A HIERARCHICAL LARGE DICTIONARY STRUCTURED FOR TWO DIMENSIONAL HASH ADDRESSING

R. AMINPOUR

A THESIS IN

THE DEPARTMENT OF

COMPUTER SCIENCE

Presented in Partial Fulfillment of the Requirements

for the degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

March, 1979

ABSTRACT

A HIERARCHICAL LARGE DICTIONARY STRUCTURED FOR TWO DIMENSIONAL HASH ADDRESSING

RASHID AMINPOUR

The results reported in this thesis indicate that a two level storage hierarchy with 2-dimensional hash addressing proves convenient for structuring of a large term dictionary in an inverted file organization. The two dimensional hash table allows fast response to question terms specified in truncated form, and a method of random access file organization greatly reduces the number of disk accesses. The data are loaded in decreasing order of frequency so that the most frequently referenced items are stored in core and the rest of the words are located on a number of fixed size random access files.

The major parameters affecting performance of two dimensional hashing techniques are considered. Several schemes are designed, implemented, and studied with regard to effect of parameters. The applicablity and performance of each method is discussed. Particular attention is focused on the resolution of collisions in the hash table and dictionary. A number of algorithms are designed and implemented, and retrieval algorithms are also given.

The effects of differently dimensioned hash tables of the same size are examined and it is shown that choice of a small size of row in the hash table allows fast search for terms expressed in right truncated form.

A new method for entering and retrieving information in a hash table is described. The method is intended to be more efficient for static key sets.

For two dimensional hash addressing some guidelines for selection of the method of key-to-digits conversion, the identifier transformation, choice of a divisor, collision resolution, differently dimensioned hash tables of the same size, bucket sizes, loading order, and hierarchy of storage are considered. For examination of the effect of these factors more than 80 experiments have been performed with respect to two data sets. The results are shown in appendix I. Experiments cover a wide range of parameters, and are believed to provide a practical guideline to the design and structure of a large term dictionary in an inverted file organization.

ACKNOWLEDGEMENTS

I wish to express my deep appreciation to Prof. H.S. Heaps for his able guidance, invaluable suggestions and careful reading of drafts throughout the course of this investigation.

I am also thankful to Ginette Lavigne for her assistance in collecting some of the references and drawing diagrams used in this thesis.

A special thanks goes to my mother F. Tamjidi for her moral support, encouragement and understanding during my studies.

