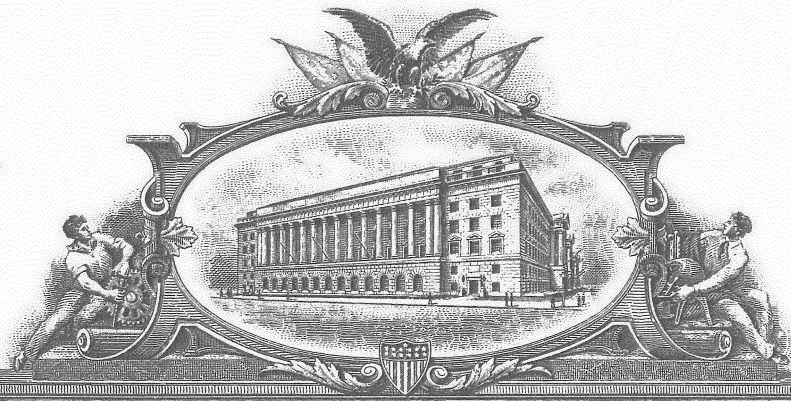


EXHIBIT 5

PART 1 OF 6

IW 7197768



THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

August 27, 2009

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE
RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS
OF:

APPLICATION NUMBER: 08/775,864

FILING DATE: *January 02, 1997*

PATENT NUMBER: 5,893,120

ISSUE DATE: *April 06, 1999*

By Authority of the

Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office



W. Montgomery
W. MONTGOMERY
Certifying Officer

BTEX0000174

707 806
 Class
 ISSUE CLASSIFICATION

KNB

5893120



UTILITY SERIAL NUMBER 08/775 864	PATENT DATE APR 06 1999	PATENT NUMBER
--	-----------------------------------	---------------

SERIAL NUMBER 08/775.864	FILING DATE 01/02/97	CLASS 389 701	SUBCLASS 616/622	GROUP ART UNIT 2387	EXAMINER Alam
-----------------------------	-------------------------	------------------	---------------------	------------------------	------------------

APPLICANTS RICHARD M. NEMES, BROOKLYN, NY.

CONTINUING DATA
 VERIFIED

VERIFIED (M)

FOREIGN/PCT APPLICATIONS
 VERIFIED

(M)

FOREIGN FILING LICENSE GRANTED 03/01/97

***** SMALL ENTITY *****

Foreign priority claimed 35 USC 119 conditions met	<input type="checkbox"/> yes <input checked="" type="checkbox"/> no	AS FILED →	STATE OR COUNTRY NY	SHEETS DRWGS. 6	TOTAL CLAIMS 8	INDEP. CLAIMS 1	FILING FEE RECEIVED \$435.00	ATTORNEY'S DOCKET NO.
Verified and Acknowledged	Examiner's Initials M							

ADDRESS RICHARD MICHAEL NEMES
 BROOKLYN NY

2821 Kings Highway, Apartment 1A
 11229-1835

ISSUE FEE IN FILE

TITLE METHODS AND APPARATUS FOR INFORMATION STORAGE AND RETRIEVAL USING A HASHING TECHNIQUE WITH EXTERNAL CHAINING AND ON-THE-FLY REMOVAL OF EXPIRED DATA

U.S. DEPT. OF COMM./PAT. & TM—PTO-436L (Rev.12-94)


PARTS OF APPLICATION FILED SEPARATELY		Michele Applications Examiner	
NOTICE OF ALLOWANCE MAILED 9.29.98		CLAIMS ALLOWED Total Claims: 8, Print Claim: 1	
ISSUE FEE Amount Due: 660.00, Date Paid: 12-10-98		DRAWING Sheets Drwg.: 6, Figs. Drwg.: 7, Print Fig.: 3	
Label Area		THOMAS G. BLACK SUPERVISORY PATENT EXAMINER GROUP 2700 Primary Examiner	
		PREPARED FOR ISSUE	
WARNING: The information disclosed herein may be restricted. Unauthorized disclosure may be prohibited by the United States Code Title 35, Sections 122, 181 and 368. Possession outside the U.S. Patent & Trademark Office is restricted to authorized employees and contractors only.			

Form PTO-436A (Rev. 8/92)

8

(FACE)

BTEX0000175

BAR CODE LABEL 	<h1>U.S. PATENT APPLICATION</h1>
---	----------------------------------

SERIAL NUMBER 08/775,864	FILING DATE 01/02/97	CLASS 395	GROUP ART UNIT 2307
-----------------------------	-------------------------	--------------	------------------------

APPLICANT	RICHARD M. NEMES, BROOKLYN, NY. **CONTINUING DATA***** VERIFIED <hr style="width: 10%; margin-left: 0;"/> **FOREIGN/PCT APPLICATIONS***** VERIFIED <hr style="width: 10%; margin-left: 0;"/> FOREIGN FILING LICENSE GRANTED 03/01/97 ***** SMALL ENTITY *****
-----------	---

STATE OR COUNTRY NY	SHEETS DRAWING 6	TOTAL CLAIMS 8	INDEPENDENT CLAIMS 4	FILING FEE RECEIVED \$425.00	ATTORNEY DOCKET NO.
------------------------	---------------------	-------------------	-------------------------	---------------------------------	---------------------

ADDRESS	RICHARD MICHAEL NEMES 1432 EAST 35TH STREET BROOKLYN NY 11234-2604
---------	--

TITLE	METHODS AND APPARATUS FOR INFORMATION STORAGE AND RETRIEVAL USING A HASHING TECHNIQUE WITH EXTERNAL CHAINING AND ON-THE-FLY REMOVAL OF EXPIRED DATA
-------	---

This is to certify that annexed hereto is a true copy from the records of the United States Patent and Trademark Office of the application which is identified above.

By authority of the
 COMMISSIONER OF PATENTS AND TRADEMARKS

Date _____ Certifying Officer _____

PATENT APPLICATION SERIAL NO. 08/775864

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

140 AA 02/28/97 08775864

1 201

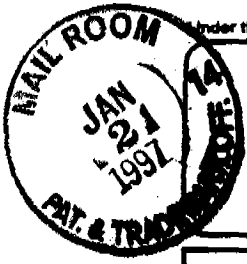
425.00 CK

PTO-1556
(5/87)

Please type a plus sign (+) inside the box →

Approved for use through 8/30/96. OMB 0851-0032
 Patent Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.



NEW UTILITY PATENT APPLICATION TRANSMITTAL

(to be used for new applications only)

Attorney Docket Number	08/775864
First Named Inventor	RICHARD M. NEMES
Total Pages in this Submission	<i>Application 22 PAGES + ABSTRACT + 6 sheets of DRAWINGS</i>

<p style="text-align: center;">APPLICATION ELEMENTS</p> <p><small>Notice: Checklist items mentioned under Application Elements section construct a new utility patent application. Please refer to MPEP Sections 506, 601, (37CFR 1.77, 1.53, 35 USC 111, 112, 113) for detailed explanation regarding completeness of an original patent application.</small></p>	<p style="text-align: center;">ACCOMPANYING APPLICATION PARTS</p>
---	--

- | | |
|--|---|
| <ol style="list-style-type: none"> 1. <input checked="" type="checkbox"/> Fee Transmittal Form (prescribed filing fee(s)) 2. Specification <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Title of the Invention <input type="checkbox"/> Cross References to Related Applications (if applicable) <input type="checkbox"/> Statement Regarding Federally-sponsored Research/Development (if applicable) <input type="checkbox"/> Reference to Microfiche Appendix (if applicable) <input checked="" type="checkbox"/> Background of the Invention <input checked="" type="checkbox"/> Brief Summary of the Invention <input checked="" type="checkbox"/> Brief Description of the Drawings (if drawings filed) <input checked="" type="checkbox"/> Detailed Description <input checked="" type="checkbox"/> Claim or Claims <input checked="" type="checkbox"/> Abstract of the Disclosure 3. <input checked="" type="checkbox"/> Drawing(s) (when necessary as prescribed by 35 USC 113) 4. <input checked="" type="checkbox"/> Executed Declaration 5. Genetic Sequence Submission (if applicable, all must be included) <ul style="list-style-type: none"> <input type="checkbox"/> Paper Copy <input type="checkbox"/> Computer Readable Copy <input type="checkbox"/> Statement Verifying Identical Paper and Computer Readable Copy | <ol style="list-style-type: none"> 6. <input type="checkbox"/> Assignment Papers 7. <input type="checkbox"/> Certified Copy of Priority Document(s) (if foreign priority is claimed) 8. <input type="checkbox"/> Computer Program in Microfiche 9. <input type="checkbox"/> English Translation Document (if applicable) 10. <input type="checkbox"/> Information Disclosure Statement/PTO-1449 <input type="checkbox"/> Copies of IDS Citations 11. <input type="checkbox"/> Petition Checklist and Accompanying Petition 12. <input type="checkbox"/> Preliminary Amendment 13. <input type="checkbox"/> Proprietary Information 14. <input checked="" type="checkbox"/> Return Receipt Postcard 15. <input checked="" type="checkbox"/> Small Entity Statement 16. <input checked="" type="checkbox"/> Additional Enclosures (please identify below): |
|--|---|

LIST OF REFERENCES

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm or Individual name	RICHARD M. NEMES
Signature	<i>Richard M. Nemes</i>
Date	DECEMBER 20, 1996

FOR OFFICIAL USE ONLY			
Application Number		Class	Independent Claims
Date of Receipt	Application Type	GAU	Total Claims
	Filing Date	Foreign Filing License?	Drawing Sheets
	Small Entity	Foreign Address?	Special Handling?

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.



