILE,'SORTED TERM FREQUENCIES AND ASSOCIATED INDEXES');
WRITELN('SORTED TERM FREQUENCIES AND ASSOCIATED INDEXES');
WRITELN;
WRITELN;
WRITELN(FREQCTR_FILE);
WITH CTR_REC DO
  BEGIN
    I:=1;
    WRITELN;
    WRITELN('VIEW RANKED DOCUMENTS? Y/N');
    READLN(REQUEST);
    WHILE (I < CTR) AND (REQUEST = 'Y') DO
      BEGIN
        WRITELN('SEQUENCE DOC TERM');
        WRITELN('NO. NO. FREQ. ');
        WRITELN(' _____ ');
        WRITELN;
        FOR COUNT:= I TO I+14 DO
          BEGIN
            WRITE(FREQCTR_FILE,COUNT:3);
            WRITE(COUNT:3);
            WRITE(FREQCTR_FILE,' ');
            WRITE(' ');
            WRITE(FREQCTR_FILE,INDEX_VECTOR[COUNT]:3);
            WRITE(INDEX_VECTOR[COUNT]:3);
            WRITE(FREQCTR_FILE,' ');
            WRITE(' ');
            WRITELN(FREQCTR_FILE,FREQCTR[COUNT]:3);
            WRITELN(FREQCTR[COUNT]:3);
          END;
        WRITELN;
        WRITELN('CONTINUE VIEWING? Y/N');
        READLN(REQUEST);
        I:=COUNT+1;
      END;
    END;
  PRINT_DOCS(MATCHES,INDEX_VECTOR,DOC_RECS,QUERY_OUT);
END;
{}

```

```

BEGIN { MAIN PROCEDURE}
{}
OPEN(ABSTRACT_FILE,HISTORY:=OLD);
OPEN(AUX_FILE,HISTORY:=OLD);
OPEN(CTRS_DAT,HISTORY:=OLD);
OPEN(BINARY_FILE,HISTORY:=OLD);
OPEN(UPDATE_DAT,HISTORY:=OLD);
OPEN(QUERY_OUT,HISTORY:=OLD);
OPEN(TEMP_FILE,HISTORY:=OLD);
OPEN(INDEX_FILE,HISTORY:=OLD);
OPEN(FREQCTR_FILE,HISTORY:=OLD);
{}
WRITELN('*****');
WRITELN(' *');
WRITELN(' *                WELCOME TO RETRIEVE  1.0                *');
WRITELN(' *');
WRITELN(' *                SEARCH AID SOFTWARE                *');
WRITELN(' *');
WRITELN(' *                DEVELOPED BY                *');
WRITELN(' *');
WRITELN(' *                KAY F. IRBY                *');
WRITELN(' *');
WRITELN(' *                GRADUATE RESEARCH ASSISTANT        *');
WRITELN(' *                THE UNIVERSITY OF TENNESSEE        *');
WRITELN(' *                SPACE INSTITUTE                *');
WRITELN(' *                REMOTE SENSING DIVISION            *');
WRITELN(' *');
WRITELN('*****');
WRITELN('HIT C TO CONTINUE');
READLN(REQUEST);
PAGE(OUTPUT);
REPEAT
WRITELN('*****');
WRITELN(' *');
WRITELN(' *                COMMAND MENU                *');
WRITELN(' *');
WRITELN(' * SELECTION                FUNCTION                *');
WRITELN(' * _____                _____                *');
WRITELN(' * 1..... INITIALIZATION                *');
WRITELN(' * 2..... UPDATE DATABASE                *');
WRITELN(' * 3..... PROCESS KEYWORDS                *');
WRITELN(' * 4..... BUILD INVERTED FILE                *');
WRITELN(' * 5..... PROCESS USER QUERY                *');
WRITELN(' *');
WRITELN('*****');
WRITELN('ENTER SELECTION');
READLN(REQUEST);
IF (REQUEST = '1') THEN
BEGIN
PAGE(OUTPUT);
INITLIZE(AUX_DOCTERM,ABSTRACT_TERMS,DOC_TERM,CTR_REC,AUX_FILE,
ABSTRACT_FILE,CTRS_DAT,BINARY_FILE,UPDATE_DAT,KEYWORDS);
END
ELSE

```

```

IF (REQUEST = '2') THEN
  BEGIN
    PAGE (OUTPUT);
    UPDATE (CTR_REC, DOC_RECS, UPDATE_DAT, CTRS_DAT);
  END
ELSE
IF (REQUEST = '3') THEN
  BEGIN
    PAGE (OUTPUT);
    PROCESS_KEYWORDS (CTR_REC, AUX_DOCTERM, ABSTRACT_TERMS, KEYWORDS, CTRS_DAT,
                      AUX_FILE, ABSTRACT_FILE);
  END
ELSE
IF (REQUEST = '4') THEN
  BEGIN
    PAGE (OUTPUT);
    PROCESS_SEARCH_ARRAY (AUX_DOCTERM, ABSTRACT_TERMS, DOC_TERM, CTR_REC,
                          AUX_FILE, ABSTRACT_FILE, CTRS_DAT, BINARY_FILE);
  END
ELSE
IF (REQUEST = '5') THEN
  BEGIN
    PAGE (OUTPUT);
    RESET (CTRS_DAT);
    RESET (BINARY_FILE);
    RESET (ABSTRACT_FILE);
    RESET (UPDATE_DAT);
    WRITELN (' READING IN FILES');
    WRITELN ('- PLEASE STAND BY -');
    WRITELN;
    WITH CTR_REC DO
    BEGIN
      READLN (CTRS_DAT, DOCNUM);
      READLN (CTRS_DAT, CTR);
      READLN (CTRS_DAT, DOCCTR);
      FOR I:= 1 TO DOCCTR DO
        READLN (CTRS_DAT, INDEX[I]);
    {}
    FOR I:= 1 TO DOCNUM DO
      BEGIN
        WITH DOC_RECS[I] DO
          BEGIN
            READLN (UPDATE_DAT, DOCNO);
            READLN (UPDATE_DAT, COUNT);
            FOR N:= 1 TO COUNT DO
              READLN (UPDATE_DAT, TITLE[N]);
            READLN (UPDATE_DAT, ROWS);
            FOR N:= 1 TO ROWS DO
              READLN (UPDATE_DAT, AUTHOR[N]);
            READLN (UPDATE_DAT, LINES);
            FOR N:= 1 TO LINES DO
              READLN (UPDATE_DAT, ABSTRACT[N]);
            READLN (UPDATE_DAT);
          END;
        END;
      {}
    K:=0;
    FOR I:= 1 TO CTR-1 DO
      BEGIN
        REPEAT

```



```

    READ (BINARY_FILE, CHARACTER);
    IF (CHARACTER <> BLANK) THEN
        BEGIN
            K:=K+1;
            DOC_TERM[I,K]:=CHARACTER;
        END;
    UNTIL EOLN(BINARY_FILE);
    K:=0;
    READLN(BINARY_FILE);
END;
{}
FOR I:= 1 TO CTR-1 DO
    READLN(ABSTRACT_FILE, ABSTRACT_TERMS[I]);
END;
{}
REPEAT
    PROCESS_QUERY(DOC_TERM, BINARY_FILE, CTR_REC, CTRS_DAT, ABSTRACT_TERMS,
        ABSTRACT_FILE, DOC_RECS, UPDATE_DAT, QUERY_OUT, TEMP_FILE,
        INDEX_FILE, FREQCTR_FILE);
    WRITELN('PROCESS ANOTHER QUERY? Y/N');
    READLN(RESPONSE);
    UNTIL (RESPONSE = 'N');
END;
WRITELN('PROCESS ANOTHER MENU SELECTION? Y/N');
READLN(REQUEST);
UNTIL (REQUEST = 'N');
END.

