PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 15/00 — A1

(11) International Publication Number: **WO 94/23377**

(43) International Publication Date: 13 October 1994 (13.10.94)

(21) International Application Number: PCT/US94/03362

(22) International Filing Date: 25 March 1994 (25.03.94)

(30) Priority Data:
08/039,702 30 March 1993 (30.03.93) US

(71) Applicant: SQUIBB DATA SYSTEMS, INC. [US/US]; 59 West Pierpont Street, Kingston, NY 12401 (US).

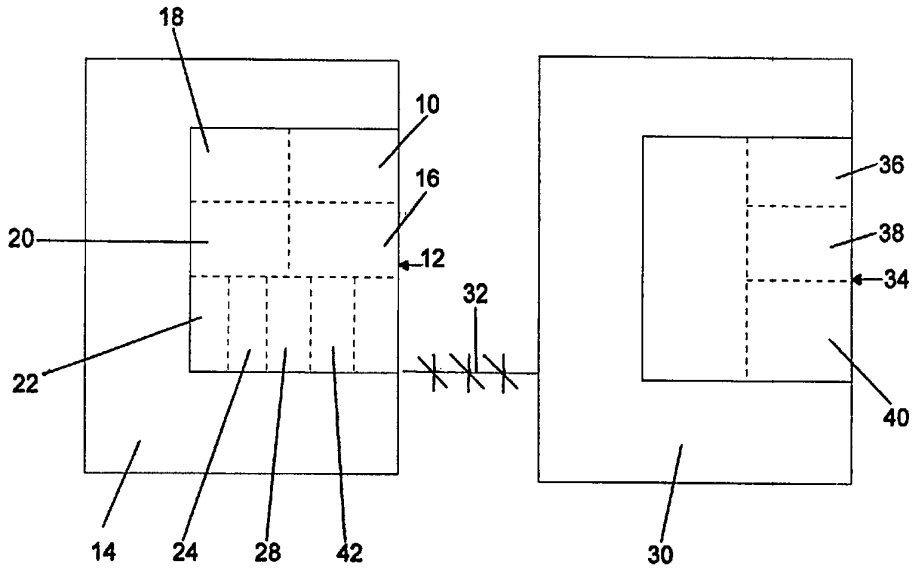
(72) Inventor: SQUIBB, Mark; 109 Ringtop Road, Kingston, NY 12401 (US).

(74) Agents: ROSEN, Daniel et al.; Rosen, Dainow & Jacobs, 489 Fifth Avenue, New York, NY 10017 (US).

(81) Designated States: AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, ES, FI, GB, GE, HU, JP, KP, KR, KZ, LK, LU, LV, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SI, SK, TT, UA, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: FILE DIFFERENCE ENGINE



(57) Abstract

Invention maintains duplicate files in safe places. A TOKEN Table (18) reflects the indices of successive segments of the file and the exclusive-or (XR) and Cyclic redundancy check (CRC) products of the characters in each segment. The XR and CRC products of segments in the updated file (10) are compared to the XR and CRC products in the TOKEN Table (18). On mismatch, the segment (window) for the updated file is bumped one character and new XR and CRC products generated and compared. The indices of the TOKEN Table (18) and the offsets are set forth in a Match Table (24). Next the updated file is scrolled through to form the TRANSITION Table (36) which is the Match Table (24) and the updated file non-matching information. The TRANSITION Table (36) may be sent to another location (30) having a copy of the earlier file thereat. A reconstruction program at the location looks at the TRANSITION Table (36) to determine where to get the characters for the updated file it is creating.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**GB**	United Kingdom	**MR**	Mauritania
AU	Australia	**GE**	Georgia	**MW**	Malawi
BB	Barbados	**GN**	Guinea	**NE**	Niger
BE	Belgium	**GR**	Greece	**NL**	Netherlands
BF	Burkina Faso	**HU**	Hungary	**NO**	Norway
BG	Bulgaria	**IE**	Ireland	**NZ**	New Zealand
BJ	Benin	**IT**	Italy	**PL**	Poland
BR	Brazil	**JP**	Japan	**PT**	Portugal
BY	Belarus	**KE**	Kenya	**RO**	Romania
CA	Canada	**KG**	Kyrgystan	**RU**	Russian Federation
CF	Central African Republic	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CG	Congo			**SE**	Sweden
CH	Switzerland	**KR**	Republic of Korea	**SI**	Slovenia
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SK**	Slovakia
CM	Cameroon	**LI**	Liechtenstein	**SN**	Senegal
CN	China	**LK**	Sri Lanka	**TD**	Chad
CS	Czechoslovakia	**LU**	Luxembourg	**TG**	Togo
CZ	Czech Republic	**LV**	Latvia	**TJ**	Tajikistan
DE	Germany	**MC**	Monaco	**TT**	Trinidad and Tobago
DK	Denmark	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**US**	United States of America
FI	Finland	**ML**	Mali	**UZ**	Uzbekistan
FR	France	**MN**	Mongolia	**VN**	Viet Nam
GA	Gabon				

FILE DIFFERENCE ENGINE

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to techniques for representing file differences useful in computer file protect system and other systems, and more particularly to file transfer techniques useful in an electronic data backup system wherein only changes in a file are periodically sent to the backup system and in other systems.

2. Discussion of Prior Information

It is well known to off-load computers at the end of a work day to secure the data file against computer failure. It is also known to transmit the file to an off-site location for additional file security.

What is not known is the generation of a set of representations of the changes in a file, and the periodic relocation of that set of representations and its use to update the previous version of the file.

SUMMARY OF THE INVENTION

Accordingly it is an object of the invention to generate a set of representations of the changes made in a computer file during a period of time.

Another object of the invention is to generate a set of representatives of the changes made in a computer file which can be used to update an earlier version of the file, or to create a previous version of an updated file.

Still another object of the invention is to generate and to use such a set of representations in a cost and time effective manner.

The objects of the invention are achieved through computer programs designed to run on a micro- and mini- computers. A first or SCAN program is designed to create a TOKEN Table (or file signature) of mathematical representations of segments of the file as it exists at the start of a period (earlier file (EF)). The TOKEN Table reflects the indices (ordinal numbers) for all of the segments in the earlier file, and the exclusive-or (XR) and cyclic redundancy check

(CRC) products of the set of characters for each segment. Actually, two CRC products are generated for each segment; a sixteen bit one and a thirty-two bit one. The three products, XR and two CRC, are generated for speed in comparisons: the XR product is first compared because it is the fastest comparison; then the slower sixteen bit CRC one if necessary; and finally the still slower thirty-two bit CRC if necessary.

A second program is used at the end of the period to create a MATCH Table setting forth the location of segments in the current file that are identical to those in the earlier file. The MATCH Table lists the indices of all of the segments in the earlier file and the file offsets of the first character of the corresponding identical segment in the updated file. The second program calculates the mathematical representations of the first segment (window) in the updated, revised or current file, first calculating only the XR product and comparing it to the XR product for the first earlier-file segment in the TOKEN Table and noting whether a match exists. If so, it then calculates the sixteen bit CRC product and compares it to the sixteen bit early file CRC product and notes whether a match exists; if so, it finally calculates the more time consuming but more reliable thirty-two bit CRC product and compares it to the thirty-two early file CRC product and notes whether a match exists; and if so, makes an index and offset entry in the MATCH Table for the identical segments; the offset entry being the ordinal number of the first character in the current file segment string of characters. (The earlier file segments are numbered (indexed) sequentially.). If a segment match is obtained, the second program calculates one or more mathematical representations for the next segment in the current file, and compares them to the products associated with the next index in the TOKEN Table and representing the second segment of the earlier file. However, if a mismatch obtained, the window (which retains segment size) is bumped along one character, new product(s) calculated for the window characters and comparison(s) again made with the same representations of the earlier file segments in the TOKEN Table. This continues until a match obtains at which time the index for the earlier file segment and the offset of the first character in the matching current file window (segment) are

recorded in the MATCH Table. The process then continues as above to the end of the current file. Only the XR product is calculated in the event of an XR product mismatch; the sixteen bit and the thirty-two bit CRC products being generated respectively only in the event of earlier matches of the XR and sixteen bit CRC products.

A third program creates a TRANSITION Table that reflects what's in the current file that's not in the earlier table, and where. It scrolls through the list of indices and offsets in the MATCH Table, to see if each offset number differs from the previous one by the segment size. When such an offset differs from the previous one by more than the segment size, it adds the segment size to the first offset to determine the file ordinal number of the first character in the matching information, subtracts one from the second offset to determine the last character, goes to the current file and lifts therefrom that set of characters beginning with that ordinal number and stopping with the character preceding the extra-spaced offset, and adds them to the MATCH Table to create with the index a TRANSITION Table.