FEE TRANSMITTAL

AMOUNT OF PAYMENT (\$) **425**

Class **to If Known**

Application Number **08/775**

Filing Date

First Named Inventor **RICHARD M. NEMES**

Group Art Unit

Examiner Name

Attorney Docket Number

METHOD OF PAYMENT (check one)

1. The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:

Deposit Account Number

Deposit Account Name

Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17

Charge the Issue Fee Set in 37 CFR 1.16 at the Making of the Notice of Allowance, 37 CFR 1.311(b)

2. Payment Enclosed:

Check Money Order Other

FEE CALCULATION (fees effective 10/01/96)

1. FILING FEE

Large Entity Code (\$)	Small Entity Code (\$)	Fee Description	Fee Paid
101 770	201 385	Utility filing fee	385
106 320	206 160	Design filing fee	
107 530	207 265	Plant filing fee	
108 770	208 385	Reissue filing fee	
114 150	214 75	Provisional filing fee	
SUBTOTAL (1)			(\$) 385

2. CLAIMS

Total Claims	Extra	Fee from below	Fee Paid
8	-20 = 0	X	
4	-3 = 1	X	40
Multiple Dependent Claims			

Large Entity Code (\$)	Small Entity Code (\$)	Fee Description	Fee Paid
103 22	203 11	Claims in excess of 20	
102 80	202 40	Independent claims in excess of 3	
104 260	204 130	Multiple dependent claim	
109 80	209 40	Reissue independent claims over original patent	
110 22	210 11	Reissue claims in excess of 20 and over original patent	
SUBTOTAL (2)			(\$) 40

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Code (\$)	Small Entity Code (\$)	Fee Description	Fee Paid
105 130	205 65	Surcharge - late filing fee or oath	
127 50	227 25	Surcharge - late provisional filing fee or cover sheet	
139 130	139 130	Non-English specification	
147 2,460	147 2,460	For filing a request for reexamination	
112 900	112 900	Requesting publication of SIR prior to Examiner action	
113 1,790	113 1,790	Requesting publication of SIR after Examiner action	
115 110	215 55	Extension for response within first month	
116 380	216 195	Extension for response within second month	
117 830	217 465	Extension for response within third month	
118 1,470	218 735	Extension for response within fourth month	
119 300	219 150	Notice of Appeal	
120 300	220 150	Filing a brief in support of an appeal	
121 280	221 130	Request for oral hearing	
138 1,470	138 1,470	Petition to institute a public use proceeding	
140 110	240 55	Petition to revive unvoidably abandoned application	
141 1,290	241 645	Petition to revive unintentionally abandoned application	
142 1,290	242 645	Utility issue fee (or reissue)	
143 440	243 220	Design issue fee	
144 650	244 325	Plant issue fee	
122 130	122 130	Petitions to the Commissioner	
123 50	123 50	Petitions related to provisional applications	
126 230	126 230	Submission of Information Disclosure Stmt	
581 40	581 40	Recording each patent assignment per property (times number of properties)	
146 770	246 385	Filing a submission after final rejection (37 CFR 1.129(a))	
149 770	249 385	For each additional invention to be examined (37 CFR 1.129(b))	
Other fee (specify)			
Other fee (specify)			
SUBTOTAL (3)			(\$)

* Reduced by Basic Filing Fee Paid

SUBMITTED BY

Typed or Printed Name	RICHARD M. NEMES		Complete (if applicable)	
Signature	<i>Richard M. Nemes</i>	Date	DEC. 20, 1996	Reg. Number
			Deposit Account	User ID

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents.



42500 201 A

08/775864

Patent Application of

Richard M. Nemes

1432 East 35th Street, Brooklyn, New York 11234-2604, U.S.A.

A Citizen of the United States of America

5

SPECIFICATION

10 TITLE OF INVENTION

METHODS AND APPARATUS FOR INFORMATION STORAGE AND
RETRIEVAL USING A HASHING TECHNIQUE WITH EXTERNAL CHAINING
AND ON-THE-FLY REMOVAL OF EXPIRED DATA

15 CROSS-REFERENCE TO RELATED APPLICATIONS

Not Applicable

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR
DEVELOPMENT

20 Not Applicable

REFERENCE TO A MICROFICHE APPENDIX

Not Applicable

25 BACKGROUND OF THE INVENTION

This invention relates to information storage and retrieval systems, and, more particularly, to the use of hashing techniques in such systems.

Information or data stored in a computer-controlled storage mechanism can be retrieved by searching for a particular key value in the stored records. The stored record with a key
30 matching the search key value is then retrieved. Such searching techniques require repeated access to records into the storage mechanism to perform key comparisons. In large storage and retrieval systems, such searching, even if augmented by efficient search procedures such as the

binary search, often requires an excessive amount of time due to the large number of key comparisons required.

Another well-known and much faster way of storing and retrieving information from computer storage, albeit at the expense of additional storage, is the so-called "hashing" technique, also called scatter-storage or key-transformation method. In such a system, the key is operated on by a hashing function to produce a storage address in the storage space, called the hash table, which is a large one-dimensional array of record locations. This storage address is then accessed directly for the desired record. Hashing techniques are described in the classic text by D. E. Knuth entitled *The Art of Computer Programming*, Volume 3, Sorting and Searching, Addison-Wesley, Reading, Massachusetts, 1973, pp. 506-549.

Hashing functions are designed to translate the universe of keys into addresses uniformly distributed throughout the hash table. Typical hashing functions include truncation, folding, transposition, and modulo arithmetic. A disadvantage of hashing is that more than one key will inevitably translate in the same storage address, causing "collisions" in storage. Some form of collision resolution must therefore be provided. For example, the simple strategy called "linear probing," which consists of searching forward from the initial storage address to the first empty storage location, is often used.

Another method for resolving collisions is called "external chaining." In this technique, each hash table location is a pointer to the head of a linked list of records, all of whose keys translate under the hashing function to that very hash table address. The linked list is itself searched sequentially when retrieving, inserting, or deleting a record. Insertion and deletion are done by adjusting pointers in the linked list. External chaining is discussed in considerable detail in the aforementioned text by D. E. Knuth, in *Data Structures and Program Design*, Second Edition, by R. L. Kruse, Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1987, Section 6.5, "Hashing," and Section 6.6, "Analysis of Hashing," pp. 198-215, and in *Data Structures with Abstract Data Types and Pascal*, by D. F. Stubbs and N. W. Webre, Brooks/Cole Publishing Company, Monterey, California, 1985, Section 7.4, "Hashed Implementations," pp. 310-336.

Some forms of information are such that individual data items, after a limited period of time, become obsolete, and their presence in the storage system is no longer needed or desired.

Scheduling activities, for example, involve data that become obsolete once the scheduled event has occurred. An automatically-expiring data item, once it expires, needlessly occupies computer memory storage that could otherwise be put to use storing an unexpired item. Thus, expired items must eventually be removed to reclaim the storage for subsequent data insertions. In addition, the presence of many expired items results in needlessly long search times since the linked lists associated with external chaining will be longer than they otherwise would be. The goal is to remove these expired items to reclaim the storage and maintain fast access to the data.

The problem, then, is to provide the speed of access of hashing techniques for large, heavily used information storage systems having expiring data and, at the same time, prevent the performance degradation resulting from the accumulation of many expired records. Although a hashing technique for dealing with expiring data is known and disclosed in U.S. Patent Number 5,121,495, issued June 9, 1992, that technique is confined to linear probing and is entirely inapplicable to external chaining. The procedure shown there traverses, in reverse order, a consecutive sequence of records residing in the hash table array, continually relocating unexpired records to fill gaps left by the removal of expired ones.

Unlike arrays, linked lists leave no gaps when items from it are removed, and furthermore it is not possible to efficiently traverse a singly linked list in reverse order. There are significant advantages to external chaining over linear probing that sometimes make it the method of choice, as discussed in considerable detail in the aforementioned texts, and so hashing techniques for dealing with expiring data that do not use external chaining prove wholly inadequate for certain applications. For example, if the data records are large, considerable memory can be saved using external chaining instead of linear probing. Accordingly, there is a need to develop hashing techniques for external chaining with expiring data. The methods of the above-mentioned patent are limited to arrays and cannot be used with linked lists due to the significant difference in the organization of the computer's memory.

BRIEF SUMMARY OF THE INVENTION

In accordance with the illustrative embodiment of the invention, these and other problems are overcome by using a garbage collection procedure "on-the-fly" while other types of access to the storage space are taking place. In particular, during normal data insertion or

4

storage system in accordance with the present invention; and

FIG. 7 shows a general flow chart for a record deletion operation that might be used in a hashed storage system in accordance with the present invention.

To facilitate reader understanding, identical reference numerals are used to designate
5 elements common to the figures.

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 of the drawings shows a general block diagram of a computer hardware system comprising a Central Processing Unit (CPU) 10 and a Random Access Memory (RAM) unit 11.
10 Computer programs stored in the RAM 11 are accessed by CPU 10 and executed, one instruction at a time, by CPU 10. Data, stored in other portions of RAM 11, are operated on by the program instructions accessed by CPU 10 from RAM 11, all in accordance with well-known data processing techniques.

Central Processing Unit (CPU) 10 also controls and accesses a disk controller unit 12
15 that, in turn, accesses a digital data stored on one or more disk storage units such as disk storage unit 13 until required by CPU 10. At this time, such programs and data are retrieved from disk storage unit 13 in blocks and stored in RAM 11 for rapid access.