```

APPENDIX C

THESAURUS

UTSI THESES

1972
1973
1976
1979
ABSORBED
ABSTRACTS
ACCURACY
ACQUISITION
AEDC (ARNOLD ENGINEERING DEVELOPMENT CENTER)
AERIAL PHOTOGRAPHS
AEROSPACE
AETL (ARMY ENGINEERING TECHNOLOGY LABS)
AFFORESTATION
ANALYSIS
APPLICATION
ATMOSPHERIC HAZE
AUTOMATIC
BAND 4
BAND 5
BAND 6
BAND 7
BANDS
BIG SOUTH FORK
BLOCK II NAVSTAR
BOTTOMLAND
BOUNDARIES
CHAMBERS CREEK DRAINAGE SYSTEM
CHANGE
CHARACTERISTICS
CLASSIFICATION
COLOR COMPOSITES
COMMERCIAL
COMPARISON
COMPUTER
COMPUTER PROCESSING
COMPUTER TIME
COORDINATES
CORRECTION TECHNIQUES
CURVE-FITTING
DATA
DATA PROCESSING
DATA SET
DATABASE
DENSITOMETER
DENSITOMETRIC
DEPENDENCE
DETECTION
DETERMINING
DIGITAL
DRYLAND
ECOSYSTEMS
ELECTROMAGNETIC
ENGINEERING
ENHANCEMENT
ENTITIES
ENVIRONMENTAL
EQUIDENSITOMETRY
ERTS (EARTH RESOURCES TECHNOLOGY SATELLITE)
ESTIMATION
EVALUATE
EXPLOITING
FEATURE
FEEDBACK
FLOOD
FORMAT
FORT BELVOIR, VIRGINIA
GEOLOGICAL
GIS (GEOGRAPHICAL INFORMATION SYSTEM)
GPS (GLOBAL POSITIONING SYSTEM)
GRAVITY
GRID-BASED
HA'IL, SAUDI ARABIA
HEAT FLUX
HIGH
HIGH-PASS FILTER
HUNTSVILLE, ALABAMA
HYBRID RATIO FALSE COLOR
HYDROLOGY
IBM
IDENTIFICATION
IMAGE
IMAGERY
IMPROVE
INFORMATION
INFRARED
INFRARED PHOTOGRAPHY
INPUT
INQUIRIES
INTERACTION
LAND COVER
LAND USE
LANDSAT
LANDSAT 1
LANDSAT 2
LANDSAT 3
LINEAMENTS
LINEAR AND SINUSOIDAL STRETCHES
LOW
MANAGEMENT
MANAGEMENT INFORMATION
MAPPING
MAPS
MARK I
MEASUREMENTS
METHOD
MICROWAVE
MINIMIZE
MONITORING
MSS (LANDSAT MULTISPECTRAL SCANNER)
MULTI-SPATIAL
MULTI-THEME
MULTISPECTRAL
MULTISPECTRAL VIEWER
MULTIVARIATE
NATURAL RESOURCES
NEW RIVER WATERSHED
NON-PHOTOGRAPHIC
NORMILIZATION
NORTHEAST TENNESSEE
OBION-FORKED DEER RIVER BASIN
OPTICAL DENSITY
PHOTOINTERPRETATION
PHOTOVOLTAIC CELLS
PILOT
POLYGONAL
PRECISION
PROCESS
PROCESSING
PROGRAM
QTV (QUALIFICATION TEST VEHICLE)
QUANTITY
RADIATION

RADIOMETER
RASTER
RBV (RETURN BEAM VIDICON)
RECALL
REDUCTION
REFLECTED
REGIONAL
REGISTRATION
REMOTE SENSING
REMOVE
RETRIEVAL
RIYADH, SAUDI ARABIA
ROSE DIAGRAMS
SAHARA DESERT
SAND DUNES MOVEMENT
SAND FIXATION
SATELLITE
SAVANNAH, TENNESSEE
SCIENTIFIC
SEASONS
SENSORS
SIMULATED NATURAL COLOR
SINGLE-THEME
SOFTWARE
SOIL
SOIL MOISTURE
SOLAR
SOLAR PANELS
SPACE
SPACE PLATFORM
SPACE-ACQUIRED
SPECTRAL
STATISTICAL
STORAGE
STRIP MINES
STUDY
SYSTEM
TECHNIQUES
TECTONIC MODEL
TELEVISION
TEMPORAL ANALYSIS
TEST CHAMBER
TEST SITE
THEMATIC MAPS
THERMAL VACUUM TESTING
TIME
TIME-VARIANT
TOOL
TOPOGRAPHIC
TRANSFORM
TRANSFORMATION
TRENDS
TVA (TENNESSEE VALLEY AUTHORITY)
URBAN
USGS (UNITED STATES GEOLOGICAL SURVEY)
UTM (UNIVERSAL TRANSVERSE MERCATOR)
UTSI (THE UNIVERSITY OF TN SPACE INSTITUTE)
VEGETATION
WATER POLLUTION
WATER RESOURCES
WATERSHED
WEST TENNESSEE
WETLAND
ZOOM TRANSFER SCOPE

EARTH SCIENCE

ACCURACY
ALGORITHM
APPLICATION
ATMOSPHERIC FEATURES
BRIGHTNESS TEMPERATURE MODEL
CHARACTERISTICS
CLIMATE
COMPARISON
DETERMINING
DEVELOPING
DIFFERENCES
ESTIMATION
GCM (GROUND CIRCULATION MODEL)
GEOPHYSICAL PARAMETERS
GHM (GROUND HYDROLOGY MODEL)
GROUND
HYDROLOGY
INPUT
LAND
LAND COVER
MEASUREMENTS
MICROWAVE
MODEL
NDVI (NORMALIZED VEGETATION INDEX)
NIMBUS 7
OBSERVATIONS
OCEAN
OCEANIC RAINRATES
PASSIVE
REGIONAL
REMOTE SENSING
RETRIEVAL
SATELLITE
SEA-ICE CONCENTRATION
SIGNATURES
