DOCUMENT RESUME

ED 049 633 EM 008 880

AUTHOR	Hazlett, C. B.
TITLE	MEDSIRCH: A Computerized System for the Retrieval of Multiple Choice Items.
INSTITUTION	Alberta Univ., Edmonton. Dept. of Educational Administration.; Royal Coll. of Physicians and Surgeons, Ottawa (Ontario).
REPORT NO	DERS-3-70
PUB DATE	Aug 70
NOTE	72p.; Research and Information Report
EDRS PRICE	EDRS Price MF-$0.65 HC-$3.29
DESCRIPTORS	Computer Oriented Programs, *Computer Programs, Coordinate Indexes, Costs, Data Bases, *Information Retrieval, Information Storage, *Medical Education, *Multiple Choice Tests, Search Strategies, *Test Construction
IDENTIFIERS	*Medical Search, Medsirch

ABSTRACT

Medsirch (Medical Search) is an information retrieval system designed to aid in preparing examinations for medical students. There are two versions of the system: a sequential access file suitable for shallow indexing with a broad choice of search terms and a random direct access file for deep indexing with a restricted range of choices for search items. A user who knows exactly which items he wants may retrieve them by providing a list of the appropriate item identification numbers. If a user knows only the characteristics of the items he wants, he must submit a coded profile outlining the search restrictions that should and/or should not be met by the retrieved items. A thesaurus lists the coded variables and values which describe each item: medical subspecialty, type of question (single or multiple answer), taxonomic level (factual, comprehension, or problem solving), difficulty level, last year question was used, etc. The profiling or structuring procedure for search requests is detailed. Results from the use of random and sequential versions of the system are presented in order to document a comparison of the two methods. An expanded presentation of this system appears in the author's unpublished master's thesis. (JY)

RESEARCH AND INFORMATION REPORT

DERS-3-70
August, 1970

MEDSIRCH:
A Computerized System for the Retrieval of
Multiple Choice Items

C. B. Hazlett

Grant support

DIVISION OF EDUCATIONAL RESEARCH SERVICES
FACULTY OF EDUCATION
THE UNIVERSITY OF ALBERTA
EDMONTON

DERS-3-70

RESEARCH AND INFORMATION REPORT

MEDSIRCH: A Computerized System for the Retrieval of Multiple Choice Items

By C. B. Hazlett

Developed under the auspices of the:

R. S. McLaughlin Examination and Research Centre
Royal College of Physicians and Surgeons of Canada, and

Division of Educational Research Services
Faculty of Education
The University of Alberta
Edmonton, Alberta

ABSTRACT

The purpose of this documentation is to describe the design and implemented modifications made to the Medsirch[1] retrieval system; this description includes profiling examples to illustrate the retrieval potential of this system.

Results from the use of random and sequential direct access files is also reported for purposes of comparing the desirability and feasibility of implementing such files.

---

[1]Medsirch is an acronym for medical search, a program designed to retrieve medical multiple choice questions. The original documentation of this system was described in a master thesis by this author (1969). If the reader has more extensive interest in the data management organization, record preparation and storage, updating facilities, and supporting programs he should refer to the cited thesis.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

The use of information storage and retrieval systems is a matter of everyday experience for literate people. The public library, correspondence files, accounting systems, directories, dictionaries, and so on, are all information systems. All are comprised of records to which one may address a variety of allowable questions with a reasonable expectation of retrieving a selection of records in response to each question.

Medsirch is a machine system for the storage and retrieval of multiple choice items. At present it is being used by the R. S. McLaughlin Examination and Research Centre; it is hoped, however, that some of the design features in this system will provide a basis on which other examining bodies can receive similar services for the retrieval of large masses of data.

The particular advantage of using a machine, that is, a computer, for retrieval is pointed by Baruch (1966, p. 27). He feels that computers greatest assistance is doing tasks such as sorting, filing, indexing, searching, and particularly, being alert for low probability occurrences. Indeed it is this kind of "light thinking" that computers do especially well and that intelligent people seldom do correctly.

In any retrieval system, machine based or otherwise, records are created and organized before the specific questions a system is to answer have been stated (that is, the system is created in anticipation of needs that are not fully known). Lipetz (1966, p. 178) points out that it would be impossible to design a retrieval system that could respond to all possible questions and prohibitively expensive to try to approximate such a condition. The type of questions which the Medsirch system was designed to handle is explained in chapter three of this report. Chapter four examines the limitations that the Medsirch imposes on the user's questions, chapter five specifies the cost of asking them, while Appendices A and B provide the thesaurus and profiling procedures for submitting these questions.

However, it is not only the type of questions which will be addressed to a system which influences its design. Consideration must also be given to the characteristics of the medium in which the records are to be stored and retrieved. This author (p. 90, 1969) has already indicated that one of the limitations in the retrieval field is the lack of comparisons being made between different types of file organizations. The literature is not lacking in suggesting hypothetical designs; however, this source gives little or no concrete evidence as to which file organization is most useful, efficient, and/or economical. In order to provide more information to the reader regarding the differences between sequentially and randomly accessed files, chapter two will discuss the merits and demerits of these models as related to the Medsirch system. While it is true that the discussion is in terms of searching multiple choice items, some of the features specified are applicable to any data base.

CHAPTER TWO

COMPARISON OF MEDSIRCH FILE ORGANIZATIONS

Sequential Organization

Also referred to as Direct File Organization, this method retrieves items by a sequential scan of the complete file. Salton (1968, p. 244) indicates that such a file is suitable if information is to be retrievable according to a variety of different keys "since it is not usually possible to store many copies of the same file to account for the various desired file orders." The response time for sequential file searches is not optimal, however, since a complete file scan is generally needed before any information can be retrieved. Updating files with this type of organization is also disadvantageous since rewriting sequential files is usually done by copying records from one data set to another as needed. This is expensive and would only be done when a number of records have to be altered.

Random Organization

In such a file records are stored and referenced on the basis of the relationship between the key of a record and the direct address of the location where the record is stored. This address is used when a record is stored and again when it is to be retrieved. There are three methods generally used for accessing records - direct address, dictionary look up, and calculation - the first of which was used in the Medsirch system. Direct address is used if the programmer, knowing the precise size and number of of records in his data is able to supply the direct address at storage time.

Bleier and Vorhaus (1968) found some advantages in the use of random access: (a) queries were retrieved rapidly since only relevant records were searched, and (b) the size of data base had little effect on the speed of retrieval. However they also indicated the disadvantages: (a) increased storage requirements to handle the list of addresses in core, and (b) a significant increase in the complexity of maintaining the system. Dodd (1969) also pointed out an additional shortcoming of random access files.

> Although random organization does allow for rapid access of a particular record with a known key, it is not suited for rapidly accessing a number of records. This limitation is imposed by time taken by the handware access mechanism to locate a record. [p. 122].

Dodd (1969) as well as IBM (1966) point out that records must be fixed length if stored in random access; any data base with variable length records must be either manipulated to form fixed length or be stored inefficiently as fixed length records of maximal size. Finally, IBM (1967, pp. 72-73) points out that before a random direct access data set can be used the machine must locate, format and write a skeleton record for each record in the information bank. Senko (1969, p. 121) states that this loading of a random file is 20 to 100 times longer than the corresponding loading done sequentially. Since this is very slow random access data sets are usually created and then preserved for the life of the file.

Medsirch Results

In general the Medsirch system supports the literature in the comparative use of sequential and random files. It has been found, for instance, that updating sequentially is only justified when there is a large number of records to be inserted, deleted, and/or modified (cf. p. 3). It has also been found that to skeletalize a random file took approximately 8 minutes for 10,000 records, a time which prohibited the use of creating a temporary random file for each batch of retrievals. (The reader should note that this time taken to set up the random access file should be spread over the life of the file.) The relative cost of setting up skeleton records is inversely related to the number of requests made to the bank between updates. If the bank is moderately active, requiring regular updates, this installation cost reduces the efficiency of random access noticeably. In the Medsirch system at the present time there is almost a one to one relationship between the number of requests and the number of updates. As such random access installations costs are enormous, relatively speaking. On the other hand, 10,000 records are transferred from tape to sequential disc in approximately 0.21 minutes (that is, in support of Senko (cf. p. 4) Medsirch found sequential loading to be 40 times faster than random loading). Sequential loading time is so slight that a temporary data set can be created for each batch of search requests and thus eliminates the cost of permanent disc storage.