Central Processing Unit (CPU) 10 also controls an Input/Output (I/O) controller 14 that, in turn, provides access to a plurality of input devices such as CRT (cathode ray tube) terminal
20 15, as well as a plurality of output devices such as printer 16. Terminal 15 provides a mechanism for a computer user to introduce instructions and commands into the computer system of FIG. 1, and may be supplemented with other input devices such as magnetic tape readers, remotely located terminals, optical readers, and other types of input devices. Similarly, printer 16 provides a mechanism for displaying the results of the operation of the computer system of FIG. 1 for the
25 computer user. Printer 16 may similarly be supplemented by line printers, cathode ray tube displays, phototypesetters, laser printers, graphical plotters, and other types of output devices.

The constituents of the computer system of FIG. 1 and their cooperative operation are well-known in the art and are typical of all computer systems, from small personal computers to large mainframe systems. The architecture and operation of such systems are well-known and
30 will not be further described here.

10

FIG. 2 shows a graphical representation of a typical software architecture for a computer system such as that shown in FIG. 1. The software of FIG. 2 comprises a user access mechanism 20 that, for simple personal computers, may consist of nothing more than turning the system on. In larger systems, providing service to many users, login and password procedures would typically be implemented in user access mechanism 20. Once user access mechanism 20 has completed the login procedure, the user is placed in the operating system environment 21. Operating system 21 coordinates the activities of all of the hardware components of the computer system (shown in FIG. 1) and provides a number of utility programs 22 of general use to the computer user. Utilities 22 might, for example, comprise basic file access and manipulation programs, system maintenance facilities, and programming language compilers.

The computer software system of FIG. 2 typically also includes application programs such as application software 23, 24, . . . , 25. Application software 23 through 25 might, for example, comprise a text editor, document formatting software, a spreadsheet program, a database management system, a game program, and so forth.

The present invention is concerned with information storage and retrieval. It can be application software packages 23-25, or used by other parts of the system, such as user access software 20 or operating system 21 software. The information storage and retrieval technique provided by the present invention are herein disclosed as flowcharts in FIGS. 3 through 7, and shown as PASCAL-like pseudocode in the APPENDIX to this specification.

Before proceeding to a description of one embodiment of the present invention, it is first useful to discuss hashing techniques in general. Many fast techniques for storing and retrieving data are known in the prior art. In situations where storage space is considered cheap compared with retrieval time, a technique called hashing is often used. In classic hashing, each record in the information storage system includes a distinguished field unique in value to each record, called the key, which is used as the basis for storing and retrieving the associated record. Taken as a whole, a hash table is a large, one-dimensional array of logically contiguous, consecutively numbered, fixed-size storage units. Such a table of records is typically stored in RAM 11 of FIG. 1, where each record is an identifiable and addressable location in physical memory. A hashing function translates the key into a hash table array subscript, which is used as an index into the array where searches for the data record begin. The hashing function can be any operation on

the key that results in subscripts mostly uniformly distributed across the table. Known hashing functions include truncation, folding, transposition, modulo arithmetic, and combinations of these operations. Unfortunately, hashing functions generally do not produce unique locations in the hash table, in that many distinct keys map to the same location, producing what are called
5 collisions. Some form of collision resolution is required in all hashing systems. In every occurrence of collision, finding an alternate location for a collided record is necessary. Moreover, the alternate location must be readily reachable during future searches for the displaced record.

A common collision resolution strategy, with which the present invention is concerned,
10 is called external chaining. Under external chaining, each hash table entry stores all of the records that collided at that location by storing not the records themselves, but instead a pointer to the head of a linked list of those same records. Such linked lists are formed by storing the records individually in dynamically allocated storage and maintaining with each record a pointer to the location of the next record in the chain of collided records. When a search key is hashed
15 to a hash table entry, the pointer found there is used to locate the first record. If the search key does not match the key found there, the pointer there is used to locate the second record. In this way, the "chain" of records is traversed sequentially until the desired record is found or until the end of the chain is reached. Deletion of records involves merely adjusting the pointers to bypass the deleted record and returning the storage it occupied to the available storage pool maintained
20 by the system.

Hashing techniques have been used classically for very fast access to static, short term data such as a compiler symbol table. Typically, in such storage systems, deletions are infrequent and the need for the storage system disappears quickly. In some common types of data storage systems, however, the storage system is long lived and records can become obsolete merely by
25 the passage of time or by the occurrence of some event. If such expired, lapsed, or obsolete records are not removed from the system, they will, in time, seriously degrade the performance of the retrieval system. Degradation shows up in two ways. First, the presence of expired records lengthens search times since they cause the external chains to be longer than they otherwise would be. Second, expired records occupy dynamically allocated memory storage that
30 could be returned to the system memory pool for useful allocation. Thus, when the system

8

memory pool is depleted, a new data item can be inserted into the storage system only if the memory occupied by an expired one is reclaimed.

Referring then to FIG. 3, there is shown a flowchart of a *search table* procedure for searching the hash table preparatory to inserting, retrieving, or deleting a record, in accordance with the present invention, and involving the dynamic removal of expired records in a targeted linked list. Starting in box 30 of the *search table* procedure of FIG. 3, the search key of the record being searched for is hashed in box 31 to provide the subscript of an array element. In box 32, the hash table array location indicated by the subscript generated in box 31 is accessed to provide the pointer to the target linked list. Decision box 33 examines the pointer value to determine whether the end of the linked list has been reached. If the end has been reached, decision box 34 is entered to determine if a key match was previously found in decision box 39 (as will be described below). If so, the search is successful and returns success in box 35, followed by the procedure's termination in terminal box 37. If not, box 36 is entered where failure is returned and the procedure again terminates in box 37.

If the end of the list has not been reached as determined by decision box 33, decision box 38 is entered to determine if the record pointed to has expired. This is determined by comparing some portion of the contents of the record to some external condition. A timestamp in the record, for example, could be compared with the current time-of-day value maintained by all computers. Alternatively, the occurrence of an event can be compared with a field identifying that event in the record. In any case, if the record has not expired, decision box 39 is entered to determine if the key in this record matches the search key. If it does, the address of the record is saved in box 40 and box 41 is entered. If the record does not match the search key, the procedure bypasses box 40 and proceeds directly to box 41. In box 41, the procedure advances forward to the next record in the linked list and the procedure returns to box 33.

If decision box 38 determines that the record under question has expired, box 42 is entered to perform the on-the-fly removal of the expired record from the linked list and the return of the storage it occupies to the system storage pool, as will be described in connection with FIG. 4. In general, the *remove* procedure of box 42 (FIG. 4) operates to remove an element from the linked list by adjusting its predecessor's pointer to bypass that element. (However, if the element to be removed is the first element of the list, then there is no predecessor and the

9

hash table array entry is adjusted instead.) On completion of procedure *remove* invoked from box 42, the *search table* procedure returns to box 33.

It can be seen that the *search table* procedure of FIG. 3 operates to examine the entire linked list of records of which the searched-for record is a part, and to remove expired records, returning storage to the storage pool with each removal. If the storage pool is depleted and many expired records remain despite such automatic garbage collection, then the insertion of new records is inhibited (boxes 76 and 77 of FIG. 5) until a deletion is made by the *delete* procedure (FIG. 7) or until the *search table* procedure has had a chance to replenish the storage pool through its on-the-fly garbage collection.

Though the *search table* procedure as shown in FIG. 3, implemented in the APPENDIX as PASCAL-like pseudocode, and described above appears in connection with an information storage and retrieval system using the hashing technique with external chaining, its on-the-fly removal technique while traversing a linked list can be used anywhere a linked list of records with expiring data appears, even in contexts unrelated to hashing. A person skilled in the art will appreciate that this technique can be readily applied to manipulate linked lists not necessarily used with hashing.

The *search table* procedure shown in FIG. 3, implemented as pseudocode in the APPENDIX, and described above traverses the entire linked list removing all expired records as it searches for a key match. The procedure can be readily adapted to remove some but not all of the expired records, thereby shortening the linked list traversal time and speeding up the search at the expense of perhaps leaving some expired records in the list. For example, the procedure can be modified to terminate when a key match occurs. (PASCAL-like pseudocode for this alternate version of *search table* appears in the APPENDIX.) The implementor even has the prerogative of choosing among these strategies dynamically at the time *search table* is invoked by the caller, thus sometimes removing all expired records, at other times removing some but not all of them, and yet at other times choosing to remove none of them. Such a dynamic runtime decision might be based on factors such as, for example, how much memory is available in the system storage pool, general system load, time of day, the number of records currently residing in the information system, and other factors both internal and external to the information storage and retrieval system itself. A person skilled in the art will appreciate that the

technique of removing all expired records while searching the linked list can be expanded to include techniques whereby not necessarily all expired records are removed, and that the decision regarding if and how many records to delete can be a dynamic one.

In FIG. 4 there is shown a flowchart of a *remove* procedure that removes a record from
5 the retrieval system, either an unexpired record through the *delete* procedure as will be noted in connection with FIG. 7, or an expired record through the *search table* procedure as noted in connection with FIG. 3. In general, this is accomplished by the invoking procedure, being either the *delete* procedure (FIG. 7) or the *search table* procedure (FIG. 3), passing to the *remove* procedure a pointer to a linked list element to remove, a pointer to that element's predecessor
10 element in the same linked list, and the subscript of the hash table array location containing the pointer to the head of the linked list from which the element is to be removed. In the case that the element to be removed is the first element of the linked list, the predecessor pointer passed to the *remove* procedure by the invoking procedure has the NIL (sometimes called NULL, or EMPTY) value, indicating to the *remove* procedure that the element to be removed has no
15 predecessor in the list. The invoking procedure expects the *remove* procedure, on completion, to have advanced the passed pointer that originally pointed to the now-removed element so that it points to the successor element in that linked list, or NIL if the removed element was the final element. (The *search table* procedure of FIG. 3, in particular, makes use of the *remove* procedure's advancing this passed pointer in the described way; it is made use of in that box 33
20 of FIG. 3 is entered directly following completion of box 42, as was described above in connection with FIG. 3.)