SMMR (SCANNING MULTICHANNEL MICROWAVE RADIOMETER)
SNOW COVER
SOIL MOISTURE
SURFACE
SURFACE TEMPERATURE
VALIDATING
VEGETATION

RADAR/SONAR

ACCURACY
ACQUISITION
ADVANTAGES
AIRBORNE
ALASKA
ALIGNMENT
ANALYSIS
APPALACHIAN
APPALACHIAN VALLEY
APPLICATION
ARCHAEOLOGICAL
AREA
ARGENTINA
ASCENDING
AVAILABLE
AVERAGE TREE HEIGHT
BACKSCATTER CURVES
BASAL AREA
BOREAL FOREST
CALIFORNIA
CARTOGRAPHY
CHARACTERIZATION
COMMISSION
COMPARISON
COMPUTER-ASSISTED
CONDITION
CONSTRUCTION
CONTINENTAL SLOPE
CORRECTION
CULTURAL
DATA
DECIDUOUS
DEM (DIGITAL ELEVATION MODEL)
DESCENDING
DETECTION
DETERMINING
DIGITAL
DISADVANTAGES
DISCONTINUITIES
DISTRIBUTION
DRAINAGE
ENVIRONMENTAL
ERRORS
ESCARPMENTS
EVALUATION
FAULTS
FEATURES
FILM
FIXED LOOK ANGLE
FLIGHT PATHS
FLOOD
FLOOD PLAINS
FOREST
FREIBURG, WEST GERMANY
GEOGRAPHY
GEOLOGIC
GEOLOGY
GEOMETRIC
GEOMORPHIC
GLORIA (GEOLOGICAL LONG-RANGE INCLINED ASDIC)
GRANT RIVER
GROUND
GULF OF MEXICO
HAZARDOUS WASTE DISPOSAL
HIGH-RESOLUTION
HYDROLOGIC
IDENTIFICATION
IMAGE
IMAGE BASED
IMAGE GREY VALUES
IMAGERY
IMAGES
IMPLEMENTATION
INPUT
INTERACTION
INTERPRETATION
INUNDATION
LAND COVER
LANDSLIDE
LIKE-POLARIZED
MANAGEMENT
MAP-TO-IMAGE-CORRESPONDENCE
MAPPING
MAPS
MARINE TERRACES
METHOD
MICROWAVE
MIPS (MINI IMAGE PROCESSING SYSTEM)
MISSISSIPPI RIVER
MORPHOLOGY
MOSAICS
MULTIPOLARIZATION
NATURAL GAS DEPOSITS
NORTHEASTERN
NORTHERN CALIFORNIA
NOVEMBER
OCTOBER
OFF-NADIR
OMMISSION
OPEN SURFACE WATER
OPTICAL
ORBIT
OVERLAYS
PARAMETERS
PATHWAYS
PENETRATING
PLANIMETRIC
POTASI, WISCONSIN
PREDICTION
PROCESSING
PRODUCT
PRODUCTS
PROGRAM
PUBLIC
QUADRANGLES
RADAR
RADARGRAMMETRIC
RADARGRAMMETRY
RADIATION
RADIOMETRIC
RECCONNAISSANCE
REGRESSION
RESOURCE
RESULTS
RIDGE PROVINCE
ROCKSLIDE
SALT
SAR (SYNTHETIC APERTURE RADAR)
SEA-FLOOR
SEDIMENT
SENSOR
SIDESCAN

SIMULATION
SIR (SHUTTLE IMAGING RADAR)
SIR-A (SHUTTLE IMAGING RADAR)
SIR-B (SHUTTLE IMAGING RADAR)
SITE
SLAR (SIDE LOOKING AIRBORNE RADAR)
SMART (RADARGRAMMETRIC SOFTWARE SYSTEM)
SOFTWARE
SOIL
SOIL PROFILES
SONAR
SOUTHEASTERN
SPACE SHUTTLE
STANDS
STRUCTURAL
STRUCTURE
STUDY
SUBSURFACE
SURFICIAL
SYNOPTIC VIEW
SYSTEM
TECHNIQUES
TENNESSEE
TOPOGRAPHIC
TOPOGRAPHY
TOTAL-TREE BIOMASS
TRANSPORT
TRANSPORTATION
U.S. (UNITED STATES)
UPPER BENUE TROUGH, NIGERIA
URBAN DEVELOPMENT
USE
USGS (UNITED STATES GEOLOGICAL SURVEY)
WATER TABLE
X-BAND

LAND COVER/LAND USE

1976
1977
1981
1982
1984
1985
ACQUIRED
AERIAL PHOTOGRAPHS
AFFECT
AGRICULTURAL
ANALYSIS
ANOMALY
APPLICATION
ARABIAN HIGHLANDS
AREA
AUGUST
AUTOMATED
BACKSCATTERING COEFFICIENTS
BAND 3
BAND 4
BAND 5
BAND 6
BIOMASS
BIOPHYSICAL
CALCULATIONS
CHANGE
CHARACTERISTICS
CLASSIFICATION
CLEMSON UNIVERSITY
COASTAL
COLOR COMPOSITES
COMPARISON
COMPUTER-COMPATIBLE TAPES
CONNECTICUT
CROP
DATA
DATA SETS
DECEMBER
DENSITY SLICING
DEPARTMENT OF FORESTRY
DESERT
DETECTION
DETERMINING
DEVELOPMENT
DIGITAL
DIRECTION
DISTRIBUTION
EDGE
ENHANCEMENT
ENIGMA
ENVIRONMENT
ENVIRONMENTAL
ESTIMATING
EVALUATE
EXTRACTION
FEATURES
FISH
FLOOD
FOREST
FORESTLAND
FORMAT
GEOBOTANICAL
GEOLOGIC
GIS (GEOGRAPHICAL INFORMATI
GROUND WATER
GROUND-BASED
GROWTH
HABITAT
HAZARD
HERBACEOUS
HEURISTIC
HONDURAS
HURRICANE
HURRICANE FIFI
HYDROLOGIC
IDENTIFICATION
IMAGE RATIOING
IMAGERY
IMPACT
IMPERIAL VALLEY, CALIFORNIA
INFORMATION
INTERPRETATION