Thus creation of the Transition Table involves assuring that every character in current file is accounted for in the TRANSITION Table. The MATCH Table provides all of the information necessary for this accounting. Each entry in the beginning column represents a match in the early file of segment characters to the current file characters at location beginning. The matching segment in the early file is located at that offset, which is equal to the index times the segment size in early file.

Essentially the same process is followed for a deletion. The second program, if no match obtained for an earlier file segment by the end of the updated file (or over a predetermined number of segments as conditioned by the character of the file), would have proceeded to endeavor to match the next index mathematical representations in the TOKEN Table with a current file segment, with no offset entry having been made in the MATCH Table for the index of the segment that was unmatched. On proceeding with the index and representations of the next earlier-file segment, the window of the current file would be bumped along, and the index and

offset number entered in the MATCH Table when the match of the mathematical representations occurred. The third program on scrolling through the MATCH Table offsets, notes the missing offset, notes the preceding offset, adds the segment size to the previous offset and copies from that number forward the reduced characters if any in the current file before the next offset, into the TRANSITION Table and in association with the index number of the unmatched segment.

The TRANSITION Table is used to update a copy of the earlier file. Typically, a fourth program and the earlier version of the file are on an off-site location and the TRANSITION Table representations are electronically transmitted thereto. The fourth program will examine the indexes and offsets of the TRANSITION Table, copying segments from the earlier file where the succeeding offset just differs by the segment size, into what is to be a duplicate version of the updated file, making additions where the offset numbers differ from the preceding ones by more than the segment size with the information provided in the TRANSITION Table, and substitutions from the TRANSITION Table where the offset numbers are missing.

As observed earlier,the TOKEN Table mathematical representations of file segments may be the products of exclusive-oring of the characters in successive earlier file segments and of generating two cyclic redundancy check (CRC) products for each earlier file segments. Corresponding XR products are most quickly generated, but do not detect character order differentiating; a sixteen bit CRC will catch most of these transpositions; a relatively slowly generated thirty-two bit CRC product will detect essentially all of them.

As observed earlier the MATCH Table is generated by the second program generating mathematical representations of the segment sized windows of the current file, and comparing the representations of a window with an index's associated mathematical representations in the TOKEN Table. As long as matches obtain, successive window sized segments of the current file are addressed and a MATCH Tale listing reflecting the early file segment index and the current segment first character offset is generated. Normally three mathematical representations of

each segment obtain--an exclusive-or (XR) one and sixteen bit and thirty-two bit cyclic redundancy check (CRC) ones. In the interests of speed, the XR products are compared first, and if a mismatch occurs in them, it is clear that the segments are unmatched. However, even if the XR products match, the segments may not match because the XR operation is not sensitive to the transposition of characters. Accordingly, it is also necessary on XR match, to compare the sixteen bit CRC product. On sixteen bit CRC match, it is desirable to do a thirty-two bit CRC match for most applications to achieve practically one hundred percent certainty. The generation of the CRC product is a relatively slow process and is avoided where possible as on XR mismatch. However, the great benefit of avoiding CRC calculations occurs in operations subsequent to segment mismatch.

As observed earlier, upon detection of a mismatch, a segment sized window representing only a one character displacement of the window in the current file is operated upon to determine its mathematical representations and compare them with the representations of the just compared TOKEN Table representations, then on mismatch upon successor windows until a match obtains or the end of file is reached. By generating first the quickly generated exclusive- or (XR) products, and only on match generating the more slowly generated CRC products, a significant amount of time can be saved.

Applicant has further discovered that even the exclusive-oring process can be expedited on a one-character shift of the window under consideration. Thus the new XR product need not involve the exclusive-oring of each of the characters of the new window: rather only the exiting character and the entering character need be exclusive-ored with the existing XR product of the just tested segment. The second exclusive-oring of the exiting character amounts to a subtraction of it from the segment product.

Another feature of the invention is that the amount of updating material that must be transmitted to the off-site is minimal; normally being less than five percent (5%) of the current file.

An advantage of the invention is that it provides an easy way to secure a user's data from fire, theft and tampering.

Another advantage is that is provides an inexpensive disaster recovery insurance.

A further advantage is that it eliminates the tedious chore of computer backup, and allows the user's office time to be dedicated more fully to the productivity and profitability of his or her business.

Yet another advantage of the invention is that programs embodying the invention can be incorporated in larger programs for handling large model files which are immune to character insertions and deletions and grow in size to accommodate new records. Thus under certain circumstances, it is possible to skip creation of MATCH and TRANSITION Tables by windowing techniques.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects, features, and advantages of the invention will become apparent from a reading of the following specification when considered with the appended drawings wherein:

Figure 1 is a diagram of a system according to the invention;

Figure 2 is a representation of the contents of a user's earlier file;

Figure 3 is a representation of the contents of the user's updated file;

Figure 4 sets forth a TOKEN Table which consists of the indices and the exclusive-or (XR) and cyclic redundancy check (CRC) products of successive segments of the earlier program;

Figure 5 sets forth a MATCH reflecting a comparison of the TOKEN Table contents with the identical segments of the current program;

Figure 6 sets forth a TRANSITION Table reflecting the differences in the two files of Figs. 2 and 3;

Figure 7 is a flow chart setting forth the method of the first or TOKEN Table generating program;

Figure 8 is a flow chart setting forth the method of the second or MATCH Table generating program;

Figure 9 is a flow chart setting forth the method of the third or TRANSITION Table generating program;

Figure 10 is a flow chart setting forth the method of the fourth or reconstruction program;

Figure 11 sets forth a MATCH Table having an alternate format to that of Figure 5;

Figure 12 sets forth a TRANSITION Table having an alternate (IBE) format; and

Figure 13 sets forth a TRANSITION Table having another (IBC) format.

DETAILED DESCRIPTION OF EMBODIMENT

The system concept of the invention is shown in Fig. 1. A user maintaining a data file 10 (Figs. 1 and 2) such as "This is a test file." in a memory generally indicated by the number 12 of a computer 14, would at the start of the workday, activate a first program 16 (Fig. 7) also in the computer memory to partition the earlier file into five characters segments, generate XR and CRC products for each segment, and list each segment by its index (ordinal number) and its products in a TOKEN Table 18 (Fig. 1 and 4) in the memory 12 and that he might care to store for the workday on a disk drive (not shown) to maximize available memory space. During the day, the user would update, as by inserting the word "radical", the file 10 so that it reads "This is a radical test file." (Fig. 3), using a conventional data base program 20 also in the memory. At the end of the workday, the user would activate a second program 22 (Fig. 8), then located in the memory 12, to create in the memory 12 a MATCH Table 24 (Figs. 1 and 5) consisting of indices from the TOKEN Table and the offsets of the first characters of segments (windows) of the updated file that result from the matching of the exclusive-or (XR) and cyclic redundancy check (CRC) products of updated file segments with the products associated with the indices.

The third program (Fig. 9) works in conjunction with the MATCH Table to develop the TRANSITION Table which succinctly defines what is or is not in the current file that was not or was in the earlier file. It does this by scrolling through the offsets in the MATCH Table. It looks at the offsets for successive indices, checking to see if it differs from the previous offset by the segment size. When it doesn't, it copies the current file material between the end of the last segment and the start of the segment with the greater than segment size offset, into the TRANSITION Table, there to be associated with the index of the greater than segment size offset.

It results that the TRANSITION Table reflects the changes obtaining in the current program over the earlier file.

The TRANSITION Table is then electronically sent, using conventional modems and communication programs, to the off-site computer 30 over telephone wire 32. Computer 30 has a memory generally indicated by the number 34 which receives the TRANSITION Table in section 36. The earlier file would normally already be resident in section 38 of the memory 34, representing the file as it was updated at the end of the previous day. The fourth program (Fig. 10) creates a duplicate of the current file by inserting or deleting information according to the dictates of the TRANSITION Table in memory section 36 and the contents of the earlier file in memory section 38. As long as the offsets for successive indices differ by the segment size, the program copies the segments for the indices from the earlier file into the memory section 40. When an addition is indicated as at index 2 because the offset figure (18) is larger than the normal segment size (5) over the previous offset figure (5), the fourth program looks for the additional information (here "radical") in the related area of the TRANSITION Table and inserts it in the duplicate file, after the tenth character (number 9). The fourth program then continues reviewing the TRANSITION Table and copying from the earlier file until another non-segment-size-distant offset (here none) is detected.