The *remove* procedure causes actual removal of the designated element by adjusting the predecessor pointer so that it bypasses the element to be removed. In the case that the predecessor pointer has the NIL value, the hash table array entry indicated by the passed
25 subscript plays the role of the predecessor pointer and is adjusted the same way in its stead. Following pointer adjustments, the storage occupied by the removed element is returned to the system storage pool for future allocation.

Beginning, then, at starting box 50 of FIG. 4, the pointer to the list element to remove is advanced in box 51 so that it points to its successor in the linked list. Next, decision box 52
30 determines if the element to remove is the first element in the containing linked list by testing

the predecessor pointer for the NIL value, as described above. If so, box 54 is entered to adjust the linked list head pointer in the hash table array to bypass the first element, after which the procedure continues on to box 55. If not, box 53 is entered where the predecessor pointer is adjusted to bypass the element to remove, after which the procedure proceeds, once again, to box 55. Finally, in box 55 the storage occupied by the bypassed element is returned to the system storage pool and the procedure terminates in terminal box 56.

Fig. 5 shows a detailed flowchart of an *insert* procedure suitable for use in the information storage and retrieval system of the present invention. The *insert* procedure of FIG. 5 begins at starting box 70 from which box 71 is entered. In box 71, the *search table* procedure of FIG. 3 is invoked with the search key of the record to be inserted. As noted in connection with FIG. 3, the *search table* procedure finds the linked list element whose key value of the record contained therein matches the search key and, at the same time, removes expired records on-the-fly from that linked list. Decision box 72 is then entered where it is determined whether the *search table* procedure found a record with matching key value. If so, box 73 is entered where the record to be inserted is put into the linked list element in the position of the old record with matching key value. In box 74, the *insert* procedure reports that the old record has been replaced by the new record and the procedure terminates in terminal box 75.

Returning to decision box 72, if a matching record is not found, decision box 76 is entered to determine if there is sufficient storage in the system storage pool to accommodate a new linked list element. If not, box 77 is entered to report that the storage system is full and the record cannot be inserted. Following that, the procedure terminates in terminal box 75.

If decision box 76 determines that sufficient storage can be allocated from the system storage pool for a new linked list element, then box 78 is entered where the actual memory allocation is made. In box 79, the record to be inserted is copied into the storage allocated in box 78, and box 80 is entered. In box 80, the linked list element containing the record copied into it in box 79 is inserted into the linked list to which the contained record hashed. The procedure then reports that the record was inserted into the information storage and retrieval system in box 81 and the procedure terminates in box 75.

FIG. 6 shows a detailed flowchart of a *retrieve* procedure used to retrieve a record from the information storage and retrieval system. Starting in box 90, the *search table* procedure of

FIG. 3 is invoked in box 91, using the key of the record to be retrieved as the search key. In decision box 92 it is determined if a record with a matching key was found by the *search table* procedure. If not, box 93 is entered to report failure of the *retrieve* procedure, and the procedure terminates in terminal box 96. If a matching record was found, box 94 is entered to copy the matching record into a record store for processing by the calling program, box 95 is entered to return an indication of successful retrieval, and the procedure terminates in terminal box 96.

FIG. 7 shows a detailed flowchart of a *delete* procedure useful for actively removing records from the information storage and retrieval system. Starting at box 100, the procedure of FIG. 7 invokes the *search table* procedure of FIG. 3 in box 101, using the key of the record to be deleted as the search key. In decision box 102, it is determined if the *search table* procedure was able to find a record with matching key. If not, box 103 is entered to report failure of the deletion procedure, and the procedure terminates in terminal box 106. If a matching record was found, as determined by decision box 102, the *remove* procedure of FIG. 4 is invoked in box 104. As noted in connection with FIG. 4, the *remove* procedure causes removal of a designated linked list element from its containing linked list. Box 105 is then entered to report successful deletion to the calling program, and the procedure terminates in terminal box 106.

The attached APPENDIX contains PASCAL-like pseudocode listings for all of the programmed components necessary to implement an information storage and retrieval system operating in accordance with the present invention. Any person of ordinary skill in the art will have no difficulty implementing the disclosed system and procedures shown in the APPENDIX, including programs for all common hardware and system software arrangements, on the basis of this description, including flowcharts and information shown in the APPENDIX.

It should also be clear to those skilled in the art that other embodiments of the present invention may be made by those skilled in the art without departing from the teachings of the present invention. It is also clear to those skilled in the art that the invention can be used in diverse computer applications, and that it is not limited to the use of hash tables, but is applicable to other techniques requiring linked lists with expiring records.

Appendix

Functions Provided

5 The following functions are made available to the application program:

1. *insert (record: record_type)*

10 Returns *replaced* if a record associated with *record.key* was found and subsequently replaced.

Returns *inserted* if a record associated with *record.key* was not found and the passed record was subsequently inserted.

15 Returns *full* if a record associated with *record.key* was not found and the passed record could not be inserted because no memory is available.

20 2. *retrieve (record: record_type)*

Returns *success* if record associated with *record.key* was found and assigned to *record*.

25 Returns *failure* if search was unsuccessful.

3. *delete (record_key: record_key_type)*

30 Returns *success* if record associated with *record_key* was found and subsequently deleted.

Returns *failure* if not found.

35

Definitions

The following formal definitions are required for specifying the insertion, retrieval, and deletion procedures. They are global to all procedures and functions shown below.

40

```

1. const table_size                                     /* Size of hash table. */

2. type list_element_pointer =  $\uparrow$ list_element         /* Pointer to elements of linked list. */

5 3. type list_element =                                  /* Each element of linked list. */

    record
        record_contents: record_type;
        next: list_element_pointer                       /* Singly-linked list. */
10 end

4. var table: array [0 .. table_size - 1] of list_element_pointer /* Hash table. */
    /* Each array entry is pointer to head of list. */

15 Initial state of table: table[i] = nil  $\forall i$  0  $\leq$  i < table_size /* Initially empty. */

```

Insert Procedure

```

20 function insert (record: record_type): (replaced, inserted, full);

    var position: list_element_pointer;                  /* Pointer into list of found record, */
    /* or new element if not found. */

25 dummy_pointer: list_element_pointer;                /* Don't need position's predecessor. */

    index: 0 .. table_size - 1;                        /* Table index mapped to by hash function. */

    begin

30 if search_table (record.key, position, dummy_pointer, index) /* Record already exist? */

        then begin                                       /* Yes, update it with passed record. */

35 position\ record_contents := record;

        return (replaced)

        end

40 else                                                   /* No, insert new record at head of list, */

```

LE

```

    if no memory available then return (full)           /* if memory available to do so. */
                                                       /* Memory is available for a node. */
    else begin
    5         new(position);                             /* Dynamically allocate new node. */
                                                       /* Hook it in. */
        position^.record_contents := record;
        position^.next := table[index];
    10        table[index] := position;
        return (inserted)
    15    end                                           /* else begin */
end                                                 /* insert */
```

20

Retrieve Procedure

```

function retrieve (var record: record_type): (success, failure);
    25  var position: list_element_pointer;           /* Pointer into list of found record. */
        dummy_pointer: list_element_pointer;       /* Don't need position's predecessor. */
        dummy_index: 0 .. table_size - 1;         /* Don't need table index mapped to by hash function. */
    30  begin
        if search_table (record.key, position, dummy_pointer, dummy_index) /* Record exist? */
    35  then begin                                     /* Yes, return it to caller. */
            record := position^.record_contents;
            return (success)
    40  end
end
```

110

else return (*failure*) /* No, report failure. */

end /* retrieve */

5

Delete Procedure

function *delete* (*record_key*: *record_key_type*): (*success*, *failure*);

10

var *position*: *list_element_pointer*; /* Pointer into list of found record. */

previous_position: *list_element_pointer*; /* Points to *position*'s predecessor. */

15

index: 0 .. *table_size* - 1; /* Table index mapped to by hash function. */

begin

20

if *search_table* (*record_key*, *position*, *previous_position*, *index*) /* Record exist? */

then begin /* Yes, remove it. */

remove (*position*, *previous_position*, *index*);

25

return (*success*)

end

30

else return (*failure*) /* No, report failure. */

end /* delete */

35

Search Table Procedure

function *search_table* (*record_key*: *record_key_type*;

var *position*: *list_element_pointer*;

var *previous_position*: *list_element_pointer*;

40

var *index*: 0 .. *table_size* - 1): *boolean*;

17

```

/* Search table for record_key and delete expired records in target list; if found, position is made to
point to located record and previous_position to its predecessor, and TRUE is returned; otherwise
FALSE is returned. Index is set to table subscript that is mapped to by hash function in either
case.*/

5
var p: list_element_pointer; /* Used for traversing chain. */

    previous_p: list_element_pointer; /* Points to p's predecessor. */

10 begin

    index := hash (record_key); /* hash returns value in the range 0 .. table_size - 1. */

    p := table[index]; /* Initialization before loop. */

15    previous_p := nil; /* Ditto */

    position := nil; /* Ditto */

20    previous_position := nil; /* Ditto */

    while p ≠ nil /* HEART OF THE TECHNIQUE: Traverse entire list, deleting */
                    /* expired records as we search. */
    begin

25        if p↑.record_contents is expired

            then remove (p, previous_p, index) /* ON-THE-FLY REMOVAL OF EXPIRED RECORD! */

30        else begin

            if position = nil then if p↑.record_contents.key = record_key
                                        /* If this is record wanted,*/
                then begin position := p; previous_position := previous_p end;
                                        /* save its position. */
35            previous_p := p; /* Advance to */

                p := p↑.next /* next record. */

40        end /* else begin */

    end;