INTRACLASS
INVENTORY
JULY
L-BAND
LAND COVER
LAND USE
LANDSAT
LANDSAT 4
LEVEL
LIKENESS
LINE
LINEAR
LOCAL
MAGNITUDE
MAIN PASS
MAPPING
MAPS
MARSH
MEASUREMENTS
MERGING
METHOD
MISSISSIPPI DELTA
MISSOURI
MODEL
MONITOR
MONITORING
MOWD (MINISTRY OF WATER DEVELOPMENT)
MSS (LANDSAT MULTISPECTRAL SCANNER)
NAIROBI, KENYA
NETWORK
NEW YORK STATE
NORTHEASTERN
NOVEMBER
OBSERVATIONS
OCTOBER
OKLAHOMA
OPTICAL REFLECTANCE-SPECTRA
OXYGEN
PHOTO-OPTICAL
PHOTOGRAPHIC
PHOTOINTERPRETATION
PLAIN
PLANTS
POLYGONAL
PREDICT
PRINCIPAL COMPONENT ANALYSIS
QUANTITATIVE
QUANTITY
RADIOMETER
RECONNAISSANCE

REFLECTANCE
REGRESSION
REMOTE SENSING
RICHARD B. RUSSELL LAKE
ROAD
RRSF (REGIONAL REMOTE SENSING FACILITY)
SAMPLING
SATELLITE
SEARCH
SEQUENTIAL
SHRUBS
SIGNATURE
SIGNATURES
SILVER MINE
SIR-B
SITE
SOUTH CAROLINA
SOUTHEASTERN
SPECTRAL
SPLAY SETS
STATISTICS
STUDY
SURFACE
TECHNIQUES
TEXTURE
THEMATIC MAPPER
THERMAL
TIDAL
TIME-VARIANT
TOPOGRAPHIC
TRANSFORMED DIVERGENCE
TREE
TREES
URBAN
USE
VARIABILITY
VARIABLES
VEGETATION
VISIBLE
VISUAL
VOLCANO
WATER
WETLANDS
WOODY

PLANT SCIENCE

1983
1985
35MM
70MM
AERIAL
AERIAL PHOTOGRAPHS
AGRICULTURAL
AGROMETEOROLOGICAL
AIR TEMPERATURE
ANALYSIS
ASSESSMENT
BANDS
BIOLOGICAL
CALCULATOR
CAMERA
CANOPY
CANOPY DENSITY
CHARACTERISTICS
CITRUS TREE
CLASSIFICATION
COLOR
COLOR SLIDES
COMPARISON
COMPUTER
CORRELATIONS
COTTON
DAMAGE
DATA
DEER
DENSITY
DETERMINING
DEVELOPING
DIFFERENCES
DIGITAL
DISTANCE-TO-EDGE
DISTRIBUTION
DIURNAL
DROUGHT-STRESSED
ECOSYSTEMS
EFFECT
EFFECTS
ELK
ENERGY BUDGET
ESTIMATION
EVALUATION
EVAPOTRANSPIRATIONAL
FALL
FEATURES
FIELD
FILM
FILTER
FOREST
FORESTLAND
FREQUENCY
FUNCTIONAL
GIS (GEOGRAPHICAL INFORMATION SYSTEM)
GRASS (GEOGRAPHICAL RESOURCE ANALYSIS SUPPORT SYSTEM)
GROUND-BASED
HABITAT MANAGEMENT
HISTOGRAM
IMAGE
IMAGE PROCESSING
IMAGERY
IN SITU
INDEX
INFRARED
INTERCEPTION
INTERPRETATION
INTERRELATIONS
INVENTORY
IRRIGATED
LANDSAT
LEAF
LEAF AREA INDEX
LEAF CHLOROPHYLL
LEAF WATER CONTENT
LIGHT
LOCATION
MAGNETIC RESONANCE IMAGING
MANAGEMENT DECISIONS
MATHEMATICAL
MEASUREMENTS
MEDICAL
METHOD
MICROSCOPE
MID-INFRARED
MIXED CONIFER FOREST
MOBILE
MODEL
MULTIBAND
MULTISPECTRAL
NEAR-INFRARED
NONPUBESCENT
NONSTRESSED
NUCLEAR
OREGON
PAR (PHOTOSYNTHETICALLY ACTIVE RADIATION)
PATTERNS
PER MU YIELD
PHENOLOGICAL
PHOTOGRAPHY
PHOTOINTERPRETATION
PHYLLOXERA
PLANT
PREDICT
PRINCIPAL COMPONENT ANALYSIS
PRODUCTIVITY
PROGRAM
PROPERTIES
PUBESCENCE
PVI (PERPENDICULAR VEGETATION INDEX)
RADIANCE VECTORS
RADIO
RADIOMETER
RANGELAND
RECORDINGS
REFLECTANCE
REGRESSION ANALYSIS
RELATIONSHIPS
REMOTE SENSING
REMOVE
ROOTS
RUNOFF
SALINAS VALLEY, CALIFORNIA
SATELLITE
SEASONAL
SELECTION
SENSITIVE
SHRUBS
SIERRA NEVADA
SITE
SOIL

SOLAR
SORGHUM
SOUTHERN ILLINOIS
SOYBEANS
SPATIAL
SPECTRAL
SPECTRAL CLASSES
SPECTRORADIOMETER
SPECTRORADIOMETRIC
STATIC
STATISTICAL
STEM COUNTS
STRESS
STRESSED
STRUCTURAL
STUDY
SUGARCANE
SUN RADIANCE FACTORS
SYSTEM
TECHNIQUES
TEMPERATURES
TEST SITE
TEXTURAL
THEMATIC MAPPER
THEMATIC MAPPER SIMULATOR
THERMAL
TOOL
TOTAL YIELD
TREE
USE
VEGETATION
VIDEO
VINEYARDS
VISIBLE
WARM SPRINGS INDIAN RESERVATION
WATER
WATER ABSORPTION REGION
WATER BALANCE
WATER LOSSES
WHEAT
WILDLIFE MANAGEMENT
WOOLLY STEMEDIA
YIELD
YOUNG PINE PLANTATIONS
ZENITH ANGLE