The TOKEN Table 18 (Figs. 1 and 4) is created by the first program 16 (Fig. 7) first partitioning the earlier file into fixed sized segments (here five characters) and then generating a mathematical representation for each segment by first creating

an exclusive-or (XR) hash product of the characters of each earlier file segment, and then creating a cyclic redundancy check (CRC) product of the characters of each of the segments. Characters of a segment are normally represented by bytes of eight binary bits each, which bytes are exclusive-ored in turn, the first two being exclusived-ored and then that product with the byte of the next character, and so on until the last character (here the fifth) has been exclusive-ored, and the product (exclusive-or hash) stored in the TOKEN Table 18 with the associated index.

Mathematical operations other than exclusive-oring, such as checksum may be employed, but exclusive-oring is the fastest.

Since the exclusive-oring operation is not character order definitive, a second mathematical operation (here cyclic redundancy check(CRC)) is performed by the first program on each segment and recorded in the TOKEN Table with the associated index. There are many polynomials for generating CRCs: applicant incorporates in the first program the thirty-two bit ANSI X. 3.66 CRC Checksum files that appear in FIPS PUB 71 and in FED-STD-1003, to generate the CRC entry for each index in the TOKEN Table (Fig. 4).

As CRC calculations for CRC products are very slow compared to those for XR products, it may be desirable to increase the reliability of the XR product(s). Reliability may be increased by generating intermediate XR products, such as the XR product of half of the characters in a segment. Thus given a series of arbitrarily assigned binary terms for the various characters as indicated below, with segment size equal eight, quarter and half products may be generated respectively and are shown in the right hand columns, the underlining in the more leftward columns indicating where the products are taken:

character	binary	XR (Quarters)	XR (half)
T	01100110 (C0)	(Q0)	
H	00110110 (C1)	01010000	
I	01010101 (C2)	(Q1)	(H0)
S	10110111 (C3)	11100010	10110010
-	00010000 (C4)	(Q2)	

i	01010101 (C5)	01000101	
s	10110111 (C6)	(Q3)	(H1)
-	00010000 (C7)	010100111	11100010
XR (seg.)	01010000	01010000	01010000

The segment "This_is_" may be divided into one or more equal sized parts (excepting perhaps the last to accommodate an odd segment size). In this example four separate terms are used. (Each subterm, during nonmatching window operations, may be adjusted by x-oring out the exiting character and x-oring in the incoming character.)

The quarter terms may be combined in any order or fashion. The following expressions are equivalent:

XR (segment) = C0 ^ Cl^ C2 ^ C3 ^ C4 ^ C5 ^ C6 ^ C7

= Q0 ^ Ql ^ Q2 ^ Q3

= H0 ^ H1 , where"^" = XR operation

Therefore the XR product Term may be expanded to include more information than the XR segment carries without sacrificing the runtime speed advantage of the XR window technique.

The following expressions represent combinational variations of the above rules which, in the context of this invention may be used to provide additional efficiency benefits;

Whereas the Regular XR (segment) product term is the XR sum of C[0-7] = Q[0-3] = H[0-1], subterms (Q and H...) may be introduced to add precision (quality) to the XR test.

The quality of the XR test is important. The runtime efficiency of the engine depends upon 1) the quality of the XR test; a more precise XR test solution will cause less frequent fallback to the more expensive CRC tests; and 2) the efficiency

with which the XR tests can be implemented as related to standard processor architecture.

Currently supported processor environments favor a two term XR test:

(8 bit) (8 bit)

Specifically: XR (segment) & H (0) --> 16 bits.

Using just a standard XR test (8 bits), a false compare with random data permits a CRC hit (1:[2 to 8th = 256]) times. By including a single subterm (H0) this ratio is increased to (1:[65636 = 2 to 16th]), thereby saving considerable CPU resources. Given more robust computer processing units, more subterms may be included to further enhance performance and reliability.

If sufficient terms are included, the reliability of the XR product or test may be enhanced sufficiently such that the stringent requirements for the CRC tests may be relaxed and the minimum reliability requirements achieved in a more time effective manner.

The flow chart of Fig. 7 details the operation of the first program. Characters are read from the earlier file EF and when the fifth character is encountered, an index counter is incremented, the XR and CRC products calculated, the index counter contents and products stored in the TOKEN Table, and the character counter reset.

Once the TOKEN Table has been created, the file 10 is opened for updating. In the example of this application, the earlier file "This is a test file" (Fig. 2) is updated, using conventional data base program in memory section 42, with the word "radical" followed by a space to read "This is a radical test file." (Fig. 3), and this is assumed to be its status at the end of the workday.

The second program 22 (Fig. 2) is actuated to begin the creation of the MATCH Table (Fig. 5) at day's end. It generates mathematical representations of similarly sized segments (windows) of the updated program and compares them to those of the earlier program in the TOKEN Table. Thus, the first segment of the five characters in the updated file would have its exclusive-or and CRC products compared to those in the TOKEN Table of the first segment of the earlier file, and if

found equal as they should be in the examples since the identical characters and order obtain in each, the index "0" indicating the first earlier file segment and the offset "0" indicating that the first character in the identical segment in the updated file is the first character in the updated file, are recorded in the MATCH Table. In the examples, the second segments should be found equal, too, and the index "1" and offset "5" recorded. On the third segment, a mismatch of the exclusive-or, and if not, the CRC products should obtain as the segments being compared have the characters "test" and "radic", respectively. Accordingly, the second program increments as the next updated-file segment to be mathematically treated (that is, advances or bumps the updated-file window of consideration, here five characters), one character forward and here involving the updated-file characters "adica". When compared to the mathematical representations of the number 2 segment of the earlier file, a mismatch should again be detected. The program then repeats the operation on succeeding segment windows spaced one character until a match obtains, here involving the third file segment containing the characters "test". The index "2" and the offset "18" are recorded in the MATCH Table. The representations of the next segments, of the updated file would match those in the TOKEN Table for the index "3" and hence the index "3" and offset "23" would be entered in the MATCH Table and a suitable end of the file signal noted to end further segment search.

The flow chart of Fig. 8 details the operation of the second program. The program looks at the first segment (window) of the updated file, calculates the XR product, compares it to the XR product for the first index in the TOKEN Table, and if no match obtains, bumps the window to repeat the process. If an XR match obtains, it calculates the CRC product and compares it to the CRC product for the index in the TOKEN Table. If a match obtains, the index and the file offset of the first character in the updated file are recorded in the MATCH Table and the window of consideration for the second file shifted one full segment.

As an example of a file deletion, consider that the only change to the earlier file involved deletion file involved deletion of the word "a". The updated file would now read: "This is test file.". When the second program is actuated to generate the

MATCH Table, it would find that the mathematical products for the first segments match and proceed as above. However, it would find that those of the second segments ("is a " and "is te") do not match. Nor would it find a match on bumping the segment window through the rest the updated file, thus no offset of the updated file ordinal number of the first character of an identical segment would be recorded in the MATCH Table, only the index "1" would be recorded. Then the program would return to look for matches of the third earlier file segment in the updated file. Of course here matches would obtain on the segments "test", and the index "2" and the offset "8" duly recorded in the MATCH Table. On scrolling through the MATCH Table, the third program would note that the offset "8" for the updated program segment matching the third segment in the earlier program was less than a full segment size (5) from the previous one, and thereupon copy the current-file information after the last matching segment and before the matching segment, into the TRANSITION Table. The fourth program, in creating an updated version of the original program, too would notice the nearness of the offsets and use that as its clue to substitute the TRANSITION Table information for that in the earlier file segment.

A flow chart detailing the operation of the third program is set forth in Fig. 9. The MATCH Table is read for index and offset information, the offset number compared with zero or that of the previous index, and if the difference is the segment size, advances to read the next index in the MATCH Table. If the difference was other than the segment size, it scrolls through the updated file to copy information since the preceding segment into the TRANSITION Table.

A flow chart detailing the operation of the fourth program is set forth in Fig. 10. TRANSITION Table indices and their offset number are read and offset numbers checked for zero or for segment-size differences from the previous index. If it is the case, the index segment is copied from the earlier file into what is to be the second copy of the updated file. If it is not the case, information is copied from the TRANSITION file into what is to be the second copy.

An alternate format for the MATCH Table is set forth in Fig. 11. The format visualizes not listing individually successive identical segments, but rather merely

indicating their number with the index for the first of such segments. Thus, with respect to the files of Figs. 2 and 3, the second and fourth segments of the current file would not be listed; rather the indexes for the first and third segments would have associated with them under the column "Extent" the numeral "2" to indicate that two successive segments were identical in each case. This format has the advantage of being even more concise. It in effect maintains the segment count information as the index. It is more compact for large files, where an addition may contain nine or more bytes. Hence it requires less computer memory to support it.

Alternative embodiments of the TRANSITION Table may be employed. Thus, the TRANSITION Table may include other information, such as the segment size and supervisory error detection codes, and program version information.