```

```

return (position ≠ nil)           /* Return TRUE if record located, otherwise FALSE. */

```

```

end                               /* search_table */

```

5

Alternate Version of Search Table Procedure

```

function search_table (record_key: record_key_type;
10     var position: list_element_pointer;
     var previous_position: list_element_pointer;
     var index: 0 .. table_size - 1): boolean;

15  /* SAME AS VERSION SHOWN ABOVE EXCEPT THAT THE SEARCH TERMINATES IF
     RECORD IS FOUND, INSTEAD OF ALWAYS TRAVERSING THE ENTIRE CHAIN. */

     var p: list_element_pointer;           /* Used for traversing chain. */

     previous_p: list_element_pointer;     /* Points to p's predecessor. */

20  begin

     index := hash (record_key);           /* hash returns value in the range 0 .. table_size - 1. */

25  p := table[index];                     /* Initialization before loop. */

     previous_p := nil;                     /* Ditto */

     position := nil;                       /* Ditto */

30  previous_position := nil;              /* Ditto */

     while p ≠ nil                          /* HEART OF THE TECHNIQUE: Traverse list, deleting */
                                           /* expired records as we search. */

35  begin

     if p↑.record_contents is expired

     then remove (p, previous_p, index) /* ON-THE-FLY REMOVAL OF EXPIRED RECORD! */

40  else begin

```


Richard M. Nemes

```

    if p↑.record_contents.key = record_key           /* If this is record wanted,*/
        then begin                                   /* save its position. */
            5      position := p;
                previous_position := previous_p;
                return (true)                       /* We found the record, so terminate search. */
        10     end;
            previous_p := p;                         /* Advance to */
            15     p := p↑.next                       /* next record. */
        end                                           /* else begin */
    end;
    20     return (false)                             /* Record not found. */
end                                                 /* search_table */
25
```

Remove Procedure

```

procedure remove (var elem_to_del: list_element_pointer;
30     previous_elem: list_element_pointer;
        index: 0 .. table_size - 1);

    /* Delete elem_to_del↑ from list, advancing elem_to_del to next element. previous_elem points to
        elem_to_del's predecessor, or nil if elem_to_del↑ is 1st element in list. */
    35     var p: list_element_pointer;               /* Save pointer to elem_to_del for disposal. */

    begin
        40     p := elem_to_del;                     /* Save so we can dispose when finished adjusting pointers. */

```

Richard M. Nemes

```
    elem_to_del := elem_to_del\ .next;

    if previous_elem = nil                                /* Deleting 1* element requires changing */
    then table[index] := elem_to_del                    /* head pointer, as opposed to */
    else previous_elem\ .next := elem_to_del;          /* predecessor's next pointer. */
    dispose (p)                                         /* Dynamically de-allocate node. */
10 end                                                 /* remove */
```

CLAIMS

I claim:

1. An information storage and retrieval system, the system comprising:
 - a linked list to store and provide access to records stored in a memory of the system, at
 - 5 least some of the records automatically expiring,
 - a record search means utilizing a search key to access the linked list,
 - the record search means including a means for identifying and removing at least some
 - of the expired ones of the records from the linked list when the linked list is accessed, and
 - means, utilizing the record search means, for accessing the linked list and, at the same
 - 10 time, removing at least some of the expired ones of the records in the linked list.
2. The information storage and retrieval system according to claim 1 further including means for dynamically determining maximum number for the record search means to remove in the accessed linked list of records.
- 15 3. A method for storing and retrieving information records using a linked list to store and provide access to the records, at least some of the records automatically expiring, the method comprising the steps of:
 - accessing the linked list of records,
 - identifying at least some of the automatically expired ones of the records, and
 - 20 removing at least some of the automatically expired records from the linked list when the linked list is accessed.
4. The method according to claim 3 further including the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed.
- 25 5. An information storage and retrieval system, the system comprising:
 - a hashing means to provide access to records stored in a memory of the system and using an external chaining technique to store the records with same hash address, at least some of
 - the records automatically expiring,
 - 30 a record search means utilizing a search key to access a linked list of records having the

5/7

same hash address,

the record search means including means for identifying and removing at least some expired ones of the records from the linked list of records when the linked list is accessed, and

5 means, utilizing the record search means, for inserting, retrieving, and deleting records from the system and, at the same time, removing at least some expired ones of the records in the accessed linked list of records.

6. The information storage and retrieval system according to claim 5 further including means for dynamically determining maximum number for the record search means to
10 remove in the accessed linked list of records.

7. A method for storing and retrieving information records using a hashing technique to provide access to the records and using an external chaining technique to store the records with same hash address, at least some of the records automatically expiring, the method comprising the
15 steps of:

accessing a linked list of records having same hash address,
identifying at least some of the automatically expired ones of the records,
removing at least some of the automatically expired records from the linked list when
the linked list is accessed, and

20 inserting, retrieving or deleting one of the records from the system following the step of removing.

8. The method according to claim 7 further including the step of dynamically determining maximum number of expired ones of the records to remove when the linked list is accessed.

775864



Richard M. Nemes

ABSTRACT OF THE DISCLOSURE

A method and apparatus for performing storage and retrieval in an information storage system is disclosed that uses the hashing technique with the external chaining method for collision resolution. In order to prevent performance deterioration due to the presence of
5 automatically expiring data items, a garbage collection technique is used that removes all expired records stored in the system in the external chain targeted by a probe into the data storage system. More particularly, each insertion, retrieval, or deletion of a record is an occasion to search an entire linked-list chain of records for expired items and then remove them. Because
10 an expired data item will not remain in the system long term if the system is frequently probed, it is useful for large information storage systems that are heavily used, require the fast access provided by hashing, and cannot be taken off-line for removal of expired data.

**IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE**

Declaration

As the below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled **METHODS AND APPARATUS FOR INFORMATION STORAGE AND RETRIEVAL USING A HASHING TECHNIQUE WITH EXTERNAL CHAINING AND ON-THE-FLY REMOVAL OF EXPIRED DATA** the specification of which is attached hereto.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims and any amendments filed therewith.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37 Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code § 119 (a)-(d) or § 365 (b) of any foreign application(s) for patent or inventor's certificate, or § 365 (a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below any foreign application for patent or inventor's certificate, or of any PCT international application having a filing date before that of the application on which priority is claimed:

None

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s), or § 365 (c) of any PCT international application designating the United States of America, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT international application in the manner provided by the first paragraph of Title 35, United States Code § 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations § 1.56 which became available between

the filing date of the prior application and the national or PCT international filing date of this application:

None

Direct all correspondence to:

Richard Michael Nemes
1432 East 35th Street
Brooklyn, New York 11234-2604
U.S.A.

Telephone: (718) 377-5438

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

100
Full name of sole inventor: Richard Michael Nemes

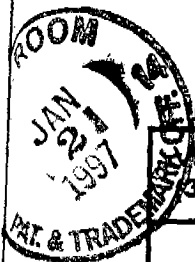
Inventor's signature: Richard Michael Nemes Date: Dec. 20, 1996

Residence: Brooklyn, Kings County, New York NY

Citizenship: United States of America

Post Office Address: 1432 East 35th Street
Brooklyn, New York 11234-2604
U.S.A.





Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

**VERIFIED STATEMENT CLAIMING SMALL ENTITY STATUS
37 CFR 1.9(f) & 1.27(b))--INDEPENDENT INVENTOR**

Docket Number (Optional)

Applicant or Patentee: RICHARD M. NEMES

Application or Patent No.: _____

Filed or Issued: _____

Title: METHODS AND APPARATUS FOR INFORMATION STORAGE AND RETRIEVAL
USING A HASHING TECHNIQUE WITH EXTERNAL CHAINING AND ON-THE-
FLY REMOVAL OF EXPIRED DATA

As a below named inventor, I hereby declare that I qualify as an independent inventor as defined in 37 CFR 1.9(c) for purposes of paying reduced fees to the Patent and Trademark Office described in:

- the specification filed herewith with title as listed above.
- the application identified above.
- the patent identified above.

I have not assigned, granted, conveyed, or licensed, and am under no obligation under contract or law to assign, grant, convey, or license, any rights in the invention to any person who would not qualify as an independent inventor under 37 CFR 1.9(c) if that person had made the invention, or to any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

Each person, concern, or organization to which I have assigned, granted, conveyed, or licensed or am under an obligation under contract or law to assign, grant, convey, or license any rights in the invention is listed below:

- No such person, concern, or organization exists.
- Each such person, concern, or organization is listed below.