PHOTOGRAMMETRY

1984
ACCURACY
ACQUISITION
ADAPTIVE SPACING GRID
AERIAL TRIANGULATION
AGRICULTURAL
ALBERTA, CANADA
ALGORITHM
ANALYSIS
ANALYTICAL PLOTTER
APPLICATION
AREA
ASSESSMENT
AVHRR (ADVANCED VERY HIGH RESOLUTION RADIOMETER)
BAND 5
BASINS
BEHAVIOR
BIRDS
BIRDSVILLE COLONY
BREAKLINES
CALCULATIONS
CAPABILITIES
CAPIR/DEM (CAPIR/DIGITAL ELEVATION MODEL)
CENSUS
CLASSIFICATION
CLUSTERING
COLLECTION
COMPUTATIONAL
COMPUTER
CONDITIONS
CONTOURS
CORN
COTTON
COVER
CROP
CURVES
DATA
DATABASE
DEM (DIGITAL ELEVATION MODEL)
DETERMINISTIC
DIFFERENCE VARIANCE
DIGITAL
DISTRIBUTION
ELEVATION
ENDANGERED SPECIES
EVALUATION
FORAGING
FRACTAL
GEOGRAPHICAL
GEORGIA
GOVERNMENT
GRID
GSFC (GODDARD SPACE FLIGHT CENTER)
HABITAT
HIGH ATLAS MOUNTAINS
IDENTIFICATION
IMAGE
INDEX
INTERACTIVE
INTERPOLATION
LAND USE
LANDSAT
LAS (LAND ANALYSIS SYSTEM)
MACROSCOPIC
MAPPING
MAPS
MATCHING
MEASUREMENTS
MEXICO
MICROSCOPIC
MIGRATORY
MODEL
MONITORING
MOROCCO
MSS (LANDSAT MULTISPECTRAL SCANNER)
OBSERVATIONS
OPERATIONS
OPTIMAL
OPTIMIZE
ORTHO-PHOTO
PARAMETERS
PHOTOGRAMMETRIC
POINT TRANSFER
POLYSITE
PRINCIPAL COMPONENT ANALYSIS
PROCEDURES
PROFILE
PROGRAM
REFINEMENT
REFLECTANCE
REGIONAL
RELATIONSHIP
REMOTE SENSING
RESOURCES
RESTRUCTURE
ROBOTICS
SAMPLE
SATELLITE
SCOP (INTERPOLATION PROGRAM)
SEASONAL TOTAL PRECIPITATION
SELECTION
SIMULATION
SITE
SNOW COVER
SOFTWARE
SOUTH CAROLINA
SOUTHEASTERN
SOYBEANS
SPACING
SPATIAL
SPECTRAL
STANDARD
STATISTICAL
STOCHASTIC
STUDY
SUPERVISED
SURFACE
SYSTEM
TECHNIQUES
TERRAIN
THEMATIC MAPPER
THEMATIC MAPS
TOBACCO
TOPOGRAPHY
TRAINING
TROPICAL FOREST
U.S. (UNITED STATES)
UNSUPERVISED
URBAN
VARIABLES
VARIOGRAM
VEGETATION
WINDOW
WOOD STORK
YUCATAN PENINSULA

HYDROSPHERIC SCIENCE

1980
1983
1985
AERIAL PHOTOGRAPHS
AIRCRAFT
ANOVA
AQUATIC
ARRAY
BANDS
CHESAPEAKE BAY
CHLOROPHYLL
COLOR
COLOR-CODING
COMPARISON
CONCENTRATION
CONTOURING
COOLING POND
COST
DATA
DELTA, CALIFORNIA
DETERMINING
DEVELOPMENT
DIFFERENCES
DIGITAL
DISPERSION
DISTRIBUTION
DYE
EMERGENT
ESTIMATING
ESTUARINE
FEASIBILITY
GENERALIZED
GENERATION
GEOGRAPHIC
HARVEST
HEALTHY
IDENTIFICATION
IMAGE PROCESSING
INFRARED
ISOTHERMS
JANUARY
JUNE
LAKES
LANDSAT
LANDSAT 2
LANDSAT 4
LOCATIONS
MACROPHYTE
MAPPING
MAPS
MISSISSIPPI
MODEL
MONITORING
MOON LAKE
MSS (LANDSAT MULTISPECTRAL SCANNER)
MULTIDATE
NORTH CAROLINA
OYSTER BEDS
PARAMETERS
PIXEL
PREDICT
PROCESSING
RATIOING
REFLECTANCE
REMOTE SENSING
RESERVOIRS
RHODAMINE WT
SALINITY CONCENTRATION
SAN FRANCISCO BAY
SATELLITE
SEEDING
SELECT
SIZE
SOUTH CAROLINA
SPECTRAL
STATISTICAL
STUDY
SUBMERGENT
SURFACE
T-TEST
TECHNIQUES
THEMATIC MAPPER
THERMAL
TOTAL SUSPENDED SOLIDS
TURBIDITY
UTM (UNIVERSAL TRANSVERSE MERCATOR)
VALUES
VARIABLES
VEGETATION
WATER QUALITY

WORKSTATION

32-BIT
ANALYSIS
APPLICATION
AUTOMATED
CARTOGRAPHIC MODEL
COLOR GRAPHICS
COMPUTER
COMPUTER PROCESSING
CONSIDERATIONS
CONVERSION
CRT
DATA
DESIGN
DEVELOPMENT
ELAS (EARTH RESOURCES LAB APPLICATIONS SOFTWARE)
ENVIRONMENTAL
FEATURES
FIXED DISK
FLOATING POINT ACCELERATOR
GEOGRAPHIC
GIS
GRASS (GEOGRAPHICAL RESOURCES ANALYSIS SUPPORT SYSTEM)
HARDWARE
IBM
IDENTIFICATION
IMAGE PROCESSING
IMAGERY
IMPLEMENTATION
INDEPENDENT
INPUT
INTEGRATION
INTERFACE
LAS (LAND ANALYSIS SYSTEM)
MADISON, WISCONSIN
MAP
MAP RECLASSIFICATION
MASSCOMP