Thus the TRANSITION Table may be written in any one of several base formats:

index beginning (offset)	IB (Fig. 6)
index beginning (offset) count	IBC (Fig. 13)
index beginning (offset) end	IBE (Fig. 12).

The IB format sets forth the index (ordinal number) of the successive identical segments and the offsets of their beginning characters. The IBC format sets forth the index of the first offset of a set of identical segments and the number (count) of such identical segments in the set. The IBE format sets forth the end character of each set of identical segments, instead of the count. IBE format facilitates carrying end-of-file (EOF) information with it.

The IB format has been used exclusively for discussion in this document to avoid confusion.

The other formats, IBC and IBE are advantageous because they require less space than the IB format.

It will be evident that applicant has provided a method, means, and articles (signatures) for readily changing a secure file to reflect changes made in a copy of it elsewhere during an operating period such as a day. Applicant first creates a signature of the file in the TOKEN Table; the signature is the index and a

mathematical representation of each of the successive equal-sized segments of the file. Then the file is updated to reflect the transaction occurring during the course of the day. At the end of the day, the mathematical representations of successive window segments of the updated file are compared with those of the earlier file in the TOKEN Table, with the window segment for the updated file being bumped one character until either a match or end of file signal is detected where upon the next earlier file segment index is compared for. A Match Table is created indicating the indexes (ordinal numbers) of the earlier file segments and the offsets (file character ordinal numbers) of the first characters of the identical updated-file segments. Next the updated file is scrolled through, and the non-matching information is copied from it, using the index numbers and offset numbers of the MATCH Table as a guide, into the TRANSITION Table. The TRANSITION Table may now be shipped to where a copy of the earlier file is, to update it according to the TRANSITION Table, by the fourth program.

As observed earlier, the programs of the invention can be incorporated in larger programs for handling large model files.

Certain files are immune to character insertions and deletions, these are large model files (LMF's). Nearly all commercial data bases create large model files. Large model files follow the following rules:

1) The file must be immune to character skewing. Data base applications, which use records and fields obey this convention. Any change or update to a field or record will have no effect upon the location (offset) of any other record in the file.

2) If the file needs to grow in size to accommodate new records, additional records are appended to the end of the file. This format permits recycling of early file space as long as the character skewing rule is not violated.

A small model file (SMF) is any file which does not obey large model file (LMF) rules.

Thus under certain circumstances, it is possible to skip the windowing portion of this algorithm, thereby saving time and computer resources.

As a large model file example, consider an:

EF "This is a test file." , and a

CF "This is a test file. mom"

Large model file rule 1 is met: The file modification (change i to would not cause the bytes associated with any other file character to be effected. Each matching character is at the same byte offset in early file and current file. No character skewing occurred.

Large model file rule 2 also is met: The additional information (mom) appeared at the end of the file.

For the purpose of this example, if more than 50% of a file matches under large model file rules, the small model file logic is to be omitted.

Considering further the above large model file,

This_ is_a_ test_ file.

This_ os_a_ test_ file. _mom

match nonmatch match match nonmatch

It can be seen that the first, third and fourth five-character segments match, and the second and fifth don't. Thus the statistics are that three segments match, therefore more than 50% current file was matched by early file.

The MATCH Table would be:

O O

2 10

3 15

Therefore the TRANSITION Table should read:

O O

2 10

3 15 \0

os_a_ _ mom

This method of pretest, yields substantial runtime reductions on large model files. The windowing technique cannot yield superior results on files which follow large model file rules, it is therefor unnecessary to attempt to resolve such a large model file using the small model file windowing technique.

The larger program employs a two pass technique to efficiently process both large model files and small model files, it is unnecessary to employ the small model files windowing technique.

The first pass assumes that the file is a large model file. Each corresponding file segment from the TOKEN Table is compared to the corresponding current file segment. Comparison statistics are maintained. If more than a user specified percentage of the file matches using large model file rules, (or large model files rules were given at runtime) the small model file resolution logic is skipped, thereby saving considerable computer resources and time.

If the first pass comparison yields sufficiently poor results, or small model file rules were invoked at runtime, the small model file logic is invoked and the file is processed normally.

In addition to the above another computer back-up application, the invention lends itself to other applications. One such application is a media distribution application. A data vendor may maintain a large data base, say of sales tax rates for various cities and states, which is periodically updated. A mail order business may be a user of such a data base in order to charge the correct sales tax on each of its various order fulfillments. The data base maintainer would initially send the mail order business the complete data base. He would also generate a TOKEN Table of the data base as it then existed, using the first program. Upon updating his data base, he would develop a MATCH Table and a TRANSITION Table, using the second and third programs, and he would transmit the TRANSITION Table to the user. The user would update his data base, using the fourth program. The process would be repeated for subsequent updates.

In another application, the invention is used to harmonize files being independently updated at two different locations. Each location would generate a TRANSITION Table to reflect its changes, using the second and third programs, and send the Transition Table to the other location. The receiving location would then use the fourth program with the other's TRANSITION Table to update its file with the other locations changes.

In an archive maintenance application, a data base is continuously modified, say daily, and it is desired to maintain a complete daily archive. Initially, the data base as it exists at the start, would be copied onto the storage media and a TOKEN Table developed using the first program. At the end of the first day, a TRANSITION Table would be developed, using the second and third programs and copied onto the storage media having the original data base; also a new TOKEN Table would be created. At the end of the second day, a new TRANSITION Table would be developed, and copied onto the storage media; also a new TOKEN Table. By continuing this daily process a complete archive can be maintained on significantly less media than would normally be required, and the data base can be restored, using the fourth program, as it existed on any given day.

Another application involves a WORM Work File Maintenance System. (A WORM is a Write Only Read Many storage device.) Such a system may include a server having a computer and a normal read/write/erase disk drive (DASD) and a WORM. Several other computers are connected to the computer server through a high speed network. At the end of the first day, all data from the other computers is copied onto the DASD. At the end of the second day, the data on the other computers is compared to the data on the DASD to generate the TRANSITION Table which is copied on the WORM. The data from the other computers is copied onto the DASD, either directly or by updating, to replace that of the previous day. On the third day, the process of the second day is repeated, except for the new TRANSITION Table being added to the TRANSITION Table already on the WORM. The fourth and further days essentially involve repeats of the third day. The data as it existed on a given day can be reconstructed by working backwards from the current data on the DASD and the TRANSITION Tables on the WORM. Thus - all history is maintained, and no additional supporting backup is required for the discrete and physically separate computers.

The invention also is useful in word processing applications. Instead of full copies of the updated versions, only TRANSITION Tables need be maintained. The initial version of a document is stored as is on a backup device. A TRANSITION

Table is developed for the first updated version and stored on the backup device too. A TRANSITION Table is developed for the next updated version, reflecting the differences between it and the previous updated version, and it too is added to the back-up- device. The backup device thus holds the original version of the document and TRANSITION Tables reflecting subsequent documents in a minimum of storage space, and any version of the document can be recreated.

From a consideration of the above, it is evident that it is not a requirement of the invention that the early file be retained. This invention was adopted to facilitate the description of the invention. Kindly consider the below application which reverses the role of early file and current file, allowing the early file to be created from the current file and the TRANSITION Table.

In a word processor application, it is the intent of the user to maintain a logical path back to every revision of a work, while retaining only the current file and the TRANSITION Table.

Assume that the author is working on the fifth generation of the work and has the fourth saved locally. The author generates a TOKEN Table for 5(5.h) and compares it to 4. The TRANSITION Table created is in a "reverse" sense. Restated, the fifth generation plus the TRANSITION Table equals the fourth generation. The author can recover the fourth generation of the document by applying the T54 file to the current file (CF) to recover early file (EF). The earlier fourth 4 can now be erased.

In this example, the invention does not require the present availability of the early file.

A printout listing of the computer program embodying the invention follows. It consist of the Terac.exe modules in object code in hexadecimal representation and the Terac Supervisory lynx.ksh module in UNIX Script representation, and the caret.exe modules in object code in hexadecimal representation.

While there has been disclosed preferred embodiments of the invention, it will be appreciated that other embodiments embodying principles of the invention can be created by those skilled in the art without departing from the spirit or wording of the appended claims.

WHAT IS CLAIMED IS:

1. A combination comprising a storage device, a file stored in said storage device, and a token table stored in said storage device, said token table comprising first and second different hashing mathematical representations of fixed equal length character segments of said file.

2. A combination according to claim 1, wherein said first hashing representation is an exclusive-or representation of said segments of equal character lengths.

3. A combination according to claim 1, wherein the segments cover successive sets of characters and are indexed for identification.

4. A combination as claimed in claim 1, wherein the first mathematical representation comprises an exclusive-or signature term of each segment of said file.