Separate verified statements are required from each named person, concern, or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27)

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b))

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

<u>RICHARD M. NEMES</u>	_____	_____
NAME OF INVENTOR	NAME OF INVENTOR	NAME OF INVENTOR
<u>Richard M. Nemes</u>	_____	_____
Signature of inventor	Signature of inventor	Signature of inventor
<u>DEC. 20, 1996</u>	_____	_____
Date	Date	Date

Burden Hour Statement: This form is estimated to take 0.5 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

Richard M. Nemes

Sheet 1 of 6

FIG. 1

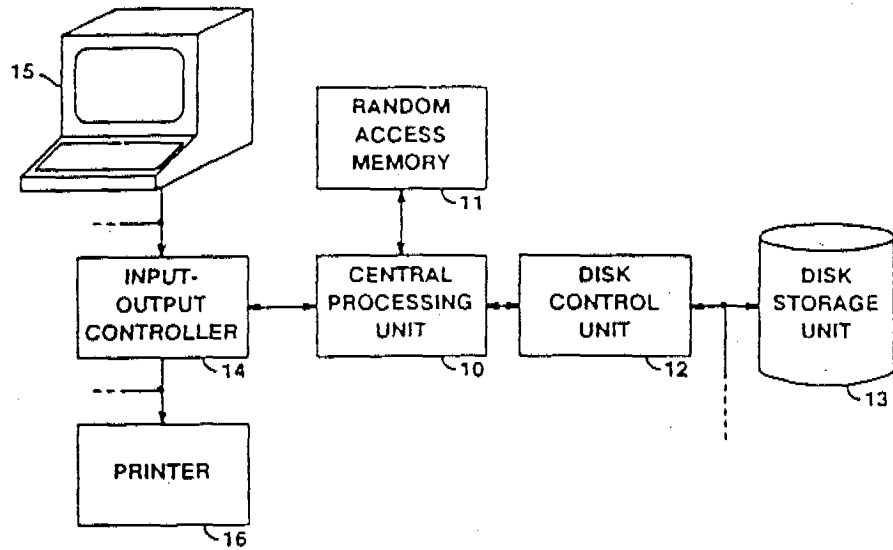


FIG. 2

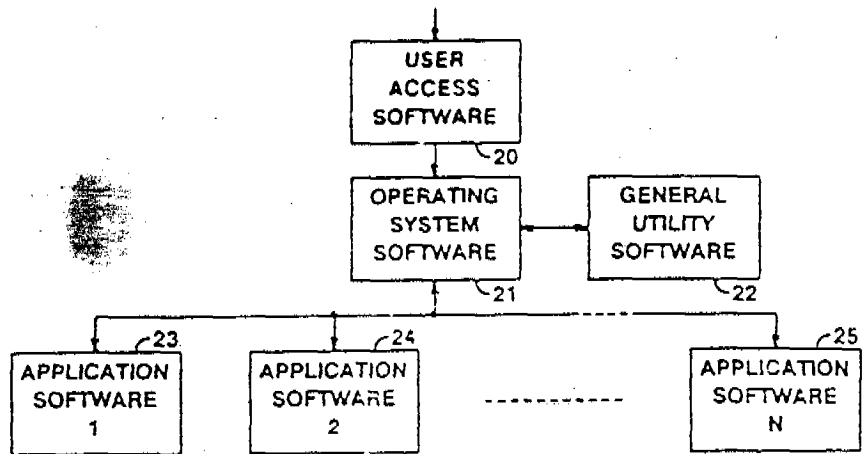


FIG. 3

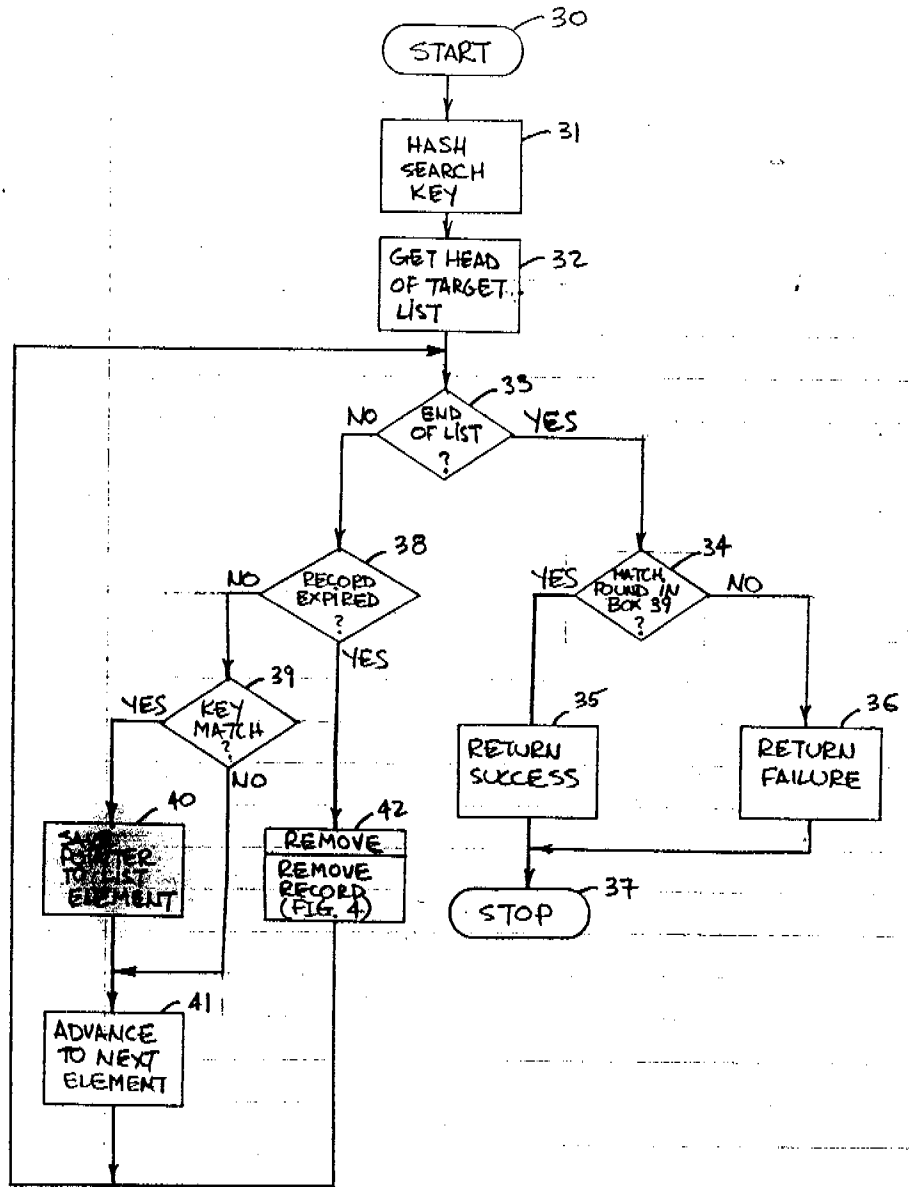
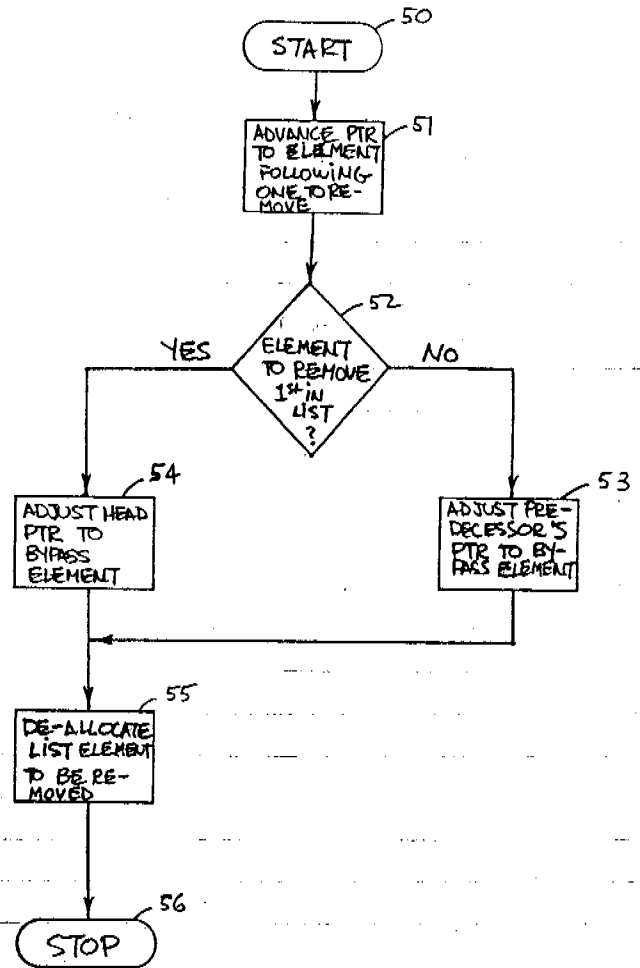