Since random files require fixed length records (cf. p. 4), and the data base of the Medsirch system was variable in length, the author chose to make the two compatible by programming. This required little effort and did not in any way distract or add to the feasibility of random access in the Medsirch system.

However, in random files the job control language (JCL) for the IBM 360/67 does not handle blocksizes longer than the logical record length (IBM (1967, p. 56)). It is here that sequential files show a distinct advantage since JCL will accomodate a blocksize of 7294 bytes on sequential disc. Using IBM's (1967) figures it is possible to show what this advantage is. There is an average access time of 75 milliseconds, average rotational delay of 12.5 milliseconds, and a transmission time of .26 milliseconds (for a total of 87.76 milliseconds) per 80-byte-record. Thus to access 91 80-byte-records it would take approximately 8 seconds (91 × 87.76 milliseconds). If these 80-byte-records were blocked with 91 records per block it would take only 1/10 of a second to access, or 80 times as fast. Thus sequential files which are blocked in this way can access 91 sequential records, 80 times faster than an unblocked random file accessing those same 91 records one at a time. Thus if a search is only made for 10 records within a bank of 10,000 records random access would take 8/10 of a second (10 × 87.76 milliseconds); a sequential search would take 11 seconds (.1 (10,000 ÷ 91)) to access the same 10 records. However, to access 200 records (2% of the bank) random access would take over 17 seconds (200 × 87.76 milliseconds) and sequential access would still be 11 seconds. That is to say, the number of records being accessed has negligible time effect in sequential files since the entire file must be searched for each request; this does not apply to random access since only relevant records are accessed. The reader should note that these figures reflect the differences between input/output (I/O) times for sequential and random access.

Dodd (cf. p. 4) mentioned that random access was unsuited for accessing "a number of records". More specifically, if the sequential file has block sizes 91 times as great as the block sizes of random files, the execution time for I/O will be less in sequential files if one is accessing more than 1 1/2% of the records in a bank of 10,000 80-byte-records. Since Bleier and Vorhaus (cf. p. 4) found random access almost invariant to the size of the pool, one cannot make a generalized statement regarding this 1 1/2% trade off between random and sequential files. It works out, in fact, that if the pool had 100,000 records, one would have to access more than 12 1/2% of the pool before sequential I/O time would be less than random access I/O time. The following algorithm can be used by the reader to estimate the trade off for his data base of 80 byte records.

$.1(N \div 91) = T$

Where N = number of records in total bank
91 = maximum blocking factor for 80 byte records
T = I/O execution time (seconds) for sequential search of bank.

$\frac{T}{.08776} = R$

Where R is the number of records retrieved at even trade-off between random and sequential I/O time.

To convert R to a percentage:

$\frac{R}{N} \times 100 = P\ {}^{\circ}/_{\circ}$

Thus if one is retrieving less than P °/₀ of the bank random access I/O time will be less.

The above algorithm does not reflect the trade-off in terms of total execution time unless the amount of calculations done independent of I/O remains constant in both random and sequential programs: Table 1 shows that in the Medsirch system the amount of calculations independent of I/O

TABLE 1

COMPARATIVE EXECUTION TIMES
(in average minutes per multiple choice item)

File Organization	Implemented Program	Loading Time[1]	Average Search Times I/O	CALCULATION	TOTAL
Sequential	Medsirch - 3	0.21	0.0017	0.030	0.0317
Random	Medsirch - 4	8.00	0.008	0.0085	0.0165

[1]This time must be included in comparing sequential and random execution times. Loading time has been kept separate from the "Average Search Times" in this table since the relative cost of loading time in random files is inversely related to the number of requests made to the bank between updates (cf. p. 5). All cited figures are based on the use of the IBM 360/67 computer.

is greater for sequential searches; this is mainly due to the following reasons. (1) Medsirch - 3 (sequential search) was developed and modified over a period of two years, with each additional feature being added separately and on the basis of programming simplicity, not on the basis of execution efficiency. Medsirch - 4 (random search) was developed after Medsirch - 3, with all features being incorporated simultaneously; as such Medsirch - 4 was written in a more efficient manner. (If the reader has programming experience he will appreciate the difference between these two situational requirements.) Until Medsirch - 3 is completely rewritten for maximal efficiency, the calculation time estimated for Medsirch - 3 should be regarded as an upper limit. (2) The nature of the data base in the Medsirch system necessitates more calculations when sequentially searched. If only one record (or a fixed number of records) was selected per retrieval, this additional calculation would not be necessary. Multiple choice questions, however, vary in length from five to 100 records, and thus a check must be made on each record to determine if it is the last record for a particular multiple choice question.

One additional comment should be made here regarding the above algorithm. If one were to combine 91 records into one read/write statement so that logical record lengths were increased to the blocksize used in sequential searches one might overcome the limitation of no JCL blocking in random access. This would of course impose at least two constraints. (1) All records would have to be read/written under the same format, namely alphanumeric, and as such only logical (not arithmetic) comparisons would be used. The implication of this is discussed later (cf. p. 20).

(2) One would have to hold in core, addresses to locate the appropriate 7280 byte record as well as the part of that record which was wanted for retrieval. Therefore while the cost of I/O time may be reduced, the cost of core storage would be increased. The issue of core storage provides another basis for comparing sequential and random searches in the Medsirch system and will be now discussed.

Medsirch - 3 (sequential) and Medsirch - 4 (random) required 96K and 188K bytes of core storage respectively. These figures reflect the fact that additional space is needed for dictionaries and addresses when random direct access is used. Furthermore, as the item pool increases core requirements for Medsirch - 4 go up by a ratio of 1K for each nine additional multiple choice questions while core requirements for Medirch - 3 remain relatively unchanged.

The differences in execution times and core requirements of sequential and random access indirectly determines the useability of these two files. The cited core requirements for Medsirch - 4 (random search) is based on a pool of 648 multiple choice questions; if the pool was twice as large (1296 items) core requirements would be 254K. It is obvious that as the item pool increases one might have to reduce the choice of search terms; for example, instead of using all of the 57 variables in Appendix A, use only 26 variables for each batch of requests.

On the other hand, not only is the core requirements of Medsirch - 3 relatively unaffected by the size of the pool, but it is also relatively unaffected by the number of search terms in Appendix A. Medsirch - 3 is, however, restrictive in the number of terms one may use simultaneously.

This is due to the fact that items which do not meet all, but do meet some, search terms may also be considered as relevant by the user. Such items in Medsirch - 3 are written onto additional temporary data sets, and may be retrieved later if the main pool does not provide enough items meeting all search terms. Thus if a large number (e.g."X") simultaneous search terms were used, it would also be necessary to use "X" additional data sets in a generalized program. I/O time was found to increase significantly with each additional simultaneous search term, and partially accounted for the fact that Medsirch - 3 I/O time was not always significantly less than Medsirch - 4 when large portions of the pool were retrieved.

Salton has pointed out the applicability of sequential files (cf. p. 3). More specifically this author suggests that if one's data lends itself to deep indexing, but within a restricted range of choices for search terms, random access seems to offer the greatest flexibility. On the other hand, if one's data requires a very broad choice for search terms and can be searched with shallow indexing, sequential searches seem to be a more viable alternative than random searches. However, one must also consider the average proportion of the total pool being retrieved as well as the feasible amount of core storage, before deciding which file--sequential or random--is most suitable to his particular needs.

Since shallowing indexing with a broad choice of search terms is suitable to the needs of the R. S. McLaughlin Centre, and because the average retrieval time per item for Medsirch - 4 is not significantly better than Medsirch - 3, this author must concur with Senko's (1969, p. 121) statement that "the applicability and desirability of random access ... become [sic] extremely restrictive." In summary the reader should consult Table 2 for a list of the summarized differences between random and sequential files as found in the Medsirch system. What is now necessary is an investigation to determine where in this continuum of useability list files are to be placed.

TABLE 2

DIFFERENCES BETWEEN SEQUENTIAL AND RANDOM FILES
(as found in the Medsirch System)

A. Random:

1. Core requirements are greater than sequential.
2. No JCL provision for blocking:
   (a) I/O time thus is increased;
   (b) if blocking done by programming:
      (i) only logical comparisons possible,
      (ii) core requirements are increased further.
3. As the number of records in the bank increases:
   (a) core requirements increase,
   (b) execution time remains relatively constant.
4. Suitable for deep indexing.
5. Not suitable to a large choice of search terms.
6. Permanent disc space required:
   -loading time is 40 times greater than sequential.
7. Updating:
   (a) if records are deleted or replaced execution time is efficient;
   (b) if records are inserted as additions efficiency is poor.
8. Must use only fixed length records.
9. Adequate maintenance of file is more involved.
10. Not suited to retrieving large portions of bank.