MATHEMATICAL STRUCTURE
MICROCOMPUTER
MICROPROCESSOR-BASED
MODULAR
NATURAL LANGUAGE
OVERLAYING
PHOTOINTERPRETATION
PMAP (PROFESSIONAL MAP ANALYSIS PACKAGE)
PRIMITIVE OPERATORS
REMOTE SENSING
SET
SOFTWARE
SPATIAL INTERPOLATION
STEREOSCOPIC
SYSTEM
TECHNIQUES
TERRAIN
TRANSPORTABLE
UNIVERSITY OF WISCONSIN
UNIX
UNIX-BASED
WORKSTATION

IMAGE PROCESSING

AERIAL PHOTOGRAPHS
ALGORITHM
ALTERNATIVE
APPLICATION
AREA
AUTOMATED
CANADA
CHANGE
CLASSICAL
CLASSIFICATION
COMPARISON
COMPUTER
CONTENT
CONVERSION
CORRELATION
DATA
DATA FLOW
DATA STRUCTURES
DENSITY SLICING
DESCRIPTORS
DETECTION
DIFFERENCE
DIGITAL
DISSIMILAR
EARTH
EDGE
EDGE ENHANCEMENT
ENHANCEMENT
FASTER
FEATURES
FILTERING
FORESTLAND
FOURIER TRANSFORM
FREQUENCY
FUNCTIONS
GIS (GEOGRAPHICAL INFORMATION SYSTEM)
GRAY VALUES
HIGH-FREQUENCY
IDENTIFICATION
IMAGE
IMAGE PROCESSING
IMAGERY
ISSUES
KNOWLEDGE BASED
LAND COVER
LANDSAT
LARGE-SCALE
LINE-ORIENTED
LINEAR
MANUAL
MARR-HILDRETH
MATCHING
METHOD
MICROCOMPUTER
MSS (LANDSAT MULTISPECTRAL SCANNER)
NOISE
OIL/GAS WELLS
PERFORMANCE
PIXEL
PLANIMETRIC
PRINCIPAL COMPONENT ANALYSIS
PROCEDURES
PROCESSING
RADAR
RASTER
RECOGNITION
REDUCTION
REMOTE SENSING
REMOVE
RESEARCH
RESOLUTION
RESOURCES
RINGING
SAR (SYNTHETIC APERTURE RADAR)
SATELLITE
SHAPES
SOFTWARE
SPATIAL
SPECTRAL
STRATEGIES
STRUCTURE
SYSTEM
TASSELED CAP TRANSFORMATION
TECHNIQUES
THEMATIC MAPPER
THERMAL
UNIVERSITY OF WATERLOO
UNIVERSITY OF WISCONSIN-MADISON
USE
VECTOR
VISION
ZERO CROSSINGS

APPENDIX D
EXAMPLES OF PROGRAM EXECUTION

MAIN MENU

OPTION

SELECTION

1.....UPDATE DATABASE

2.....INITIALIZE CTRS

3.....PROCESS KEYWORDS

4.....BUILD INVERTED FILE

5.....PROCESS QUERY

PROCEDURE PROCESS_KEYWORDS

enter number of documents to process
> 2 <return>

> enter number of keywords for document 1
> 5 <return>

enter term no. 1: aerodynamic <return>
enter term no. 2: flow <return>
enter term no. 3: drag <return>
enter term no. 4: compressible flow <return>
enter term no. 5: aircraft <return>

enter number of keywords for document 2
> 6 <return>

enter term no. 1: remote sensing <return>
enter term no. 2: Landsat <return>
enter term no. 3: aerial photography <return>
enter term no. 4: photointerpretation <return>
enter term no. 5: mapping <return>
enter term no. 6: analysis <return>

Total number of documents processed: 2
Total number of unique terms: 11

PROCEDURE UPDATE

enter number of documents to process

> 1 <return>

enter document number

> 01 <return>

enter title , type '/' when finished

Optimum Sampling Spacing in Digital Elevation Models

/ <return>

enter author(s), type '/' when finished

Gregory James Smith <return>

Hampton Institute of Technology <return>

1934 Neil Avenue <return>

Hampton, Virginia <return>

/ <return>

enter abstract--do not exceed 14 lines of text, hit
'/' when finished

The paper documents a study specifying the necessary methodology and tools for the creation of a digital elevation database. In particular, it documents the required procedures and densities for data capture. The choice of the optimal sampling spacing, meeting pre-given accuracy specifications for the digital elevation model, is based on statistical analysis of selected terrain profiles in the projected area.