5. A combination according to claim 1, wherein the second mathematical representation comprises a cyclic redundancy product of the characters of the respective segments of the file.

6. A combination according to claim 1, wherein the second mathematical representation comprises order sensitive hash representation of the characters of the respective segments of said file.

7. A computer apparatus comprising a storage device, a file stored in said storage device, means for generating a token table signature of said file and storing it in said memory, said means for generating and storing a signature comprising means for generating first and second different hashing mathematical representations of fixed equal length character segments of said file.

8. An apparatus according to claim 7, wherein the means for generating said first hashing mathematical representation comprises means for generating an exclusive-or representation of said segments of equal character length.

9. An apparatus according to claim 7, wherein the segments cover successive sets of characters and are indexed for identification.

10. An apparatus according to claim 7, wherein the means for generating the first hashing mathematical representation comprises means for generating an exclusive-or signature of the characters of the respective segments of the file.

11. An apparatus according to claim 7, wherein the means for generating the second hashing mathematical representation comprises means for generating order sensitive cyclic redundancy check of the characters of the respective segments of said file.

12. An apparatus according to claim 7 wherein the means for generating the first and second hashing mathematical representations comprises means for generating a first hashing mathematical representation that is an exclusive-or representation and a second hashing mathematical representation that is an order sensitive hash term of the characters of the respective segments.

13. A method for generating a file signature, in a computer having a memory with a file therein and data processing means for producing a signature of said file, said method comprising generating first and second different hashing mathematical representations of fixed equal length character segments of said file.

14. A method according to claim 13, wherein the step of generating said first representation comprises generating an exclusive-or representation of said segments of said file.

15. A method according to claim 13, wherein the segments cover successive sets of characters, comprising indexing said sets for identification.

16. A method according to claim 13, wherein the step of generating the first mathematical representation comprises generating an exclusive-or term of each segment of said file.

17. A method according to claim 13, wherein the step of generating the second mathematical representation comprises generating a cyclic redundancy product of the characters of the respective segments.

18. A method according to, wherein the step of generating said second mathematical representation comprises generating an order sensitive hash representation of the characters of the segments.

19. A combination comprising a memory, a signature and a first data file stored in said memory, said signature comprising a difference between said first data file and a second data file with respect to one another, said first and second data files each having successive segments of characters, said signature further comprising indexes of successive segments in said first and second data files and offsets indicating a displacement from a reference point of identical character segments in the first and second files.

20. A combination according to claim 19, wherein the segments are of equal character size and are indexed.

21. A combination according to claim 19, wherein each segment has a first character and each offset indicates the ordinal number in the file of the first character in an identical character segment.

22. A combination according to claim 19, wherein the differences of one file of said first and second data files with respect to the other reflect added information and the segments are of a fixed size, representations of added information associated with corresponding offsets differing by more than the segment size from the previous offset.

23. A combination according to claim 19 wherein the differences of one file with respect to another file reflect information which has been reduced to a lesser amount, at least one of which has no offset, representations of reduced information associated with indexes not having an offset.

24. A computer apparatus having a memory with first and second files stored therein, and means for generating a signature of differences in one of said files with respect to another of said files, said files each having successive segments of characters, comprising data processing means for generating first and second different hashing mathematical representations of fixed equal length character segments of said files,

means for comparing said hashing mathematical representation of said file to identify identical character sequences of said files, means for generating a difference file of portions of said files that are different, and means for generating offsets indicating a displacement from a reference point of identical character segments in said first and second files.

25. An apparatus according to claim 24, comprising means for generating successive equal sized segments in said files and means for indexing said files.

26. An apparatus according to claim 24, wherein each segment has a first character and comprising means for generating offsets which each indicate an offset in one file of the first character in respective identical character segments.

27. An apparatus according to claim 24, wherein differences in one file with respect to another reflect added information and the matching segments are of a fixed size, and further comprising means for associating representations of added information with corresponding offsets differing by more than the segment size from a previous offset.

28. An apparatus according to claim 24 wherein differences of one file with respect to another reflect information which has been reduced to a lesser amount, whereby at least one corresponding index does not have an offset, and further comprising means for associating representations of reduced information with corresponding indexes not having offsets.

29. The method of claim 13 wherein said memory has first and second files stored therein further comprising generating a signature for storage in the memory of differences in said first file with respect to said second file, each of said first and second files having successive segments of characters, comprising a first step of generating, using the data processing means, indexes of successive segments in the second file, and generating offsets indicating a displacement from a reference point of identical character segments in the first file.

30. A method according to claim 29, further comprising generating succesive equal size segments and indexing them by ordinal numbers.

31. A method according to claim 29, wherein each segment has a first character and the second step comprises generating offsets which each identify the ordinal number in the first file of the first character in an identical character segments.

32. A method according to claim 29, wherein the differences in said first file with respect to said second file reflect added information and the segments are of a fixed size and further comprising the step of associating representations of added information with corresponding offsets differing by more than the segment size from the previous one.

33. A method according to claim 29 wherein differences in one file with respect to another reflect information which has been reduced to a lesser amount, whereby at least one index does not have an offset, and further comprising the step of associating representations of reduced information with corresponding indexes not having offsets.

34. In a method of creating, in a computer having a memory with first file and a difference file stored therein, and data processing means, a copy of said second file having fixed length segments of identical characters at different locations from a copy of a first file in memory, comprising generating mathematical representations constituting an index of segments of the first file, and generating mathematical representations constituting an index of segments of the second file, and creating in memory a record of representations that are identical to representations of segments in the first file and of the location or offsets of identical segments in the second file and of information which is different in the second file and its location or offset therein, said information in said record which is different in said second file comprising a difference file.

35. A method according to claim 34, wherein a second computer has a memory and a data processing means, said method comprising has writing in the second

computer memory a copy of the second file, using said difference file and a copy of the first file.

36. A method according to claim 35 for recomposing said second file, comprising writing a copy of the second file in memory by using said first file as a source for coexistent segments and the difference file as a source for segments that are not common to said first and second files.

37. A method according to claim 34, wherein each segment has a first character and the record identifies identical segments by their first file indices and by the file offsets of the first characters of identical segments in the second file.

38. A method according to claim 37, wherein the segments are of a fixed size and are sequentially arranged and the second file is comprised of information that has been added to the first file and information which has been reduced to be of a length less that the length of a segment, said different information constituting added information associated with corresponding offsets differing by more than segment size from the previous one, the reduced information being associated with indices having no offsets.

39. A method for creating a base window segment token comprising creating a file in a computer memory by reading a segment of a file in a computer memory, calculating an exclusive-or signature that is an exclusive-or representation of the characters of respective segments of the file, to create a base window segment token, and adjusting the token to reflect a window segment token of an overlapping segment by exclusive-oring each character entering and leaving set domains for the first mentioned window from said file in said computer memory the added or deleted characters with the product to obtain the exclusive-or product of the set of characters with added or deleted characters.

40. A file comparison method comprising searching a first token table comprised of first and second different hashing mathematical representations of fixed first equal length character segments of a first file for a match with a second token table comprised of first and second different hashing mathematical representations of fixed

equal length character segments of a second file, and adjusting said second token table to correspond to a character segment displaced from said first segment in response to the absence of a comparison with said second token table until the second file is exhausted or a match of the first and second tokens is found.

41. A storage device having first and second files and a signature stored therein, said signature representing differences of said first file with respect to the second file, said signature comprising indexes of fixed length segments in the second file, and offsets indicating a displacement of identical segments in the first file.

42. A storage device according to claim 41, wherein the files have extents of successive identical segments, and including representations of the extent of successive identical segments.

43. A method according to claim 34, wherein the first file is a maintained data base created from the record maintained in memory protected from ordinary user intervention and the second file is a user file, comprising updating the second file.

44. A method according to claim 35, comprising storing copies of the first file at separate locations and a copy at one location is, continually and independently of the other, modified thereat to create a second file, and a copy of the other location's file is created using its record at the second location.

45. A method of periodically maintaining, in a computer having a read/write storage device and data processing means, an archival record of a data base that is being continually updated in the read/write storage device, comprising periodically generating, using the data processing means, exclusive-or mathematical representations and order sensitive hash products of fixed length segments of the data base, and creating a record in the read/write storage device of those representations and has products that are identical to representations and hash products of segments in the data base at the end of the previous periods of said periodical updating.

46. A method according to claim 45, comprising storing each periodically created record in the read/write storage device, and permanently storing the data base as originally created and each of the records in the read/write storage device.

47. A method according to claim 46, comprising modifying said archival record with a word processor program, and the data base is a word processing document.

48. A method according to claim 46 in which the computer also has a write once, read many times device, wherein the data base resides on a read/write storage device, comprising storing a copy of the data base and the records in said write once, read many times device for a permanent archival record.