B. Sequential:

1. Core requirements is less than random.
2. JCL blocking is available:
   -I/O time for maximally blocked 80-byte-records is approximately 5% of execution time.
3. As the number of records in the bank increases:
   (a) core requirements remain relatively constant;
   (b) execution time is increased.
4. Suited to shallow indexing.
5. Allows a great variety of searchable terms.
6. Temporary disc space is only needed.
7. Updating:
   (a) requires rewriting entire data set;
   (b) no particular difference between deletions, changes or additions.
8. Fixed or variable length records can be used.
9. Maintenance of file is minimal.
10. Not suited to retrieving small number of records from bank.

CHAPTER THREE

MEDSIRCH STRATEGY

In order to search each item (I) in the pool it was categorized as $I_{V_{1,k};\ V_{2,k};\ \ldots;\ V_{57,k}}$, where $V_{j,j=1-57}$ are variables identifying such item parameters as area of subspecialty, type of question, taxonomic level, etc. Each variable (V_j) has its own subdivisions (k); that is, each variable has certain values. For example, variable $V_{1,k}$ (area of subspeciality) may take values of k=1,2,...,23 where each value of k stands for allergy, cardiovascular, ..., physiology respectively. The reader should refer to Appendix A for a list of all variables and the values each variable may take. This thesaurus contains all search terms (i.e., search restrictions ($V_{jk's}$)) available in the Medsirch system.

The basic strategy for retrieval in this system is flowcharted in Figure 1. The reader may wish to consult this chart as the following explanation is given.

The Medsirch strategy makes provision for retrieving items on the basis of prior knowledge of the item bank and also on the basis of no prior knowledge. If the user knows exactly which items he wants he may retrieve them by providing a list of item identification numbers (Figure 1: C, N, O). If the user does not know exactly which items he wants, but does know the characteristics of such items, he must then submit a request specifying what search restrictions ($V_{jk's}$) items should or should not meet. Such a request is called a profile.

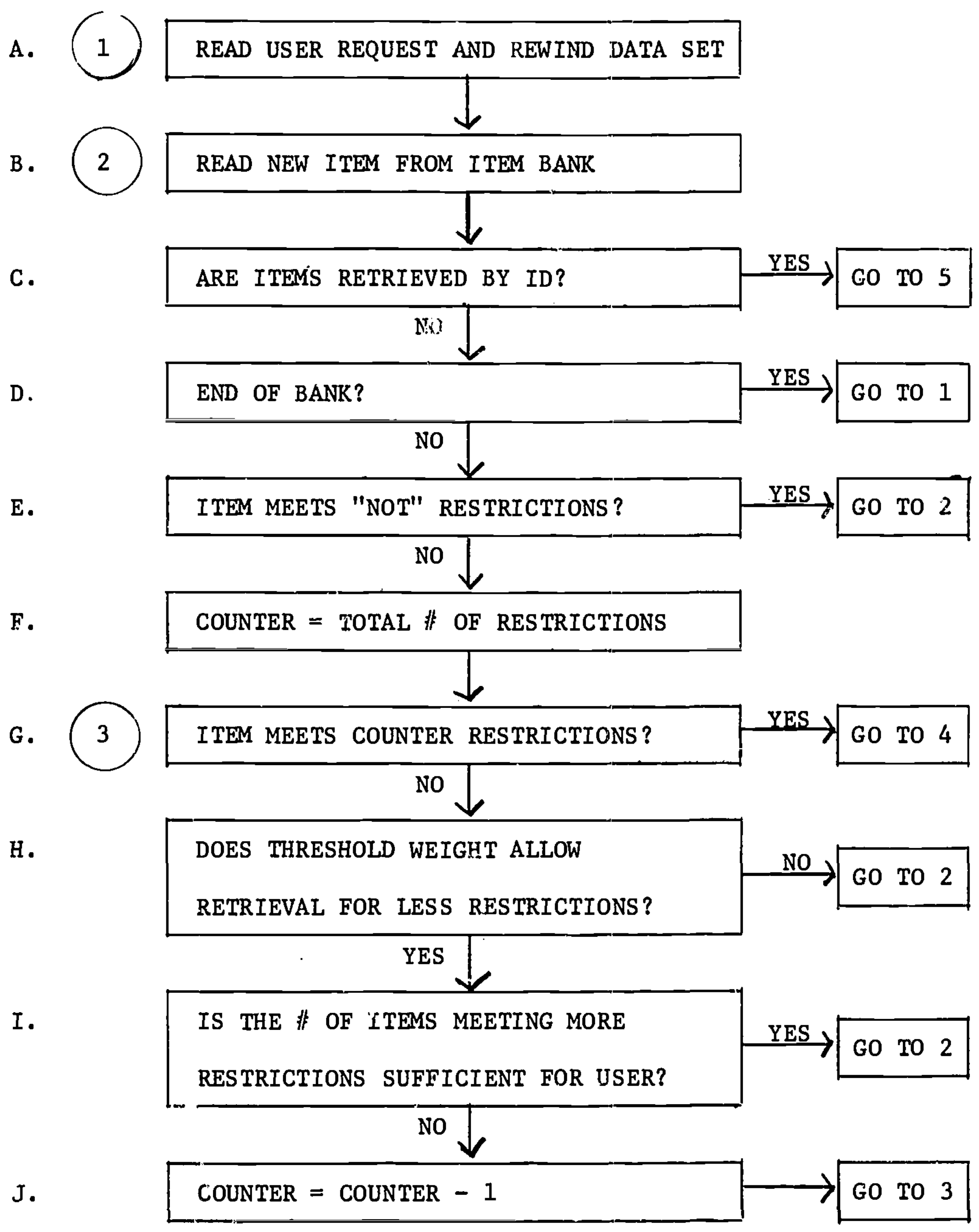

Figure 1. Strategy for Medsirch

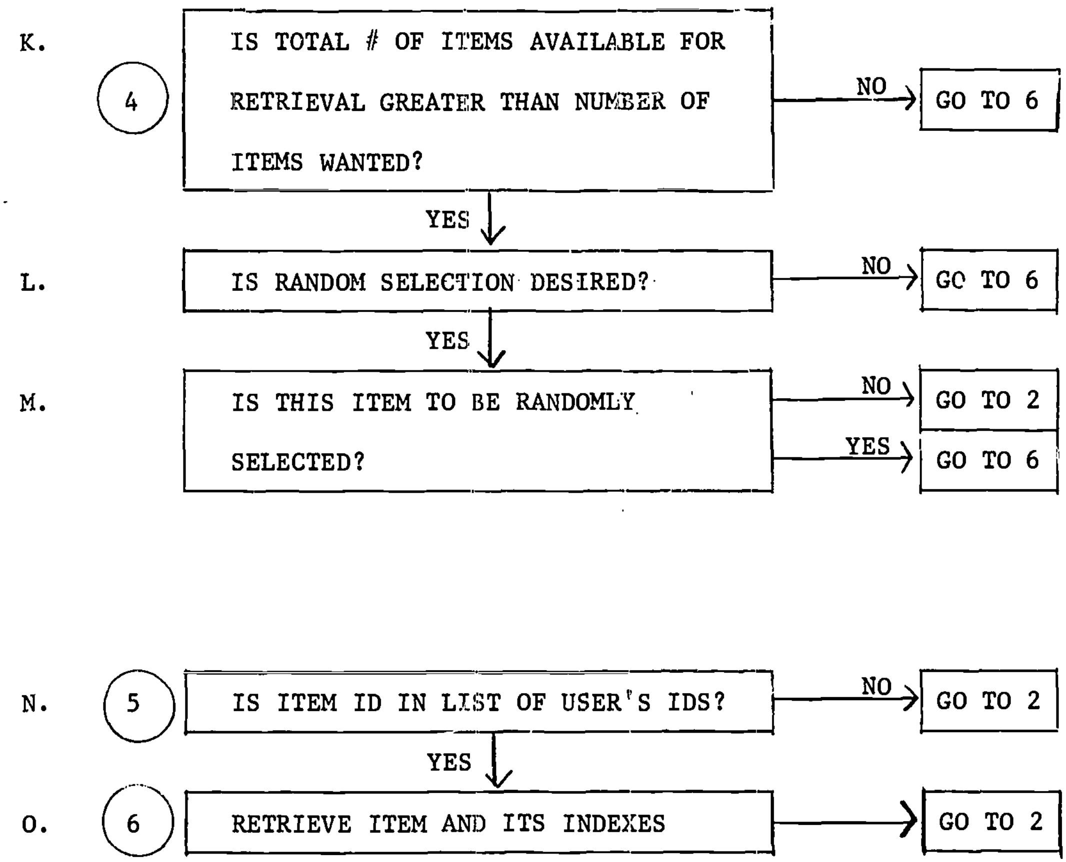

Figure 1. Strategy for Medsirch

For each profile submitted by the user a search is made of the entire bank (Figure 1: A, B, D). An item which is immediately ignored may have one or more of the following characteristics. (1) It may possess some "Not" characteristics (that is, an item may have a characteristic which the user does not want); see Figure 1: E. (2) An item may meet no search restrictions (that is, it does not match any terms ($V_{j,k}$'s) in the user's profile; see Figure 1: F - J. (3) The number of search restrictions it does meet may be below the threshold weight, where threshold weight is defined as the number of search restrictions that must be met by an item in order for it to be retrieved; see Figure 1: H.