/ <return>

Number of documents Processed: 01

Total number of documents in collection: 01

PROCEDURE PROCESS QUERY

enter up to 20 query terms, type 'q' to quit

>study
>air temperature
>analysis
>citrus tree
>vineyards
>yield
q

Document-Query Matches = 36

View Ranked Documents? Y/N
>Y

Sorted Term Frequencies and Associated Indexes

sequence no.	document no.	term frequency
1	58	3
2	4	2
3	5	2
4	6	2
5	28	2
6	2	1
7	3	1
8	9	1
9	11	1
10	12	1

Continue Viewing? Y/N
>N

DISPLAY MENU	
<u>OPTION</u>	<u>FUNCTION</u>
1.....	DISPLAY TO SCREEN
2.....	WRITE TO DISK
3.....	BOTH(1) AND (2)

DOCUMENT SELECTION MENU	
<u>OPTION</u>	<u>FUNCTION</u>
1.....	SELECT ALL DOCS
2.....	SPECIFY NUMBER

TERM FREQUENCY DATA

QUERY TERMS

APPLICATION
UNIX-BASED
WORKSTATION
COLOR GRAPHICS
DESIGN
DEVELOPMENT
IMAGE PROCESSING
INTEGRATION

DOCUMENT-QUERY MATCHES = 22

SORTED TERM FREQUENCIES AND ASSOCIATED INDEXES

1	93	4
2	92	3
3	94	3
4	1	1
5	3	1
6	5	1
7	8	1
8	12	1
9	14	1
10	20	1
11	24	1
12	25	1
13	30	1
14	31	1
15	42	1

Development of a 32-Bit Unix-Based ELAS Workstation

Thomas D. Cheng
 955 Westmoreland Road, #206
 Colorado Springs, Colorado 80907
 phone: (303) 531-6348

Bruce A. Splering, Ronnie W. Pearson
 NASA/NSTL Earth Resources Lab
 NSTL, Mississippi 39529
 (601) 688-3588 or 688-1934

A mini/micorcomputer Unix-based image analysis workstation has been designed and is being implemented to use the Earth Resources Laboratory Applications Software (ELAS). The hardware system includes a MASSCOMP 5600 computer, which is a 32-bit Unix-based system (compatible with AT&T System V and Berkely 4.2 BSD operating system), a floating point accelerator, a 474-megabyte fixed disk, a tri-density magnetic tape drive, and an 1152 by 910 by 12-plane color graphics/image interface. The software conversion includes reconfiguring the ELAS driver Master Task, recompiling and then testing the converted application modules. This hardware and software configuration is a self-sufficient image analysis workstation which can be used as a stand-alone system, or networked with other compatible workstations.

Software and Workstation Design Considerations for GRASS

James Westervelt, William Goran
 U.S. Army Construction Engineering Research Laboratory
 Champaign, Illinois 61820

Products are designed in consideration of end users needs, current technology available resources of time and funding, and the developer's expertise. This paper discusses how these elements influenced the software and workstation design of the Geographical Resources Analysis and Support System (GRASS). The implementation to accommodate many different applications and emerging workstation designs.

Implementation of the Land Analysis System on a Workstation

Lyndon R. Oleson
 U.S. Geological Survey
 EROS Data Center, Sioux Falls, SD 57198

The Land Analysis System (LAS) provides a broad range of functional capabilities in the general areas of image processing and analysis, tabular data processing and custom product generation. To enhance the functionality and utility of LAS to its users, implementations of LAS are being extended to microprocessor-based workstations. The LAS host-computer and workstation implementation approach centers on the development of highly transportable and functionally modular software and the utilization of hardware-independent interfaces and integration techniques. This includes the conversion of LAS to the UNIX operating system environment to further reduce hardware dependencies and to expand the utility of LAS to users on a broad range of processors.

Urban Change Detection with Satellite Data:
An Example of Riyadh, Saudi Arabia

Khalaf Ali Alhaidey
UTSI THESIS

Information about changes in urban areas are of great interest to city planners for prediction and planning future services. Remote Sensing technology has played a major role in the study and investigation of Earth's resources and the environmental conditions. The Landsat Multi-spectral Scanner-acquired data have been of great use in different applications in multidisciplinary experiments on a global scale. An important application of the Landsat data has been directed to the study of time-variant processes, such as detecting and monitoring changes in urban landuse. The Landsat MSS images of 1973 and 1979 of Riyadh, Saudi Arabia, were subjected to computer processing for detecting changes in urban development. Computer processing consisted of enhancing, registering and subtracting the two images to highlight the changes in urban development during that period. The products of the computer processing were thematic overlays and maps and statistics.

03

Application of Landsat Imagery to Monitoring
Sand Dunes Movement in the Sahara Desert

Ahmed Mokhter Brera
UTSI THESIS

Each year the encroachment of the Sahara Desert upon previously arable land lays waste to vast areas in Africa. Major programs of sand fixation and afforestation have been undertaken by many African countries. These progressive efforts should be supported by the analysis and definition of factors relating to dune movement, formation, and growth. This work is best performed by analysis of satellite imagery for both functional and economic reasons. Analysis of the imagery available from Landsat 1 and Landsat 2 for the period between 1972 to 1976 over selected test areas in the Libyan Desert has been undertaken. The first step in the use of Landsat imagery is to employ it for identification and mapping of distinct units on the ground. This was carried out using band 5 and band 7 imagery which contain more information suitable for our purpose than do the other bands. The results obtained in this thesis were supported by ground truth data in the form of topographic, geologic, and soil maps.