49. A method according to claim 46, wherein a copy of the original version of the data base is in the read/write storage device, and comprising creating a copy on the read/write storage device of the data base as it existed at any given period of time by using the records up to that time and the original version of the data base and the data processing means.

50. A method for producing a copy of a second file that is an updated version of an original file, comprising producing a token set from each of the original file and an updated file, comparing the token sets of the original file and updated files while offsetting said token sets to identify the locations of identical sequences in the original file and the updated version of said original files.

51. The method of claim 50 wherein said steps of producing a token set comprises producing first and second different mathematical representation from the same segments of said original file and from the matching segments of said updated version of said original file, and comparing the first and second mathematical representations of each of said original file and updated version of said original file.

52. The method of claim 50 further comprising wherein a residue remains of unmatched information following said step of comparing, and further comprising combining said residue with information concerning portions of said original and updated version of said original files, to produce a difference signature file.

53. The method of claim 50 further comprising, generating a list of difference signature between the original and updated files from a residue of said comparison, said residue corresponding to segments of the files that do not compare, and producing said copy of said second file from said list and a copy of only one of said original and second files.

54. The method of claim 50 further comprising transmitting said list to a remote location for said step of producing a copy of said second file, whereby the provision of only one of said second and original files at said remote location is necessary.

55. A method for producing a first file representative of differences between second and third files, comprising generating first and second hash tables from of successive equal length segments of each of said second and third files, respectively, wherein a segment is a set of successive characters, comparing said first and second hash tables and successively offsetting the positions of the hash tables, with respect to said second and third files, to identify segments of said second and third files that match one another, and producing said first file by listing segments of said second and third files that match one another.

56. The method of claim 55 wherein a residue remains after said step of comparing of parts of said second and third files that do not match, comprising adding said residue to said first file.

57. The method of claim 55 wherein said step of forming said first and second hash tables comprises generating successive first and second different mathematical representations of equal length segments of said second and third files, and said step of comparing comprises successively comparing said first and second mathematical representations of said segments of said second and third files.

58. The method of claim 57 wherein said step of forming said first representation comprises forming an exclusive-or representation of successive equal length segments of said second and third files.

59. The method of claim 57 wherein said step of forming said second representation comprises forming an order sensitive hash representation of successive equal length segments of said second and third files.

60. A combination according to claim 2 wherein said exclusive-or representation is a representation that is a combination of the products of exclusive-or representations of a plurality of separate sets, said sets being divisions of a segment.

61. A method for producing a duplicate copy of an updated file from an original file or a duplicate of an original file and difference signature, comprising creating a difference signature, said step of creating a difference signature comprising:

(1) creating a token table by:

(a) producing a token table from an original file by producing a token set for each equal sized contiguous segment of said original file, each token set comprising a primary exclusive-or based token and at least one order sensitive secondary token or cyclic redundance term;

(b) generating the primary exclusive-or token by dividing each segment into sets, generating the exclusive-or product of each set, and concatenating the set exclusive-or products to produce the primary token;

(2) generating a difference signature, using the token table and an updated file, by:

(a) defining a window of consideration for the updated file of a size equivalent to the segment size used to create the token set for the original file;

(b) calculating a primary token for the window of consideration;

(c) searching the token table for a matching primary token;

(d) generating a token in response to finding the primary token from the window in the token table secondary window segment, and comparing it to the secondary token in the corresponding token set;

(e) logging the offset of the current window segment and the offset of the corresponding token table entry to the difference signature to correlate the relative

locations of the matching segment in the original and updated files, in response to finding a match between terms of the secondary token from the window and corresponding terms from the token set;

(f) advancing the window of consideration to the segment beyond the matched text and resuming the method at step (2b) above;

(g) advancing the window of consideration to the next overlapping window in response to the failure of the primary token search or the failure of the secondary token to match the corresponding token set entries;

(h) including the character leaving the window domain in the difference signature as a part of a non-matching segment;

(i) generating a primary token for a new window of consideration from the previous by adjusting the primary token from the previous window by:

(j) dividing the primary token into components corresponding to the sets defined in step (1c), adjusting each component by exclusive-oring characters transitional characters for each set domain and subsequently recomposing the primary token for the new window of consideration;

(k) repeating the cycle of steps (2a) through (2k) until the updated file is exhausted;

(3) then using the difference signature and the original file or duplicate thereof to assemble a duplicate of the updated file by:

(a) using the original file as the source for matching segments;

(b) using the difference signature as the source for non-matching segments; and

(l) assembling the matching and non-matching segments.

62. The method of claims 61 comprising responding to a failed token search by causing the contents of the entire window segment to be included in the difference signature when the updated file is known to be free of character skewing.

63. The method of claim 62 comprising advancing the window of consideration to the next overlapping window.

64. The method of claim 61 comprising constraining the search for the primary window token to the token set representing a corresponding offset from the updated file in the token table when the second file is not subject to character skewing.

65. The method of claim 61 comprising limiting the search for the primary token in the token table to a corresponding bounded region of the token table.

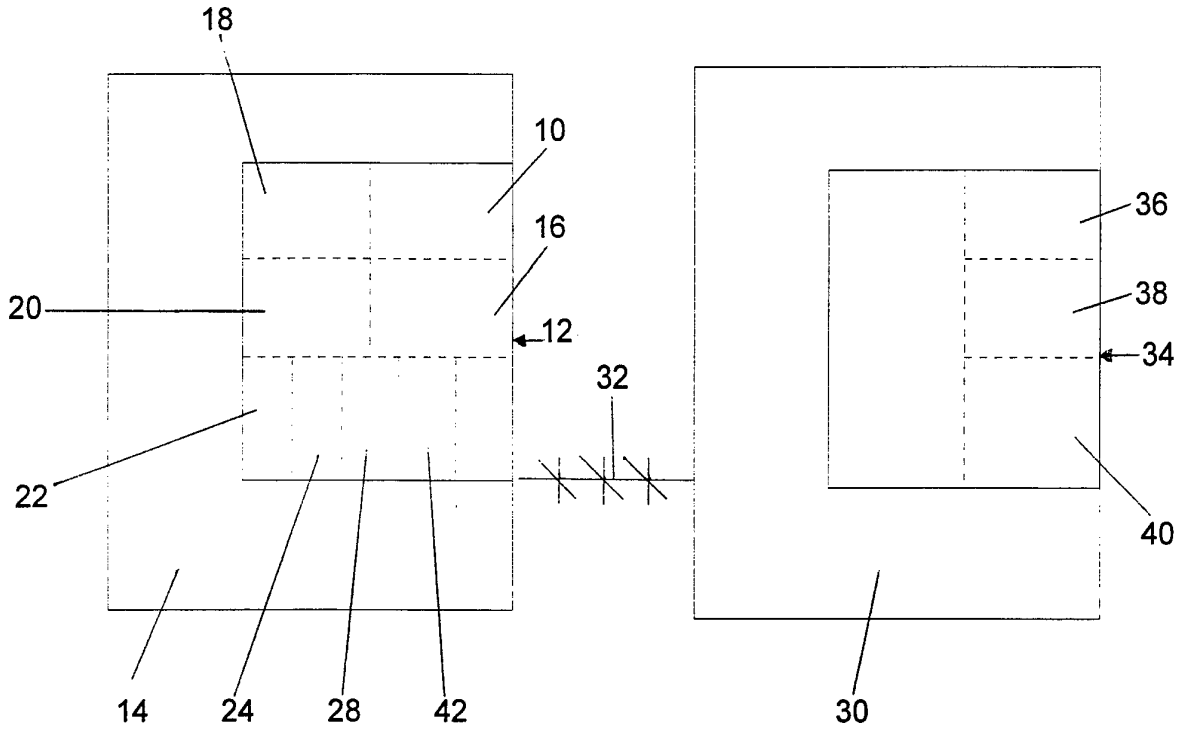


FIG. 1

```
0       1       2       3
1       1       1       1       1
THIS    IS   A  TEST    FILE.
```

FIG. 2

```
1       1       1       1       1       1
THIS    IS   A  RADICAL TEST    FILE.
```

FIG. 3

TOKEN TABLE

INDEX	XR PROD	CRC 16-BIT PROD	CRC 32-BIT PROD
0	XR0	CRC0	CRC0
1	XR1	CRC1	CRC1
2	XR2	CRC2	CRC2
3	XR3	CRC3	CRC3

FIG. 4

MATCH TABLE

E.F. INDEX	U.F. OFFSET
0	0
1	5
2	18
3	23

FIG. 5

TRANSITION TABLE (IB FORMAT)