FIG. 4



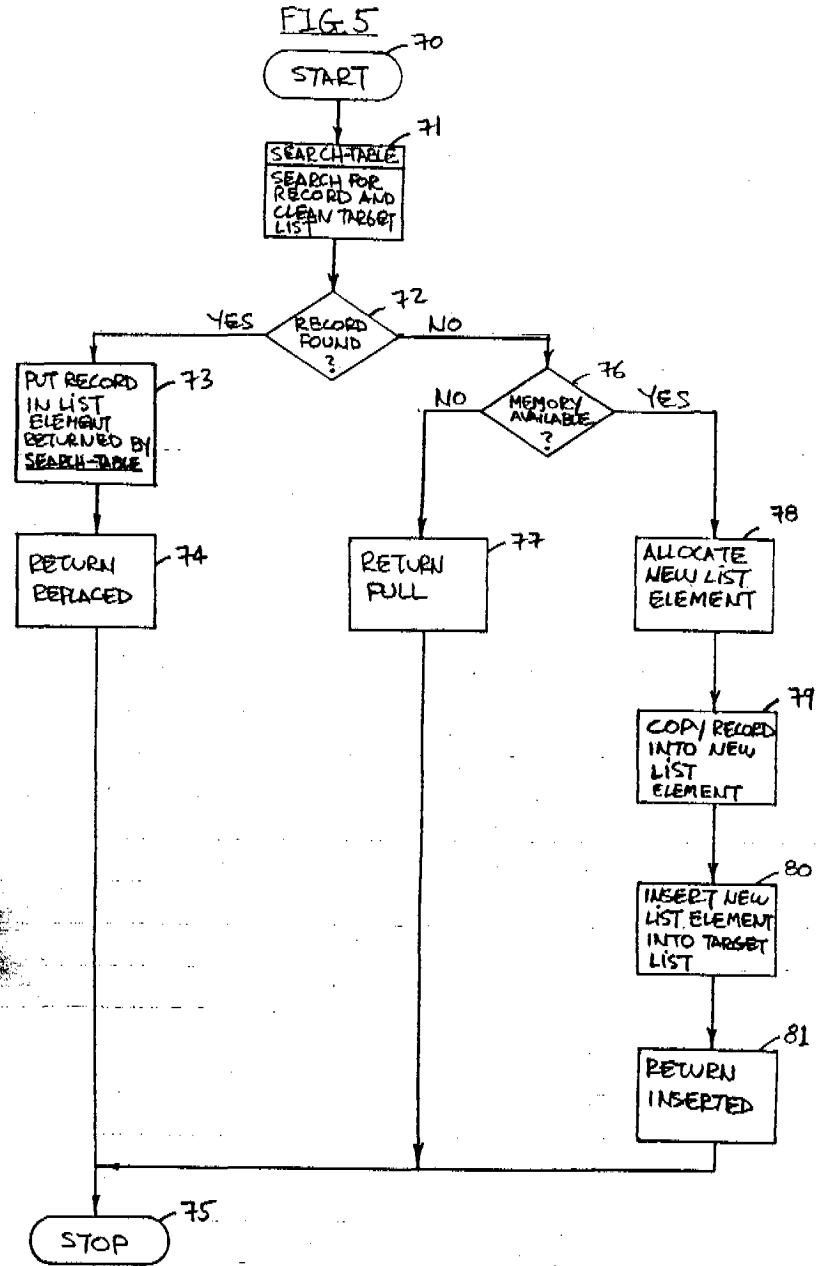


FIG. 6

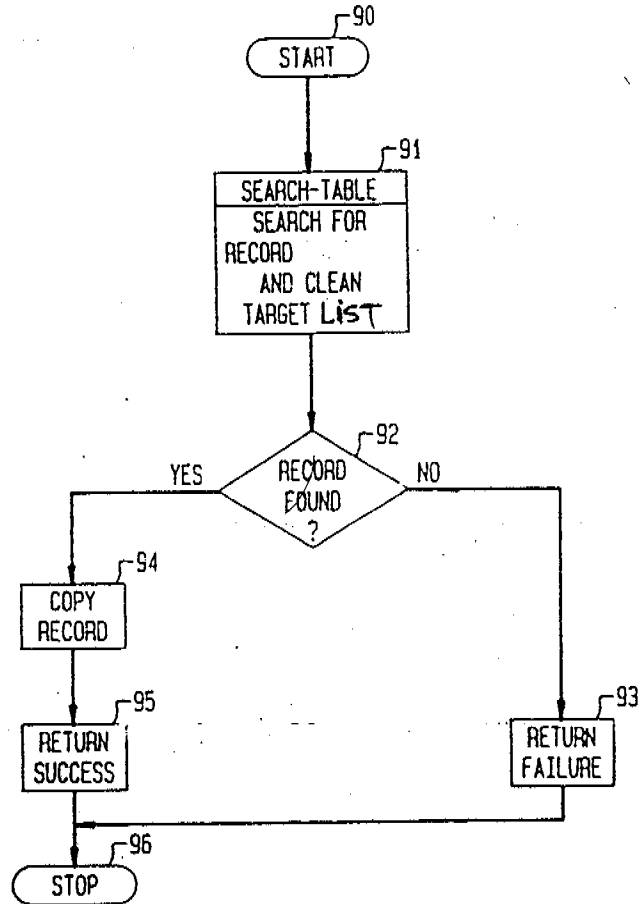
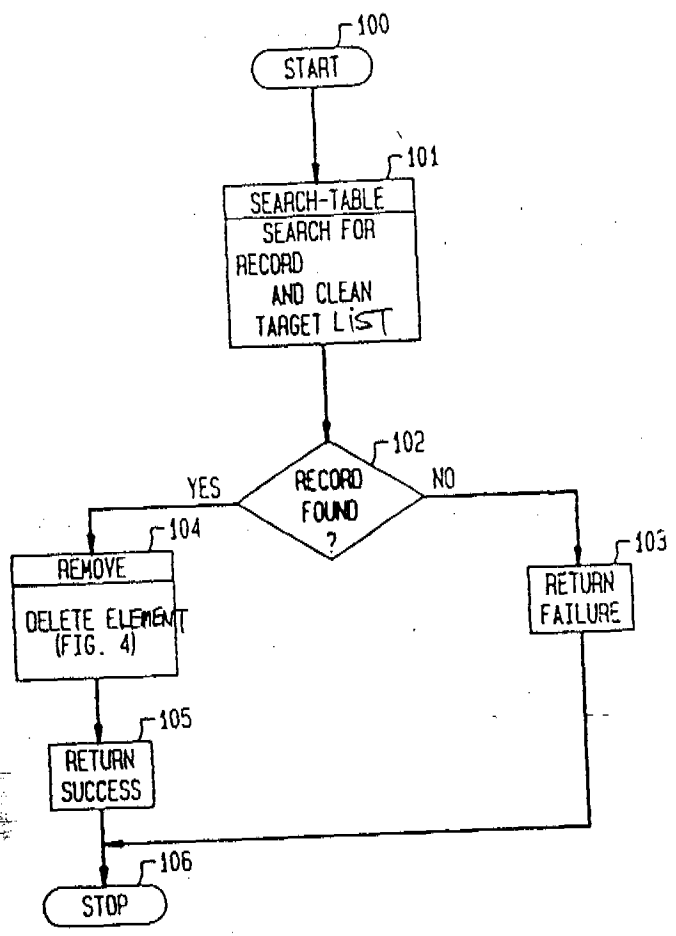


FIG. 7



paper #2

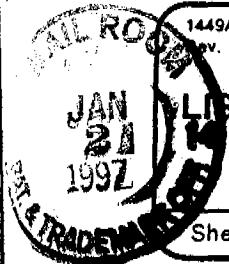
PTO/SB/08A (6-95)

Approved for use through 9/30/98, OMB 0651-0031

Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE

Please type a plus sign (+) inside this box

+



1449A/PTO
Rev. 10/95

U.S. Department of Commerce
Patent and Trademark Office

Complete If Known

LIST OF PRIOR ART CITED BY APPLICANT

(use as many sheets as necessary)

Application Number	
Filing Date	
First Named Inventor	Richard M. Nemes
Group Art Unit	
Examiner Name	
Attorney Docket Number	

Sheet 2 of 2

U.S. PATENT DOCUMENTS						
Examiner Initials ¹	Cite No. ¹	U.S. Patent Document		Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document MM-DD-YYYY	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number	Kind Code ² (if known)			
RJA	4	5,121,495		Richard M. Nemes	06-09-1992	707/3

FOREIGN PATENT DOCUMENTS								
Examiner Initials ¹	Cite No. ¹	Foreign Patent Document			Name of Patentee or Applicant of Cited Document	Date of Publication of Cited Document MM-DD-YYYY	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁴
		Office ³	Number ³	Kind Code ⁵ (if known)				

Examiner Signature	<i>MA Law</i>	Date Considered	4/13/98
--------------------	---------------	-----------------	---------

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Unique citation designation number. ² See attached Kinds of U.S. Patent Documents. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.1⁶ if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take .2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

paper 7d

PTO/SB/08B (6-95)

Approved for use through 07/31/96. OMB 0651-0031

Patent and Trademark Office U.S. DEPARTMENT OF COMMERCE

Please type a plus sign (+) inside this

1449B/PTO Rev. 10/95 U.S. Department of Commerce Patent and Trademark Office

Complete if Known

LIST OF PRIOR ART CITED BY APPLICANT

(use as many sheets as necessary)

Sheet 1 of 2

Application Number	
Filing Date	
First Named Inventor	Richard M. Nemes
Group Art Unit	
Examiner Name	
Attorney Docket Number	



Handwritten signatures and initials, including 'JVA' and 'JVA', and a date '4/13/98'.

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS

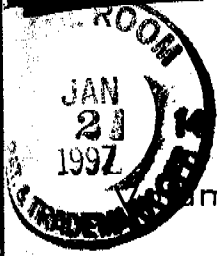
Examiner Initials	Cite No. 1	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, country, where published, source.
JVA	1	D.E. KNUTH, <u>The Art Of Computer Programming, Volume 3, Sorting and Searching</u> , Addison-Wesley, Reading, Massachusetts, 1973, pp. 506-549.
JVA	2	R.L. KRUSE, <u>Data Structures and Program Design, Second Edition</u> , Prentice-Hall, Englewood Cliffs, New Jersey, 1987, Section 6.5, "Hashing," and Section 6.6, "Analysis of Hashing," pp. 198-215.
JVA	3	D.F. STUBBS and N.W. WEBRE, <u>Data Structures with Abstract Data Types and Pascal</u> , Brooks/Cole Publishing Company, Monterey, California, 1985, Section 7.4, "Hashed Implementations," pp. 310-336.

Examiner Signature	JVA	Date Considered	4/13/98
--------------------	-----	-----------------	---------

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Unique citation designation number. 2 Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take .2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.