The remaining items are considered potential retrievals, the number of which that is actually retrieved will be decided upon by the interaction of the user's request, the number of documents (i.e., number of multiple choice items) wanted, and the number available for retrieval.

Basic to most retrieval designs is an iterative feature for approximating the user's need if the nature of the bank dictates that the complete request of the user cannot be fulfilled. In the Medsirch system this is accomplished by first retrieving items which meet all restrictions. If this constitutes an insufficient number of retrieved documents, items meeting one less restriction are also selected. If the total number of items selected to this point is still not enough, those documents meeting two less restrictions are retrieved, and so on, until enough items are retrieved or until the threshold weight is reached; see Figure 1: F-M,O.

If the search must iterate to select items which do not meet all restrictions, the user may specify which search restrictions he considers most important. With this information the computer can select items which do not meet all restrictions, but do meet the most important restrictions. In this case, to minimize the amount of effort required by the user in preparing his profile, one of the following user's needs is assumed to exist. (1) The user considers that the order in which he has specified his search terms is important. Hence, if items are to be retrieved that meet less than the total number of restrictions (for example, four restrictions) then items meeting the first three restrictions are required next, then if necessary, the first two restrictions, etc. (2) The user considers that the order in which he has specified the restrictions is unimportant. In this case an iterative search would take items meeting any three restrictions, then any two restrictions, etc. (3) The user wants to preserve the order of his restrictions only up to a certain point, for example, the threshold weight. In this case iterative searches would take any combination of restrictions after the first 'x' number of restrictions had been met. In preparing his profile the user is only required to indicate which one of these three conditons is most suitable to himself.

In general the number of items obtained at any given iteration would be least in case (1) and greatest in case (2) with case (3) providing a number somewhere between these two extremes. Of course the more items obtained at each iteration, the less likely it would be that any further iterations were necessary.

Finally, the user has the option of asking for a random selection of items if the opportunity presents itself; if he does not avail himself of this feature all items, at any given iteration, will be retrieved; see Figure 1: L, M, O. For example, assume the user wanted 10 items meeting four restrictions, and that the bank had 20 such items; the user could retrieve all 20 items or 10 randomly selected from the 20 items available. If searches proceed to less restrictive items the random feature still works. For example, assume the same conditons as before but that only 8 items were available meeting the four restrictions, with 12 additional items meeting just three restrictions. In this case 8 items meeting the four restrictions would be retrieved first; the user could then retrieve the next 12 items or obtain two randomly selected from the 12 in order to get the 10 items he wanted. Note, however, that if the threshold weight had indicated that only items meeting four restrictions were wanted, then randomly selecting the two items, or retrieving all 12, would have been impossible.

To prepare a request the user must use the parameter values specified in the Medsirch thesaurus (Appendix A) and follow the format specifications as given in the Medsirch documentation (Appendix B). The latter Appendix also provides profiling examples.

CHAPTER FOUR

EVALUATION OF MEDSIRCH STRATEGY

The reader may question the use of numbers instead of using the actual words (see Appendix A) for coding and searching. The question is a valid one since there is reason to believe the user may feel more comfortable using the verbalization of his mother tongue rather than an abstract numbering system. However, this author purposely avoided the use of words for the following reasons. (1) Word searches usely involves some form of truncation, which necessarily reduces the readability of output. Medsirch output is directly useable with full text, proper spacing, and complete verbalization of the descriptors.
(2) While truncation is not imperative with word searches, the problem of added storage, user misspellings, and excessive keypunching for both storing and searching becomes more prominent. (3) Logical comparisons are necessary in word searches. In terms of the computer this is less efficient than arithmetic comparisons which are possible if numbers are used for searches. (4) The use of word searches raises the question as to why not search the text of a multiple choice item. It is this author's opinion that multiple choice questions cannot, at the present time, be searched in t is manner. Lipetz (1966) points out that "Satisfactory comparison ... requires the ability to recognize the important features in the word. This is not an easy task to turn over to a machine [p. 177]." Abelson (1968, p. 419) agrees with this point

of view, emphasizing the need for human judgement in information retrieval. He feels that professionals in individual fields of scientific research are essential custodians of knowledge who cannot be replaced by archives of any kind.

The reader may also question the lack of weighting facilities for each search restriction and the lack of opportunity for the user to express his own strategy with logical operators. This author has tried using some retrieval programs with these options and has encountered the frustrating experience of either obtaining too few relevant articles or so many retrievals that it was impossible to meaningfully use them. In some cases one had to resubmit his profile in order to get what he knew were available articles but had, in previous requests, been unable to find. While the Medsirch system may not eliminate all such frustration, it does not require the user to laboriously devise his own weighting and logical scheme. Most, if not all, advantages of allowing the user to specify his weights and logic is accomplished in the Medsirch system by simply specifying three numbers, one each for the number of the items wanted, threshold weight, and importance of the order of the restrictions. In essence the weighting system and logic scheme is turned over to the computer.

However, the Medsirch system is still hampered by many of the problems in other retrieval systems.

(1) The user is still required to learn the system's profiling technique before he can maximize its usefulness.

(2) The system is not generalizable to any retrieval of information; (i.e., Appendix A is a limited thesaurus).

(3) The computer has not been utilized to its fullest advantage for automatic retrievals.

(4) At present the Medsirch strategy is linear in nature; if the user is defining new items as relevant or non-relevant on the basis of what he has already received, he may in fact be redefining relevancy throughout retrieval. The Medsirch system cannot adapt to this peculiar interaction between the user and the pool of potentially relevant items. One must learn more about the characteristics of each user before there can be less need for the user to do his own profiling.

CHAPTER FIVE

COST OF IMPLEMENTING MEDSIRCH SYSTEM

Before one is able to use the Medsirch system he must of course prepare his item bank. Each multiple choice item is punched onto cards along with two cards holding its descriptors (cf. Appendix A); these descriptors or indexes must be punched according to rigid format specifications. A Fortran program (CHECK) is available for checking the keypunching; other programs are available for stacking cards onto tape (UTILITY), sequentially revising the item pool (UPDATE), dumping the item pool (BANDUM), counting the number of records and items in the pool as well as dumping the pool of indexes (COUNT), and creating a tape for holding the addresses and descriptors of all items to be searched randomly (DICT). While all of these additional programs are not essential, they do facilitate the maintenance of the item pool[1] which, if properly done, allows Medsirch - 3 or Medsirch - 4 to get more efficient and/or adequate retrievals.

Table 3 provides a list of the costs in implementing the Medsirch system, including human requirements (that is, typing, coding, keypunching, revising, selecting relevant items) and machine requirements (that is, tapes, discs, core, execution time). Cost is not given in terms of monetary values since financial cost of human requirements

---

[1]Any new user should not underestimate the importance of maintenance of any pool of data. It is suggested that a specific timetable be established in developing the pool, maintaining it, and retrieving data.

as well as computer time and core space is relative to one's institution. Figures are also included for modified hardware requirements; the reader is cautioned that any suggested modifications made, may reduce efficiency and/or user satisfaction.