E.F. INDEX	U.F. OFFSET	ADJUSTMENT
0	0	
1	5	
2	18	"RADICAL"
3	23	

FIG. 6

MATCH TABLE

E.F. INDEX	U.F. OFFSET	EXTENT
0	0	2
2	18	2

FIG. 11

TRANSITION TABLE (IBE FORMAT)

E.F. INDEX	U.F. OFFSET	END	ADJUSTMENT
0	0	9	
2	18	23	"RADICAL"

FIG. 12

TRANSITION TABLE (IBC FORMAT)

E.F. INDEX	U.F. OFFSET	COUNT	ADJUSTMENT
0	0	2	
3	18	2	"RADICAL"

FIG. 13

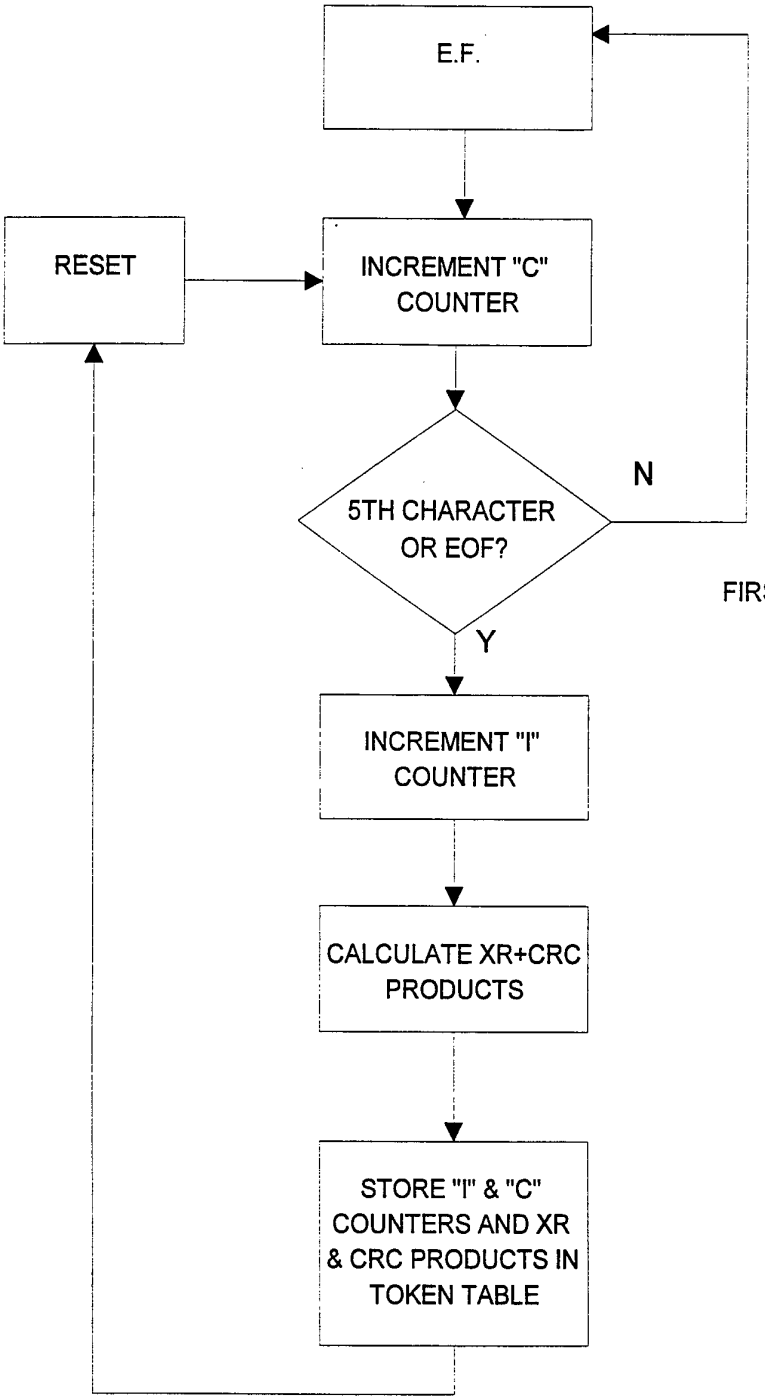
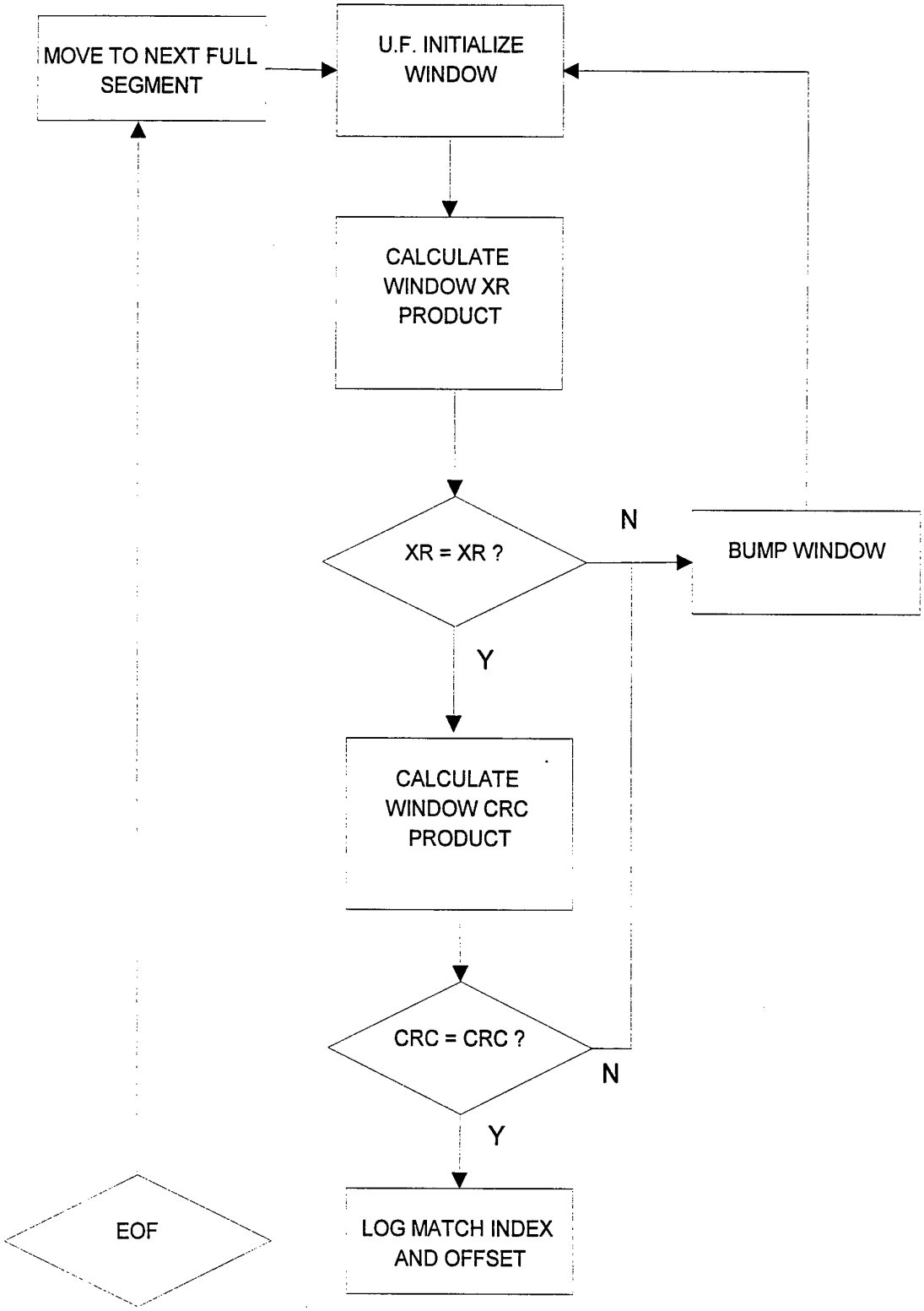

E.F.
RESET
INCREMENT "C" COUNTER
5TH CHARACTER OR EOF?
N
Y
FIRST PROGRAM
INCREMENT "I" COUNTER
CALCULATE XR+CRC PRODUCTS
STORE "I" & "C" COUNTERS AND XR & CRC PRODUCTS IN TOKEN TABLE