Volume 3 / **Sorting and Searching**

THE ART OF COMPUTER PROGRAMMING

Reading, Massachusetts
Menlo Park, California · London · Amsterdam · Don Mills, Ontario · Sydney

This book is in the
ADDISON-WESLEY SERIES IN
COMPUTER SCIENCE AND INFORMATION PROCESSING

Consulting Editors

RICHARD S. VARGA and MICHAEL A. HARRISON

Copyright © 1973 by Addison-Wesley Publishing Company, Inc. Philippines copyright 1973
by Addison-Wesley Publishing Company, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system,
or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording,
or otherwise, without the prior written permission of the publisher. Printed in the United
States of America. Published simultaneously in Canada. Library of Congress Catalog Card
No. 67-26020.

ISBN 0-201-03803-X
DEFGHIJ-MA-7987

This book forms a natural
Chapter 2, because it
basic structural ideas.
book is only for those sy
tion of general-purpose
But in fact the area of
discussing a wide variety

How are good algorithms
How can given algorithms
How can the efficiency
How can a person
same application?
In what senses can
How does the theory
How can external
with large data bases

Indeed, I believe that
somewhere in the content

This volume comprises
is concerned with sorting
been divided chiefly into
also are supplementary
tations (Section 5.1)
Chapter 6 deals with
files; this is subdivided
of keys, or by digital
problem of secondary

6.4 HASHING

So far we have considered search methods based on comparing the given argument K to the keys in the table, or using its digits to govern a branching process. A third possibility is to avoid all this rummaging around by doing some arithmetical calculation on K , computing a function $f(K)$ which is the location of K and the associated data in the table.

For example, let's consider again the set of 31 English words which we have subjected to various search strategies in Section 6.2.2 and 6.3. Table 1 shows a short MIX program which transforms each of the 31 keys into a unique number $f(K)$ between -10 and 30. If we compare this method to the MIX programs for the other methods we have considered (e.g., binary search, optimal tree search, trie memory, digital tree search), we find that it is superior from the standpoint of both space and speed, except that binary search uses slightly less space. In fact, the average time for a successful search, using the program of Table 1 with the frequency data of Fig. 12, is only about 17.8 μ , and only 41 table locations are needed to store the 31 keys.

Unfortunately it isn't very easy to discover such functions $f(K)$. There are $41^{31} \approx 10^{50}$ possible functions from a 31-element set into a 41-element set, and only $41 \cdot 40 \cdot \dots \cdot 11 = 41!/10! \approx 10^{43}$ of them will give distinct values for each argument; thus only about one of every 10 million functions will be suitable.

Functions which avoid duplicate values are surprisingly rare, even with a fairly large table. For example, the famous "birthday paradox" asserts that if 23 or more people are present in a room, chances are good that two of them

Table 1
TRANSFORMING A SET OF KEYS INTO UNIQUE ADDRESSES

Instruction	A	AND	ARE	AS	AT	BE	BUT	BY	FOR	FROM	HAD	HAVE	HE	HER
LD1N K(1:1)	1	1	1	1	1	2	2	3	6	6	8	8	8	8
LD2 K(2:2)	1	1	1	1	1	2	2	3	6	6	8	8	8	8
INC1 -8,2	9	6	10	13	14	5	14	18	2	5	-15	-15	-11	-11
J1P *+2	9	6	10	13	14	5	14	18	2	5	-15	-15	-11	-11
INC1 16,2	5	6	10	13	14	16	14	18	2	5	2	2	10	10
LE2 K(3:3)	7	6	10	13	14	16	14	18	2	5	2	2	10	10
J2Z 9F	7	6	10	13	14	16	14	18	2	5	2	2	10	10
INC1 -28,2	1	18	13			9		7	7	22	1			1
J1P 9F	1	18	13			9		7	7	22	1			1
INC1 11,2		3	3					23	20	7	35			
LDA K(4:4)		3	3					23	20	7	35			
JAZ 9F		3	3					23	20	7	35			
DEC1 -5,2									9		15			
J1N 9F									9		15			
INC1 10									19		25			
LDA K	7	3	3	13	14	16	9	18	23	19	-7	25	10	1
CMPA TABLE,1	7	3	3	13	14	16	9	18	23	19	-7	25	10	1
JNE FAILURE	7	3	3	13	14	16	9	18	23	19	-7	25	10	1

will have the same month and day. A random function which maps 23 no two keys map into the same month and day. Skeptics who doubt this result should attend the large parties they attend. [unpublished work of H. Dav. *Essays* (1939), 45. See also J. *Mecmuasi* 4 (1939), 145-163. *Theory* (New York: Wiley, 19...

On the other hand, the approach of J. *Nowinski* and W. *Turski*, *CACM* 1968, 10, 700-701, shows that a suitable function can be found for a set of 31 keys. It is amusing to solve a puzzle like this.

Of course this method has some disadvantages. It must be known in advance, and it is making it necessary to start with a more versatile method if we give keys to yield the same value. The ambiguity after $f(K)$ has been computed is a serious problem.

These considerations lead to a method known as *hashing* or *scatter search*. It chops something up or to make use of some aspects of the key in the search. We compute a *hash* value where the search begins.

The birthday paradox tells us that for $K_i \neq K_j$ which hash to the same

HIS	I	IN	IS	IT	NOT	OF	Conte
-8	-9	-9	-9	-9	-15	-16	
-8	-9	-9	-9	-9	-15	-16	
-7	-17	-2	5	6	-7	-18	
-7	-17	-2	5	6	-7	-18	
18	-1	29				25	4
18	-1	29	5	6	25	4	
18	-1	29	5	6	25	4	
12					20		
12					20		
12	-1	20	5	6	20	4	
12	-1	29	5	6	20	4	
12	-1	29	5	6	20	4	

the given argu-
branching process.
doing some arith-
the location of K

which we have
Table 1 shows
a unique number
programs for
tree search,
the standpoint
less space. In
gram of Table 1
41 table loca-

$f(K)$. There
41-element set,
distinct values
functions will be

even with a
asserts that if
two of them

will have the same month and day of birth! In other words, if we select a random function which maps 23 keys into a table of size 365, the probability that no two keys map into the same location is only 0.4927 (less than one-half). Skeptics who doubt this result should try to find the birthday mates at the next large parties they attend. [The birthday paradox apparently originated in unpublished work of H. Davenport; cf. W. W. R. Ball, *Math. Recreations and Essays* (1939), 45. See also R. von Mises, *İstanbul Üniversitesi Fen Fakültesi Mecmuası* 4 (1939), 145-163, and W. Feller, *An Introduction to Probability Theory* (New York: Wiley, 1950), Section 2.3.]

On the other hand, the approach used in Table 1 is fairly flexible [cf. M. Greniewski and W. Turski, *CACM* 6 (1963), 322-323], and for a medium-sized table a suitable function can be found after about a day's work. In fact it is rather amusing to solve a puzzle like this.

Of course this method has a serious flaw, since the contents of the table must be known in advance; adding one more key will probably ruin everything, making it necessary to start over almost from scratch. We can obtain a much more versatile method if we give up the idea of uniqueness, permitting different keys to yield the same value $f(K)$, and using a special method to resolve any ambiguity after $f(K)$ has been computed.

These considerations lead to a popular class of search methods commonly known as *hashing* or *scatter storage* techniques. The verb "to hash" means to chop something up or to make it mess out of it; the idea in hashing is to chop off some aspects of the key and to use this partial information as the basis for searching. We compute a *hash function* $h(K)$ and use this value as the address where the search begins.

The birthday paradox tells us that there will probably be distinct keys $K_i \neq K_j$ which hash to the same value $h(K_i) = h(K_j)$. Such an occurrence is

ADDRESSES

	HAVE	HE	HER
8	-8	-8	-8
8	-8	-8	-8
15	-11	-11	-11
15	-11	-11	-11
2	10	10	10
2	10	10	10
2	10	10	10
1	.	.	1
1	.	.	1
15	.	.	.
15	.	.	.
15	.	.	.
25	.	.	.
25	10	1	.
25	10	1	.
25	10	1	.

	HIS	I	IN	IS	IT	NOT	OF	ON	OR	THAT	THE	THIS	TO	WAS	WHICH	WITH	YOU
Contents of r11 after executing the instruction, given a particular key K																	
8	-8	-9	-9	-9	-9	-15	-16	-16	-16	-23	-23	-23	-23	-26	-26	-26	-28
8	-8	-9	-9	-9	-9	-15	-16	-16	-16	-23	-23	-23	-23	-26	-26	-26	-28
15	-7	-17	-2	5	6	-7	-18	-9	-5	-23	-23	-23	-15	-33	-26	-25	-20
15	-7	-17	-2	5	6	-7	-18	-9	-5	-23	-23	-23	-15	-33	-26	-25	-20
2	18	-1	29	.	.	25	4	22	30	1	1	1	17	-16	-2	0	12
2	18	-1	29	5	6	25	4	22	30	1	1	1	17	-16	-2	0	12
2	18	-1	29	5	6	25	4	22	30	1	1	1	17	-16	-2	0	12
1	12	20	.	.	.	-26	-23	-18	.	-22	-21	-5	8
1	12	20	.	.	.	-26	-22	-18	.	-22	-21	-5	8
15	-14	-6	2	.	11	-1	29	.
15	-14	-6	2	.	11	-1	29	.
15	-14	-6	2	.	11	-1	29	.
25	-10	.	-2	.	.	-5	11	.
25	-10	.	-2	.	.	-5	11	.
25	-10	.	-2	.	.	-5	11	.
25	12	-1	29	5	6	20	4	22	30	-10	-6	-2	17	11	-5	21	8
25	12	-1	29	5	6	20	4	22	30	-10	-6	-2	17	11	-5	21	8
25	12	-1	29	5	6	20	4	22	30	-10	-6	-2	17	11	-5	21	8