TERM FREQUENCY DATA

QUERY TERMS

EFFECTS
DROUGHT-STRESSED
AGRICULTURAL
INFRARED
IMAGERY
INTERPRETATION

DOCUMENT-QUERY MATCHES = 39

SORTED TERM FREQUENCIES AND ASSOCIATED INDEXES

1	46	3
2	48	3
3	58	2
4	69	2
5	2	1
6	3	1
7	6	1
8	7	1
9	9	1
10	11	1
11	12	1
12	16	1
13	17	1
14	23	1
15	24	1

Principal Component Analysis of Aerial Video Imagery

William J. Kramber, Kamlesh Lulla
 Indiana State University Remote Sensing Lab
 Dept. of Geography & Geology
 Terre Haute, Indiana 47809

Arthur J. Richardson, Paul R. Nixon
 USDA, Agricultural Research Service
 P.O. Box 267
 Weslaco, Texas 78596

Aerial video images of an agricultural test site were analyzed using principal component analysis and image processing techniques. The site--six treatments and four replications of cotton, sorghum, cantaloupe, johnsongrass, pigweed, and soil--was imaged using blue, yellow-green, red, and infrared filters over the lenses of four black-and-white video cameras, on 31 May and 24 July 1983. Separate principal component analysis procedures were applied to the May and July data as part of a methodology to assess data dimensionality and structure. Supervised minimum euclidean distance classification procedures were conducted on sets of data that consisted of all four principal components, the first three principal components, the first two, and the first principal component. Results indicated that the number of components required to accurately represent the four band data sets was three for May data and two for July data. Scatter diagrams of principal components 1 and 2 are able to determine the level of plant development.

Leaf Pubescence Effects of Woolly Stemodia (*Stemodia Tomentosa*) on Visible and Infrared Light Reflectance

J. H. Everitt, H. W. Gausman
 USDA, Agricultural Research Service
 P.O. Box 267
 Weslaco, TX 78596

The younger leaves of woolly stemodia (*Stemodia tomentosa*) are covered with a densely white-lanose pubescence (hairiness). These hairs were removed manually from the leaves upper (adaxial) surface and then their light reflectance measured over the 0.5 to 2.5 μm spectral region and compared with that of typical pubescent leaves. Pubescent leaves had higher reflectance over the entire 0.5 to 2.5 μm region than leaves which had the hairs removed, but the greatest differences occurred in the visible (0.5 to 0.75 μm) region. At the 0.55 and 0.65 μm wavelengths pubescent leaves had 117 and 190% higher reflectance, respectively, than that of leaves which had the hairs removed. These data indicate that plant species with heavily pubescent foliage should be easily distinguished from sparsely pubescent or nonpubescent plant species on remote sensing imagery that is being used to make management decisions for rangelands.

Microscope Aids Image Analysis of 70mm Color Infrared Film

G. J. Edwards, C. H. Blazquez
 University of Florida, IFAS
 Citrus Research and Education Center
 700 Experiment Station Road

William E. Wildman
 Cooperative Extension

University of California, Davis
Davis, CA 95616

Color infrared (IR) aerial photography has been used since 1983 to map the spread of phylloxera in grape vineyards of Salinas Valley, California. Annual color IR photographs along with yield measurements, are used to determine the optimum time to replace entire blocks of vines. Image analysis of 70mm aerial color IR film was used to determine the area of individual vines. A microscope replaced the video camera lens of the image analysis system to measure the small images on the 70mm film. The vine image was enhanced by placing a 10 nanomet (nm) bandpass 700 nm filter on the light source of the microscope. Image analysis of gray levels, representing the vines and light colored soil, light colored soil alone, dark colored soil, shadows with dark colored soil, determined the area and percent of total area for each gray level. Infection centers are indicated when the vine and shadow of the same vine are small.

69

Landsat Thematic Mapper Images for Hydrologic Land Use and Cover

L.J. Trolier, W.R. Philipson, W.D. Philpot
CLEARS
Cornell University
454 Hollister Hall
Ithaca, NY 14853

A Landsat Thematic Mapper (TM) image of Rochester, N.Y., was visually and digitally analyzed to determine how well 22 hydrologically important land use and cover classes could be identified. Visual interpretations at a scale of 1:70,000 recognized 16 of the classes, while digital classification with a maximum likelihood classifier recognized nine classes with a very high degree of confidence. Bands 3,4, and 5 provided most of the information for both visual and digital analyses, and would be a sufficient subset of seven TM bands for either method. Greater land use detail was interpretable through visual analysis because spatial characteristics were included, while the digital analysis recognized more general classes, representing mostly cover differences.

02

Mapping and Comparison of Landsat Lineaments with Gravity Trends in West Tennessee

Demetre P. Argialas
UTSI THESIS

Since it is indicated that earthquakes resulting from faulting are of fairly common occurrence in West Tennessee, and also that it is difficult to recognize faulting within the alluvial valley because of the unconsolidated nature of the sediments, it was considered necessary to study the lineament pattern of the area. The study contains evaluation of Landsat imagery for different spectral bands, seasons, image enhancement, and photointerpretation techniques. A method for finding gravity trends was applied to the newest gravity maps of West Tennessee. Statistical programs that express the main trends of the Landsat and gravity lineaments in rose diagrams and histograms were used. Finally, the results of the study of Landsat and gravity lineaments were compared to the tectonic model of West Tennessee.

VITA

Kay Frances Irby was born in Memphis, Tennessee, on August 22, 1962. She attended South Side High School where she was a member of the National Honor Society, Who's Who Among American High School Students, a member of the Student Council, Spanish Club, an All-American Athlete in basketball, and a member of the 1977-78 Championship State Track Team. She received her high school diploma in May, 1980.

Miss Irby enrolled at The University of Tennessee at Chattanooga where she majored in Computer Science. She was a former vice-president of Delta Sigma Theta Sorority Inc., a member of the nationally ranked varsity women's basketball team, a member of the Black Students Association, and also completed an internship in Computer Science at the Tennessee Valley Authority in Chattanooga, Tennessee. She received a Bachelor of Science degree in Computer Science in December, 1985.

Miss Irby entered The University of Tennessee Space Institute in June, 1986. She became a senator in the student government association, treasurer of The American Institute of Aeronautics and Astronautics, and helped organize a group to speak to minority students on "Career Awareness". In December of 1988, she was awarded a Master of Science degree in Engineering Science.