FIG. 7



MATCH TABLE PROGRAM FLOW CHART

FIG. 8

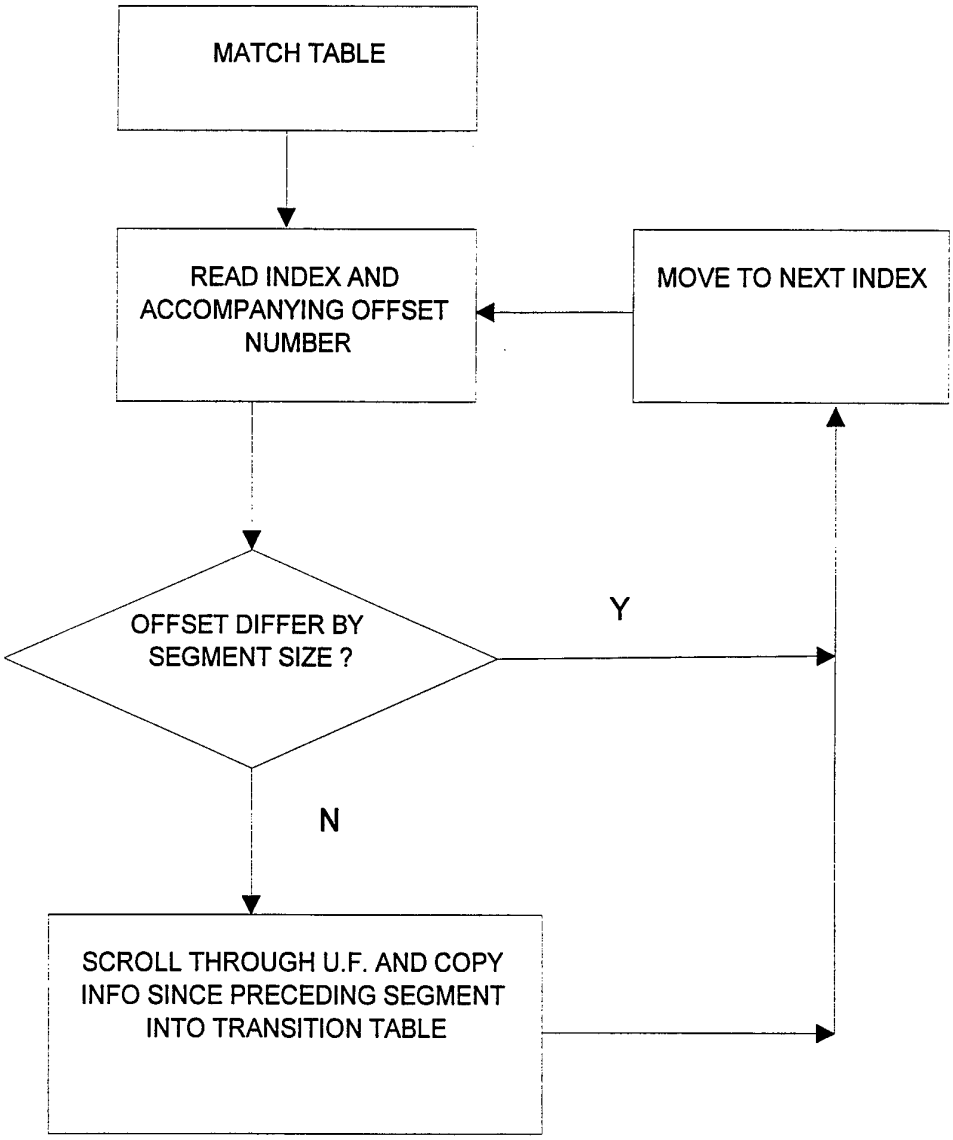

MATCH TABLE
READ INDEX AND ACCOMPANYING OFFSET NUMBER
MOVE TO NEXT INDEX
OFFSET DIFFER BY SEGMENT SIZE ?
Y
N
SCROLL THROUGH U.F. AND COPY INFO SINCE PRECEDING SEGMENT INTO TRANSITION TABLE

FIG. 9

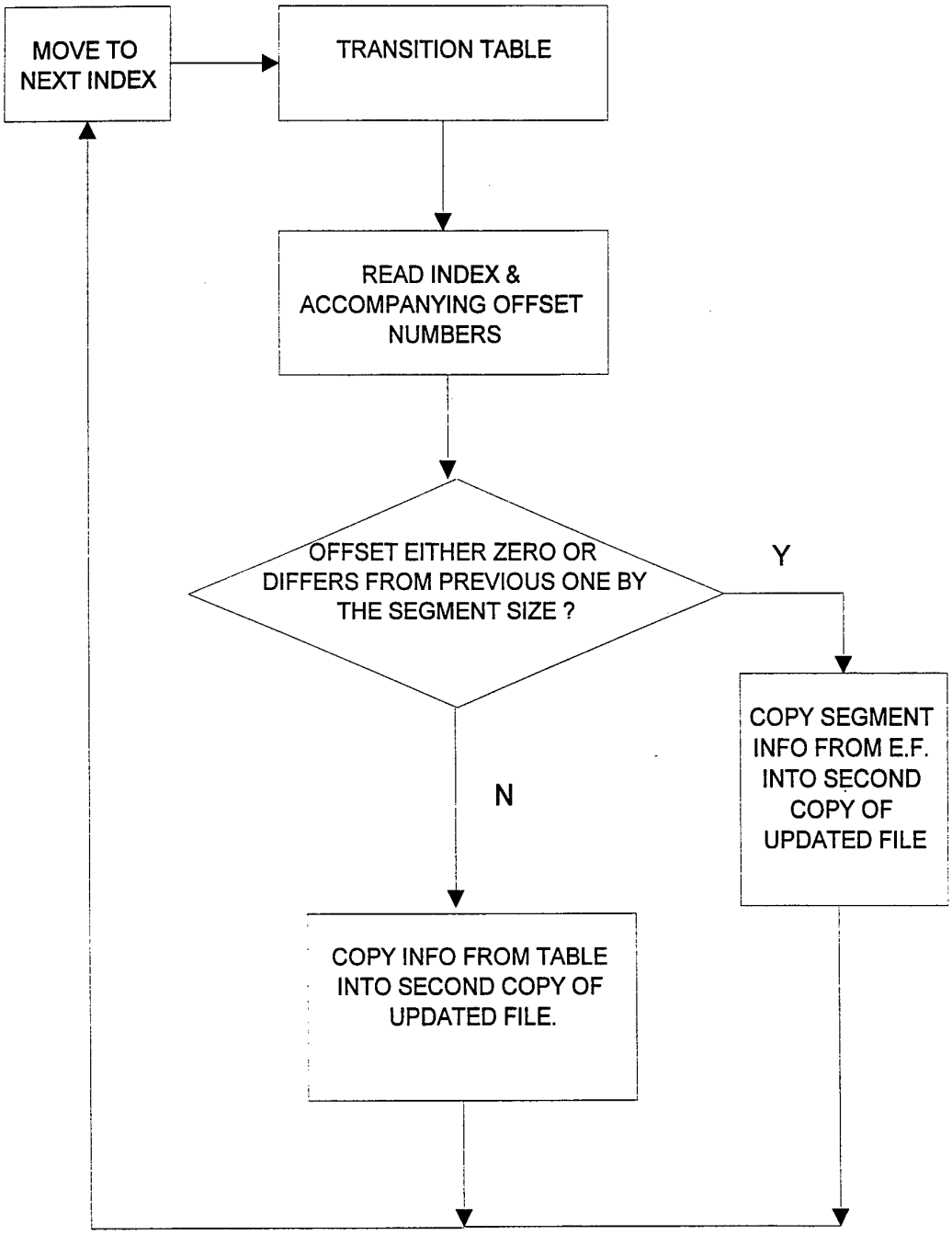


FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US94/03362

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :G06F 15/00

US CL : 395/600, 400; 364/255.7, 252.3, 962.1, 958,

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/600; 364/DIG1; 364/DIG2; 371

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, IEEE Publications, Computerselect

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	USA, 3711863, (Bloom), 16 January 1973, col. 2 line 59- col. line 9, col. 7 lines 50-52, col. 9 line 44-col.10 line 24.	1-65
X - Y	USA, 4807182, (Queen), 21 February 1989, col 1 lines 48-50, col. 2 lines 23-63, col. 5 lines 26-36, col. 7 line 35- col. 8 line 21.	1-4, 6-10, 12-16, 18-60 ---------- 5, 11, 17, 61-65
Y	USA,4641274, (Swank), 02 February 1987, abstract, col. 5 lines 16-55, col. 13 line 31- col.14 line 45.	1-65

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be part of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
30 MAY 1994	01 AUG 1994
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-9564	Authorized officer Lucien Toplu B. Hanaleo for Telephone No. (703) 305-9600

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US94/03362

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	An efficient file structure for document retrieval in the automated office environment, Du et al, IEEE transactions on knowledge and data engineering, vol,. 1, no 2, june 1989, pages 258-273.	1-65
A	The tea-leaf reader algorithm, Griffiths et al, Communications of the ACM, july 1987, v 30 n 7 p 617(4)	1-65