TABLE 3

ESTIMATED COST OF IMPLEMENTING MEDSIRCH SYSTEM

Average Minute per Multiple Choice Item

Man	
Typing	7.0 min.
Coding	
New item	3.0
Revised item	6.0
Keypunching	7.0
Selecting relevant item after retrieval	5.0
Reviewer's checking content, spelling, etc.	5.0

Computer		
Program	Input/Output	Total Execution
Check	0.00025 min.	0.005 min.
Utility	0.00023	0.C0032
Update	0.0017	0.034
Bandum	0.00025	0.005
Count	0.000028	0.00046
Dict	0.0000082	0.00082
Medsirch - 3	see Table 1, p. 8	
Medsirch - 4	see Table 1, p. 8	

TABLE 3 (continued)

ESTIMATED COST OF IMPLEMENTING MEDSIRCH SYSTEM

Hardware Requirements[1] Without Modification

Program	Amount of Core (Bytes) Execution	Blocking (2 buffers)	Total	Data Sets[2] Required
Check	4k		4k	
Utility	43k	15k	58k	1 tape
Update	54k	30k	74k	2 tapes
Bandum	45k	30k	75k	1 tape, disc space for 1 temporary data set
Count	21k	15k	36k	1 tape
Dict	32k	30k	62k	2 tapes
Medsirch - 3	24k	72k	96k	1 tape, disc space for 5 temporary data sets
Medsirch - 4	173k	15k	188k	1 tape, disc space for 1 permanent data set

[1]Total Requirements: 96k, 2 different tapes, and disc space for five temporary data sets if using sequential file; or 188k, 3 different tapes, and permanent disc space for 1 data set, and temporary disc space for 1 data set if using random file.

[2]Data sets required in addition to card reader, card puncher, and printer.

TABLE 3 (continued)

ESTIMATED COST OF IMPLEMENTING MEDSIRCH SYSTEM

Program	Hardware Requirements[3] With Modifications[4]			
	Amount of Core (Bytes)			Data Sets[2] Required
	Execution	Blocking (2 buffers)	Total	
Check	4k		4k	
Utility	43k		43k	1 tape
Update	54k		54k	2 tapes
Bandum	45k		45k	1 tape
Count	21k		21k	1 tape
Dict	32k		32k	2 tapes
Medsirch - 3	24k		24k	1 tape
Medsirch - 4	173k		173k	disc space for 1 permanent data set

[2] See footnote number 2, p. 26.

[3] Total Requirements: 54k, 2 different tapes if using sequential file; or 173k, 3 different tapes, and permanent disc space for 1 data set if using random file.

[4] Modifications possible:
a. use only tapes
b. do not block tapes
c. do not iterate to retrieve items

NOTE: If modifications (a) and (b) are used efficiency will be poorer
If modification (c) is used user satisfaction may be less.

SELECTED REFERENCES

Abelson, P. H. Custodians of knowledge. Science, 1968, 159, 1.

Baruch, J. J. Information system applications. In C. A. Cuadra (Ed.), Annual Review of Information Science and Technology. Vol. 1, New York: John Wiley & Sons, 1966.

Bleier, R. E., & Vorhaus, A. H. File organization in the SDC time-shared data management system (TDMS). Proceedings IFIP Congress 1968. Amsterdam: North-Holland, 1968. (Computing Reviews, 1969, 10 (1), 23-24).

Dodd, G. G. Elements of data management systems. Computing Surveys, 1969, 1 (2), 117-133.

Hazlett, C. B. The storage and retrieval of multiple choice items on computer. Unpublished thesis, University of Alberta, 1969.

International Business Machines. Introduction to IBM system/360 direct access storage devices and organization methods. White Plains: IBM Technical Publications Department, 1966.

International Business Machines. IBM system/360 operating system fortran IV (H) programmer's guide. White Plains: IBM Technical Publications Department, 1967.

Lipetz, B. Information storage and retrieval. In D. Flanagan (Ed.), Information. San Francisco: W. H. Freeman, 1966.

Salton, G. Automatic information and retrieval. New York: McGraw-Hill, 1968.

Senko, M. E. File organization and management information systems. In C. A. Cuadra (Ed.), Annual Review of Information and Technology. Vol. 4, Chicago: W. Benton, 1969.

APPENDIX A

Medsirch Thesaurus

This thesaurus is a list of all variables (V_j, where j = 1 - 57) and the respective values (k, where k = 1,2, . . .) which each variable may take. Each multiple choice item which is stored in the Medsirch pool must be manually coded (classified) according to each variable, except for those variables which are not applicable. Therefore a user may submit a profile with one or more V_{jk}'s to be used as descriptors of items which he wants or does not want retrieved.

VARIABLE (V_j)

1. AREA OF SUBSPECIALTY

Value (k):

"1"	Allergy, Immunology, Serology	(ALL)
"2"	Cardiovascular	(CVS)
"3"	Collagen Diseases	(COL)
"4"	Dermatology	(DERM)
"5"	Chemical of Physical Agents	(PHYSCHEM)
"6"	Endocrinology and Metabolism	(END MET)
"7"	Gastrointestinal, Liver, Pancreas	(GI)
"8"	Hematology	(HEMAT)
"9"	Infectious Diseases	(INF)
"10"	Musculoskeletal	(SKEL)
"11"	Neurology	(NEUR)
"12"	Psychological Medicine	(PSYC)
"13"	Pulmonary	(PULM)
"14"	Renal	(REN)
"15"	Therapeutics	(THER)
"16"	Anatomy	(ANAT)
"17"	Biochemistry	(BIOC)
"18"	Genetics	(GEN)
"19"	Laboratory Medicine	(LABMED)
"20"	Microbiology	(MICROB)
"21"	Pathology	(PATH)
"22"	Pharmacology	(PHARM)
"23"	Physiology	(PHYSIO)

VARIABLE (V_j)

2. TYPE OF QUESTION

Value (k):

"1"	single answer	(SING ANS)
"2"	multiple answer	(MULT ANS)

3. TAXONOMIC LEVEL

Value (k):

"1"	factual	(FACT)
"2"	comprehension	(COMP)
"3"	problem solving	(PROB)

4. CORE LEVEL

Value (k):

"1"	essential	(ESS.)
"2"	more important	(IMP.)
"3"	More unimportant	(UIMP.)

5. SECOND AREA OF SUBSPECIALTY

Value (k): (re. values for variable 1)

6. SOURCE

Value (k):

"1"	American board of interal medicine	(AMIB)
"2"	national board of medical education	(NBME)
"3"	Canada	(CAN)
"4"	United Kingdom	(UK)
"5"	other	(OTH)

VARIABLE (V_j)

7. PROVINCE

Value (k):

"1"	Alberta	(ALTA)
"2"	British Columbia	(B.C.)
"3"	Dalhousie	(DALH)
"4"	Laval	(LAVL)
"5"	McGill	(MCG)
"6"	McMaster	(MCM)
"7"	Manitoba	(MAN)
"8"	Montreal	(MTRL)
"9"	Ottawa	(OTT)
"10"	Queens	(QN)
"11"	Saskatchewan	(SASK)
"12"	Sherbrooke	(SHRB)
"13"	Toronto	(TOR)
"14"	Western Ontario	(UWO)
"15"	Calgary	(CALG)
"16"	Memorial	(MMRL)

8. AUDIO-VISUAL

Value (k):

"1"	Line	(LINE)
"2"	Photo	(PHOTO)
"3"	Color	(COLOR)
"4"	Slide	(SLIDE)
"5"	Movie	(MOVIE)
"6"	Video	(VIDEO)

9. AUDIO-VISUAL ID. LOCATION

10. CHOICE 1 OF ITEM IS CORRECT (PUNCH 1)

11. CHOICE 2 OF ITEM IS CORRECT (PUNCH 1)

VARIABLE (V_j)

12. CHOICE 3 OF ITEM IS CORRECT (PUNCH 1)

13. CHOICE 4 OF ITEM IS CORRECT (PUNCH 1)

14. CHOICE 5 OF ITEM IS CORRECT (PUNCH 1)

15. LANGUAGE

Value (k):

"1"	available in both languages	(BTH. LANG)
"2"	available in English only	(ENG. ONLY)
"3"	available in French only	(FR. ONLY)

16. NUMBER OF TIMES USED

17. LAST YEAR QUESTION USED

18. NUMBER OF QUESTION ON LAST EXAM

19. EDUCATIONAL LEVEL

Value (k):

"1"	graduate exam	(GRAD.)
"2"	undergraduate exam	(UGRAD.)

VARIABLE (V_j)

20. LOCALITY LEVEL

Value (k):

"1"	national exam	(NAT. EXAM)
"2"	local exam	(LOC. EXAM)

21. ID OF EXAM

22. NUMBER OF EXAMINEES ON LAST EXAM

23. "p" FOR LAST RECORDED TESTING YEAR (SINGLE-ANSWER-TYPE OF QUESTION)

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

24. "p" FOR SECOND LAST RECORDED TESTING YEAR (SINGLE-ANSWER TYPE OF QUESTION)

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

25. r_{bis} FOR LAST RECORDED TESTING YEAR (SINGLE-ANSWER TYPE OF QUESTION)

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

VARIABLE (V_j)

26. r_{bis} FOR SECOND LAST RECORDED TESTING YEAR (SINGLE-ANSWER TYPE OF QUESTION)

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

MULTIPLE-ANSWER TYPE OF QUESTION: "p" FOR LAST RECORDED TESTING YEAR.

VARIABLE (V_j)

27. FIRST CHOICE

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

28. SECOND CHOICE

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

29. THIRD CHOICE

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

VARIABLE (V_j)

30. FOURTH CHOICE

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

31. FIFTH CHOICE

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

32. TOTAL ITEM

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

MULTIPLE-ANSWER TYPE OF QUESTION: r_{bis} FOR LAST TESTING YEAR

VARIABLE (V_j)

33. FIRST CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

VARIABLE (V_j)

34. SECOND CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

35. THIRD CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

36. FOURTH CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

37. FIFTH CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

38. TOTAL ITEM

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

MULTIPLE-ANSWER TYPE OF QUESTION: "p" FOR SECOND LAST RECORDED TESTING YEAR.

VARIABLE (V_j)

39. FIRST CHOICE

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

40. SECOND CHOICE

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

41. THIRD CHOICE

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

42. FOURTH CHOICE

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

VARIABLE (V_j)

43. FIFTH CHOICE

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

44. TOTAL ITEM

Value (k):

"1"	difficult	(.01 - .25)
"2"	average	(.26 - .75)
"3"	easy	(.76 - 1.0)

MULTIPLE-ANSWER TYPE OF QUESTION: r_{bis} FOR SECOND LAST RECORDED TESTING YEAR

VARIABLE (V_j)

45. FIRST CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

46. SECOND CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

VARIABLE (V_j)

47. THIRD CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

48. FOURTH CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

49. FIFTH CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

50. TOTAL ITEM

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

PROPORTION ON LAST TEST SELECTING THESE CHOICES

VARIABLE (V_j)

51. FIRST CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

52. SECOND CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

53. THIRD CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

54. FOURTH CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

VARIABLE (V_j)

55. FIFTH CHOICE

Value (k):

"1"	low	(.01 - .25)
"2"	average	(.26 - .75)
"3"	high	(.76 - 1.0)

56. YEAR ITEM ENTERED ITEM BANK

57. ITEM ID

APPENDIX B

MEDSIRCH PROFILING

COMPUTER PROGRAM DOCUMENTATION

University of Alberta

Division of Educational Research Services

TITLE: Medsirch

MACHINE: IBM 360/67

LANGUAGE: Fortran IV(H)

PROGRAM TYPE: Complete

SUBPROGRAMS: TRANS, DUMP, PDISC, CALC, SORT, SORTRN, SELECT, RANDU, PARMTR, TEXT, GETID

LIMITS:
Maximum 9999 multiple choice items in bank for Medsirch-3 1000 multiple choice items in bank for Medsirch-4
- 100 records per multiple choice item
- 200 items selected randomly per request (cf. p. 19)
- 250 items selected by identification number per request (cf. p. 14)
- 57 descriptors per item (cf. p. 14)
- 4 simultaneous search terms for Medsirch-3 or
- 35 simultaneous search terms for Medsirch-4 (cf. pp. 30 - 42)
- 15 "Not" restrictions (cf. p. 17)

DESCRIPTION:

Medsirch-3 uses sequential access and Medsirch-4 uses random access to search for multiple choice items in an item bank and selects those meeting the user's specifications ($V_{jk's}$); see Appendix A. If more items are available than needed a random selection can be made. If there is not enough items available, items meeting fewer restrictions can be retrieved. A user may specify if the order of his restrictions is important and may also specify a threshold weight (cf. pp. 17, 18). The program assumes the existence of a specific format for each item and its descriptors (as defined in Hazlett (Ch. 3, 1969)), and a specific format for profiling as given in this Appendix.

PROFILING

(Card Preparation for Search Requests)

Card	Column	Title	Description
1		PARAMETER CARD	Use right justification throughout
	1 - 5		Number of items wanted
	9 - 10		Number of restrictions used (i.e. number of $V_{jk's}$) - 4 must be used in Medsirch-3 - 1 - 35 may be used in Medsirch-4 - see Example 1, pp. 51, 52
	14 - 15		Threshold Weight (i.e. minimum number of restrictions to be met) - 1 is assumed - maximum is 4 in Medsirch-3 - maximum is 35 in Medsirch-4 - see pp. 17, 18 for definition and Example 2, p. 53
	19 - 20		Priority (Importance of order of restrictions) - 0: any combination - 1: preserve order - 2: any combination after threshold - ?: give any other positive value (number must not be greater than number of restrictions used (col. 9-10)) - see p. 18 for description and Example 3, pp. 54-57
	25		Punch 1 if random selection is desired, - leave blank if not wanted - see p. 19 and Example 4, p. 58

Card	Column	Title	Description
1		PARAMETER CARD	Use right justification throughout
	26 - 30		Punch any odd integer if using random selection - this number is used to initialize random selection - if col. 25 is blank, leave these columns blank
	34 - 35		Number of "Not" variables - if none used, leave blank - see p. 17 for description and Example 5, pp. 59, 60
	40		Punch 1 if labels are being supplied by the user - if none supplied, leave blank - see Example 6 and 7, pp. 61-63
	45		Punch 1 if items being retrieved by identification number only - leave blank if not using this feature - only col. 1 - 5 and possibly col. 40 need to be filled in if this feature used - see p. 14 for description and Example 8, pp. 64, 65

Card	Column	Title	Description
2(s)		VARIABLE CARD(S)	Use right justification throughout
	1 - 5		Variable number of the first restriction (V_j')
	6 - 10		Variable number of the second restriction (V_j'')
	.		.
	.		.
	.		.
	75 - 80		- specify as many variables as that given in col. 9 - 10 of Card 1 - specify 1 - 35 variables for Medsirch-4 - specify 4 variables for Medsirch-3 - use more cards if necessary - see Appendix A for available variables ($V_{j's}$) and Example 1, pp. 51, 52 - NOTE: if col. 45 of card 1 has a 1 punched then supply ID numbers of items wanted instead of variables; no other cards are needed in profile unless using labels; if using labels supply both Variable Label, Card (3) <u>and</u> Value Label, Card (5)
3		VARIABLE LABEL CARD	Free format allowed
	1 - 80		- this card is not included if col. 40 of card 1 was left blank - if using labels, can only use one card to describe all of the variables ($V_{j's}$) specified in card(s) 2 - see Example 6, pp. 61, 62

Card	Column	Title	Description
4(s)		VALUE CARD(S)	Use right justification throughout
	1 - 5		Value (k') of the variable (V_j') used as first restriction
	6 - 10		Value (k'') of the variable (V_j'') used as second restriction
	.		.
	.		.
	.		.
	75 - 80		- the number of values specified must agree with the number of variables in card(s) 2 - use more cards if necessary - see Appendix 2 for available values (k's) and Example 1, pp. 51, 52
5		VALUE LABEL CARD	Free format allowed
	1 - 80		- this card is not included if col. 40 of card 1 was left blank - if using labels, can only use one card to describe all of the values (k's) specified in card(s) 4 - see Example 6, pp. 61, 62

Card	Column	Title	Description
6(s)*		NOT CARD(S)	Use right justification throughout - if col. 34 - 35 of card 1 were left blank cards 6 and 7 are omitted - see Example 5, pp. 59, 60
	1 - 5		Specify the variable number (V_j) which has 1 or more values which are not wanted
	6 - 10		Specify the first value (k') which is not wanted
	11 - 15		Specify the second value (k'') which is not wanted
	. . .		. . .
	75 - 80		Etc. Continue to specify all values that are not wanted. At least 1 must be given, and a maximum 15 "Not" values can be specified
7(s)*		NOT LABEL CARD(S)	Free format allowed
	1 - 80		- this card is not included if col. 40 of card 1 was left blank - if used punch titles of the variable and its values as specified in card(s) 6 - see Example 7, p. 63

* Repeat Cards 6 and 7 as many times as specified in col. 34 - 35 of card 1 (omitting card 7 if labels are not used).

Summary of Card Input

	Card	1	Parameter Card
	Card	2(s)	Variable Card(s)
*	Card	3	Variable Label Card
	Card	4(s)	Value Card(s)
*	Card	5	Value Label Card
*	Card	6(s)	Not Card(s)
*	Card	7(s)	Not Label Card(s)

* Optional

Can repeat cards 6 & 7 up to 15 times.

Can repeat cards 1 - 7 as many times as desired.

Examples of some possible card inputs of profiles

	Card No.'s:	1,2,3,4,5,6,7,6,7,6,7,6,7
	Card No.'s:	1,2,3,4,5,6,6
	Card No.'s:	1,2,3,4,5
	Card No.'s:	1,2,4,6,6,6
**	Card No.'s:	1,2,4
	Card No.'s:	1,2,3,5
**	Card No.'s:	1,2

** Note that the minimal specification in ones' profile is a card input of 1,2, & 4 if retrieving items by their descriptors, or 1,2 if retrieving items by their ID numbers.

EXAMPLE 1

Search Restrictions ($V_{jk's}$)

Definition.

The term "restriction" refers to those variables ($V_{j's}$) and their respective values (k's) which are submitted by the user to describe the characteristics of items in the pool that he wishes to retrieve. Appendix A provides a list of variables and their respective values a user may use.

The user must provide as many search restrictions ($V_{jk's}$) as the number specified in col. 9-10 of card 1 in his profile.

Sample (a). Assume the user had specified that

(i) he was using 4 restrictions;
(ii) he wanted items meeting the following characteristics:

V_j:	Variable		k:	Value
(1)	Area of Subspecialty	-	(4)	Dermatology
(2)	Type of Question	-	(1)	Single Answer
(3)	Taxonomic Level	-	(1)	Factual
(4)	Core Level	-	(3)	More Unimportant

Referring to Appendix A one can see that these 4 search restrictions ($V_{jk's}$) would be submitted in a profile in the following manner:

Card #2.	Variable Card ($V_{j's}$):	1	2	3	4
Card #4.	Value Card (k's):	4	1	1	3

That is, Area of Subspecialty is variable 1 and has a respective value of 4 if Dermatology is desired, hence the search restriction $V_{1,4}$; the second search restriction $V_{2,1}$ refers to Type of Question - Single Answer, and so on for all four restrictions.

Sample (b). Assume the user had specified that

(i) he was using 4 restrictions
(ii) he wanted items meeting the following characteristics:

V_j:	Variable		k:	Value
(1)	Area of Subspecialty	-	(17)	Biochemistry
(1)	Area of Subspecialty	-	(18)	Genetics
(18)	Educational Level	-	(1)	Graduate
(6)	Province	-	(6)	McMaster

Referring to Appendix A one can see that these 4 search restrictions ($V_{jk's}$) would be submitted in a profile in the following manner.

Variable Card ($V_{j's}$)	1	1	18	6
Value Card (k's)	17	18	1	6

Note. It is possible to submit one V_j with more than one value (k) for it. However, no items could be retrieved meeting all four restrictions since an item cannot have both biochemistry and genetics as its values for variable V_1, area of subspecialty.

Also Note. There is no restriction as to the order of V_{jk}'s. $V_{1,17}$; $V_{1,18}$; $V_{6,6}$; $V_{18,1}$ is as permissible as the above order. That is, the user determines the order in which search restrictions are specified.

EXAMPLE 2

Threshold Weight

Definition.

This term refers to minimal number of restrictions that must be met in order for an item to be retrieved. If the threshold weight is the same as the number of restrictions specified only those items meeting all restrictions will be retrieved. If the threshold weight is less than the number of restrictions specified then items meeting all restrictions will be retrieved first. If the number of items meeting all restrictions is less than the user wanted those items meeting one less restriction will be retrieved next, and so on until the number of restrictions being met by an item is less than the threshold weight. At this point retrieval is arrested.

Sample. Assume the user specified that

(a) he wanted 10 items
(b) he was using 4 restrictions
(c) threshold weight = 4 (i.e. no items were wanted if they met less than 4 restrictions)

Result. If the bank could only find 6 items meeting all four restrictions then only these 6 items would be retrieved.

However if the user had specified a threshold weight of 3 the following could have happened.

Result. The six items meeting 4 restrictions would be retrieved first
and
all or some of the items (depending on whether or not a random selection was desired) meeting 3 restrictions would also be retrieved.

Note. If the threshold weight were 2, items meeting two restrictions would be retrieved only if the total number of retrievals up to that point (i.e. the sum of items meeting four restrictions plus the number of items meeting three restrictions) was less than the number of items wanted by the user, similarly for a threshold weight of 1.

EXAMPLE 3

Priority of Variables and their Values

Description.

The order in which the user has specified his search restrictions may or may not be important to him. This feature allows him to specify this fact but will only be used if the bank does not have enough items meeting all restrictions.

Sample (a).

If the user has left this priority option blank or punched a zero then the order of restrictions is considered not important.

Assume the user wanted

(i) 10 items
(ii) 4 search restrictions (i.e., 4 $V_{jk's}$)
(iii) threshold weight = 3
(i.e. the minimal number of restrictions that must be met for retrieval is 3)
(iv) priority is blank or zero

Result. Assume the bank only had 6 items meeting 4 restrictions; these would be retrieved. Then items meeting any three of the four restrictions would be retrieved next.

Note. If the threshold weight was 2, items meeting any two restrictions would be retrieved if the total number of retrievals (those meeting 4 and 3 restrictions) was not equal to, or greater than, the number of items wanted; similarly for a threshold weight of 1.

Also Note. If the threshold weight was equal to the number of restrictions (i.e. 4) then only 6 items would be retrieved (as already illustrated in Example 2)

Sample (b).

If the user had specified the priority as 1 then the order in which he had specified his restrictions is considered important. Retrieval in this case would be the same as in Sample (a) on page 54 except that instead of "any 3 restrictions", items would be retrieved that met only the first three restrictions.

This difference may be illustrated by the following; assume search restrictions were specified as:

$V_{1,2}$; $V_{17,1}$; $V_{3,2}$; $V_{4,1}$.

If items meeting any three restrictions are acceptable, (priority is blank or zero) then items meeting

$V_{1,2}$; $V_{17,1}$; $V_{3,2}$

are considered as acceptable as

$V_{1,2}$; $V_{3,2}$; $V_{4,1}$

and

$V_{17,1}$; $V_{3,2}$; $V_{4,1}$,

etc.

If items meeting the first three restrictions are only acceptable (priority is 1) then items meeting search restrictions

$V_{1,2}$; $V_{17,1}$; $V_{3,2}$

would only be retrieved, if necessary.

Sample (c).

If priority is specified as 2 then the order is only considered important up to the threshold weight. This feature is only useful in Medsirch-3 if the threshold weight is 1, but has wider applicability in Medsirch-4.

e.g. Assume a threshold weight of 1, priority specified as 2, using Medsirch-3, and search restrictions specified as

$V_{1,2}$; $V_{17,2}$; $V_{3,2}$; $V_{4,1}$

and not enough items in bank meeting these four restrictions, then

$V_{1,2}$; $V_{17,1}$; $V_{3,2}$

is considered as good as

$V_{1,2}$; $V_{3,2}$; $V_{4,1}$

or

$V_{1,2}$; $V_{17,1}$; $V_{4,1}$.

But because priority was 2 and threshold weight was 1,

$V_{17,1}$; $V_{3,2}$; $V_{4,1}$

is not retrieved; that is, retrieved documents must have

$V_{1,2}$

as a descriptor.

e.g. The use of more than 4 restrictions in Medsirch-4 illustrates the use of this feature more vividly. Assume 7 restrictions were used with a threshold weight of 4 and priority was specified as 2. Also assume that the user had specified his search restrictions as:

$V_{1,1}$; $V_{2,1}$; $V_{3,1}$; $V_{4,1}$; $V_{5,1}$; $V_{6,1}$; $V_{7,1}$.

If not enough items meeting all seven restrictions were available then

$V_{1,1}$; $V_{2,1}$; $V_{3,1}$; $V_{4,1}$; $V_{5,1}$; $V_{6,1}$

would be as acceptable as

$V_{1,1}$; $V_{2,1}$; $V_{3,1}$; $V_{4,1}$; $V_{5,1}$; $V_{7,1}$

or

$V_{1,1}$; $V_{2,1}$; $V_{3,1}$; $V_{4,1}$; $V_{6,1}$; $V_{7,1}$

etc.

Note. Because the threshold weight was four and priority 2, retrievals here must meet the first four restrictions and then the selection will take any order of the remaining restrictions.

Sample (d).

If priority is not specified as 0,1, or 2 only Medsirch-4 can handle this option. In this case the user specifies the priority as some value greater than two but not greater than the number of restrictions being used. Retrievals will be the same as in Sample (c) except the number of first "X" restrictions must be met by an item, and then selection will take any order of the remaining restrictions.

Note. In general as one relaxes the importance of the order in which restrictions are specified more items will usually be retrieved since more combinations are possible.

EXAMPLE 4

Random Selection

Description.

This feature may be used to obtain the exact number of items desired. Whenever there are more items available than the user requires, a random selection can be taken from all those items that are potentially retrievable. This feature may be used at any point during retrieval, regardless of the number of restrictions a group of items may be meeting.

Sample (a).

Assume the user wanted 10 items meeting the four restrictions $V_{1,1}$; $V_{2,1}$; $V_{3,1}$; $V_{4,1}$; and also wanted a random selection if necessary; also assume that the bank had 15 items meeting all of the above restrictions. In such a case a random selection of 10 items out of the 15 would be given to the user. If he had not wanted a random selection, all 15 items would have been retrieved.

Sample (b).

Assume the same user specifications are used as in Sample (a) but that the bank had only 8 items meeting all four restrictions; but also assume an additional 12 items meeting three restrictions were available. Provided the user had specified a threshold weight less than 4, a random selection of 2 items would be made from those 12 items in order to supply the user with the 10 items he had requested. In this case he would obtain 8 items meeting four restrictions and two randomly selected items that met only three restrictions. If random selection had not been requested all 12 items meeting the three restrictions would have been retrieved.

Note. While a random selection can be made on any group of items meeting 1 to "X" restrictions, the threshold weight must allow retrieval to proceed to that level. In sample (b), for example, a threshold weight of 4 would have prevented the random selection of the two items.

EXAMPLE 5

"Not" Search

Description.

As the user becomes familiar with the Medsirch system, he may find this feature provides a good deal of versatility by using it in one or more of the following ways.

(1) Any restriction that is not wanted is automatically eliminated from retrieval; (see Example 5, Sample (a)).

(2) By specifying all values of a restriction except one means in practice that one is using an additional search restriction. This is particularly useful in Medsirch-3 since only four search restrictions are used in this program; by using this provision of "Not" one can increase the search terms to 19. Caution: The search cannot iterate (see Example 4) on a "Not - search - term" since all items having descriptors which the user specifies as "Not" are automatically eliminated; (see Example 5, Sample (b)).

(3) By specifying only some of the values of a restriction as "Not" one can search for more than one restriction at a time; (see Example 5, Sample (c)).

Sample (a).

Appendix A indicates that variable 4 (V_4 = "core level") has values of k equalling 1 to 3, corresponding to "essential", "more important", and "more unimportant" respectively. By punching 4 in col. 5 of the "Not" card to indicate the variable "core level" and 3 in col. 10 to indicate a value of "more unimportant" the user is automatically eliminating any possibility of retrieving items with the descriptor $V_{4,3}$ (that is, a core level of "more unimportant").

Sample (b).

Assume the user has not used "core level" as a search term, and that the user specified his "Not" card as core level with values of 2 (more important) and 3 (more unimportant), that is, his "Not" card contained $V_{4,2}$ and $V_{4,3}$. In practice then the only value that can be retrieved is a core level of 1 (essential); as such this is serving as an additional search term since only items meeting essential core level ($V_{4,1}$) can be retrieved. Note, however, that if retrievals drop back to items meeting fewer restrictions, only those items meeting essential core level will still be retrieved. As such, a "Not" variable used in this manner should only be done with a restriction (V_{jk}) that is considered highly important.

Sample (c).

Referring back to Sample (a) one will notice that items meeting a core level of "essential" or "more important" will be retrieved. Therefore this retrieval is also an example of a request using the "Not" option for searching items that may meet either $V_{4,1}$ or $V_{4,2}$;

EXAMPLE 6

Labels

Description.

Before each set of retrievals is printed the computer gives a list of the user's specifications as an aid to him in reading his output. Since the Medsirch system uses numbers and not names for search restrictions ($V_{jk's}$) only numbers will be printed unless the user supplies a card giving the name of these variables ($V_{j's}$) and values (k's). The user is allowed free format in punching these names onto cards.

Sample (a).

Without labels a user's batch of retrievals would be preceded by a heading similar to the following:

VARIABLES USED AS RESTRICTIONS:

1	2	3	4

THE CORRESPONDING VALUE OF THE ABOVE VARIABLES IS:

1	1	1	1

Sample (b).

With labels a user's batch of retrievals would be preceded by a heading similar to the following:

VARIABLES USED AS RESTRICTIONS:

user supplied)
these variable) SUBSPECIALTY, TYPE OF QUESTION, TAXONOMY, CORE
names ($V_{j's}$))

THE CORRESPONDING VALUE OF THE ABOVE VARIABLES IS:

```
user supplied  )
these value    )  ALLERGY, SINGLE, FACTUAL, ESSENTIAL
names (k's)    )
```

Note. Only one card can be used per label and the order in which he arranges these names will be the order in which they will appear in the output.

EXAMPLE 7

"Not" Labels

Description.

The only distinction between this label card and other label cards is that the user must indicate the name of the variable (V_j) first and all names of its values (k's) must follow on the same card. Free format is allowed.

Sample (a).

"Not" restrictions with one variable (V_4) and two values (k = 2) and (k = 3).

THE FOLLOWING RESTRICTIONS ARE NOT WANTED:

	VARIABLE:	VALUE(S):
user prepared this card	CORE LEVEL	MORE IMPORTANT, MORE UNIMPORTANT

Sample (b).

"Not" restrictions with two variables (V_4 and V_3) where V_4 has values of k = 2 and 3 and V_3 has values of k = 1.

THE FOLLOWING RESTRICTIONS ARE NOT WANTED:

	VARIABLE:	VALUE(S):
user prepared these cards	CORE LEVEL	MORE IMPORTANT, MORE UNIMPORTANT
	TAXONOMY	FACTUAL

EXAMPLE 8

Retrieval by Identification Numbers

Description.

If the user knows exactly which items he wants retrieved and if he can supply their identification numbers then he should prepare his profile in the following manner.

(1) Col. 1 - 5 of Card 1 (Parameter Card) contains the number of ID's being submitted (maximum is 250); and if labels are being used col. 40 should have a 1 punched in it, otherwise leave blank; col. 45 must have a 1 punched in it to indicate items are being retrieved by ID's alone.

(2) Card(s) 2 (Variable Card(s)) should contain a list of identification numbers of items wanted; the number of ID's submitted must agree with the value specified in col. 1 - 5 of Card 1 (Parameter Card); if necessary use more cards, but continue to specify the ID's in fields of five.

(3) Cards 3 and 5 are optional (label cards), and if used may describe the items being selected by ID's; note that two label cards must be submitted if labels are being used.

Sample Profile for Retrieving by ID's without Labels

Parameter Card	3 (number of items wanted)			1 (retrieving by ID)
Variable Card	176	1276	43	

That is, user is retrieving three items with ID's of 176, 1276, and 43.

Sample Profile for Retrieving by ID's with Labels

Parameter Card	3 (number of items wanted)	1 (using labels)	1 (retrieving by ID's)
Variable Card	176 1276 43		
Variable Label	(give appropriate name(s))		
Value Label	(give appropriate name(s))		

That is, user is retrieving three items with ID's of 176, 1276, and 43, and is including some labels to remind him of how selection is being made.

ACKNOWLEDGEMENTS

Acknowledgement is made of Dr. S. M. Hunka (Director of the Division of Eduational Research Services, University of Alberta) for his kind assistance in the preparation of this documentation and of Mr. R. James (senior systems analyst for the University of Alberta's Computing Centre) for his valuable assistance in setting up skeleton records for random direct access files.

The writer is indebted to the staff of R. S. McLaughlin Examination and Research Centre and to his colleagues Dr. D. P. Flathman and Mr. D. J. Precht for their support and ideas in the development of the Medsirch system.