TABLE OF CONTENTS

List of figures ix

List of tables ix

CHAPTER 1

INFORMATION RETRIEVAL FROM A TEXTUAL DATA BASE 1

1.1 Textual Data Base 1

1.2 File Structure 2

1.3 Inverted File Organization 4

1.4 File Compression 8

1.5 Objectives of Present Investigation 18

CHAPTER 2

KEY-TO-ADDRESS TRANSFORMATION TECHNIQUES 23

2.1 Scatter Storage 23

2.2 Key to Digit Conversions 24

2.3 Hashing Functions 26

2.3.1 Division Method 27

2.3.2 Midsquare Method 28

2.4 Collision Resolution Techniques 30

2.4.1 Linear Search Method and its Extensions 32

2.4.2 Random Search 35

2.4.3 The Quadratic Search and its Extensions 36

2.4.4 Other Collision Resolution Techniques 40

2.5 Choice of a Divisor 41

2.6 Table Size 43

2.6.1 Table Size M = Power of Two 44

2.6.2 Table Size M = Prime Number 44

2.6.3 Table Size M = a Power of 10 45

2.7 Scatter Table 46

CHAPTER 3

DICTIONARY STRUCTURE FOR TWO DIMENSIONAL HASH ADDRESSING 48

3.1 Specification 48

3.2 Experimental Environment 50

3.3 Collisions in Hash Table 52

3.4 Collisions in the Dictionary Dict1 61

3.5 Differently Dimensioned Hash Tables of the Same Size 67

3.6 New Method for Eliminating Collisions 69

3.7 Result of Experiment 77

CHAPTER 4

DICTIONARY SEARCH FOR TERMS IN LEVEL 1 82

4.1 Retrieval of a Term in Level 1 82

4.1.1 Retrieval Algorithm for Various Collision Schemes in Hash1 ... 84

4.1.2 Retrieval Algorithm for Various Collision Schemes in Dict1 ... 87

4.2 Retrieval of Terms Specified in Truncated Form 89

4.2.1 Retrieval of Truncated Words for Various Collision Schemes in Hash1 91

4.2.2 Retrieval of Truncated Words for Various Collision Methods in Dict1 93

4.3 Retrieval Algorithm for New Collision Scheme 94

4.3.1 Retrieval Algorithm to Search for a Term in Dict1 ... 95

4.3.2 Retrieval Algorithm to Search for Truncated Terms in Dict1 97

4.4 Estimated Times for Computation and Disk Accesses ... 97

CHAPTER 5

ORGANIZATION OF LEVEL 2 100

5.1 Specification 100

5.2 Hashing Scheme 102

5.3 Collisions Resolution Scheme 103

5.4 Eliminating the Hash Tables in the Second Level Store 105

5.5 Dictionary Search for Terms in Dict2 107

CHPATER 6

CONCLUSIONS 109

6.1 Conclusions and Final Remarks 109

REFERENCES 121

APPENDIX I 126

APPENDIX II 211

LIST OF FIGURES

1.1 Inverted File Organization 6

1.2 Inverted File Organization with Coded Data Base 9

3.1 Dictionary Structure and Access Scheme for Dict1 49

3.2 Addressing Via a Scatter Table 72

3.3 Addressing the Data File Via Two Dimensional Index Table 74

LIST OF TABLES

6.1 The Proportion of Data Base Terms Contained in 117
Dict1 for Different Values of Level1 and D

CHAPTER 1

INFORMATION RETRIEVAL FROM A TEXTUAL DATA BASE

1.1 TEXTUAL DATA BASE

A textual information retrieval system may contain files that consist of a collection of documents together with some scheme to delimit and access the individual documents within the file. For example, in a file that represents journal articles, each document might contain the author names, title, abstract, and so forth. Textual data bases often contain contexts (sentences, paragraphs, documents, etc.) and retrieval keys that consist of suitably chosen words or portions of words. The contents of a textual data base are order dependent and allow title encoding of the data. Very little formatting is necessary, although for considerations of efficiency some special flags may replace normal typography to indicate the start or end of a context (such as replacing a number of leading blanks on a line with a mark to indicate the start of a paragraph, or replacing a period with an end of sentence mark so that decimal points are not misinterpreted).

Access to textual data bases may be through keywords and phrases in text stored on random access devices. A common retrieval mechanism is through use of an inverted file of all the content words in the textual data base. In the inverted file each content word is associated with a document pointer. The

operations performed during search of a textual data base generally result in the formation of progressively smaller subsets of the data base until the number of documents is small enough to be examined manually by the user. The operations are selected by specification of search questions, each of which is an expression consisting of logical operators that act on pairs consisting of an attribute and a term.

As data bases increase in size, the data management problems associated with maintainance of the inverted file become more complex, and rapid retrieval of data becomes more difficult. For extremely large textual data bases the rapid retrieval of data by use of convential digital computers is not practical, and some form of specialized computer architecture must replace the general purpose computer. Hollaar and Stellhorn [1], and many other authors, have suggested a computer configuration that might be employed for retrieval of textual information from very large data bases.

1.2 FILE STRUCTURE

As described by Heaps [2] the textual data base discussed in the previous section may be envisaged as a set of records in which each record contains information about a single document, report, or journal article. Each record is divided into fields, and each field is associated with a particular label, or attribute, such as author names, title, abstract, keywords, and

so forth. The value of the attribute is stored within the field, and consists of a subset of "terms" chosen from a set that forms the "vocabulary" associated with the particular attribute. Some fields may be searched for the presence of terms as specified in a question statement. Each searchable field is associated with a particular attribute. There are many different ways in which such data bases may be structured for storing on computer accessible files. In general, various file organizations have been designed with a view to improving some aspect of performance. An evaluation and selection of file organizations has been discussed by Cardenas [3]. Also, several different search techniques may be used to retrieve individually identified data records from a computer memory space [4,5].

One of the simplest ways to locate an identified record stored in a sequential file is to scan the records of the file, comparing successive record identifiers with the identifier being sought. When each of N records is likely to be retrieved with equal probability the scanning requires an average of (N+1)/2 indentifier comparisons. The number of secondary memory accesses, and therefore the time required, to locate a data record is a function of the relative size of a record and a block of storage. If the records are stored in sequence (with respect to some field) on a direct access medium or in core memory there are various techniques that may be used to reduce the time of the search. For example; if a binary search is applied then for large values of N the mean number of probes approximates

LOG2(N-1). Also, if the most frequently used keys are in the higher positions in a data base the search may start at the top and work down with a resulting improvement in speed. However, it should be emphasized that the time required to sort the file in order to use such tricks may not pay off in overall program execution time. However, for small or infrequently searched files, or in situations where the keys involved are not collated in the natural sequence required by the computer being used, it may be the fastest technique. Also, the sequential search may often be used to advantage in conjunction with other techniques.

It is often desirable to store and retrieve records using more than one key. This is most easily accomplished by sorting the data in duplicate files, with the first file sorted by one key and the second by a second key. Such duplication, however, uses twice the storage space and involves double the maintenance cost, since records to be updated must be accessed and changed in both files.

1.3 INVERTED FILE ORGANIZATION

The inverted index file is a frequently used file structure for the storage of indexing information in document retrieval systems. In the inverted index organization each search key appears in a term dictionary along with a pointer to a set of document numbers. A dictionary term may then be used to indicate the beginning of the entries associated with each searchable term

as shown in fig. 1.1. The complete file may also be stored sequentially. This approach simplifies the searching of the index required by prefix and suffix operations, and provides information to optimize the order of merging the lists that arise in searches that involve multiple terms.

The inverted file structure allows the user to combine (using operators such as AND, OR, and AND NOT) lists of entries until there is formed a subset of the data base that is small enough to be output to the user. An inverted file structures also allows more rapid retrieval of information at the expense of the increased amount of storage required to hold the data base since, in fact, a sequential file is also required in order to retrieve the detail of any stored document item. The storage overhead depends on the level of the inversion. If the file is inverted at the document level then very little overhead is required, and the time necessary to merge lists will be small. However some full text searching will be required if the query asks for the co-occurrence of two terms in the same sentence or paragraph. If inversion is at the word level then no full text searching will be required, but the inverted file will probably be larger than the original data file, and the time required to merge the lists will be high.

A typical formulation of the inverted file approach makes no attempt to organize records in the sequential file according to expected usage. The implicit assumption is that references to sequential file addresses are uniformly distributed, and the cost

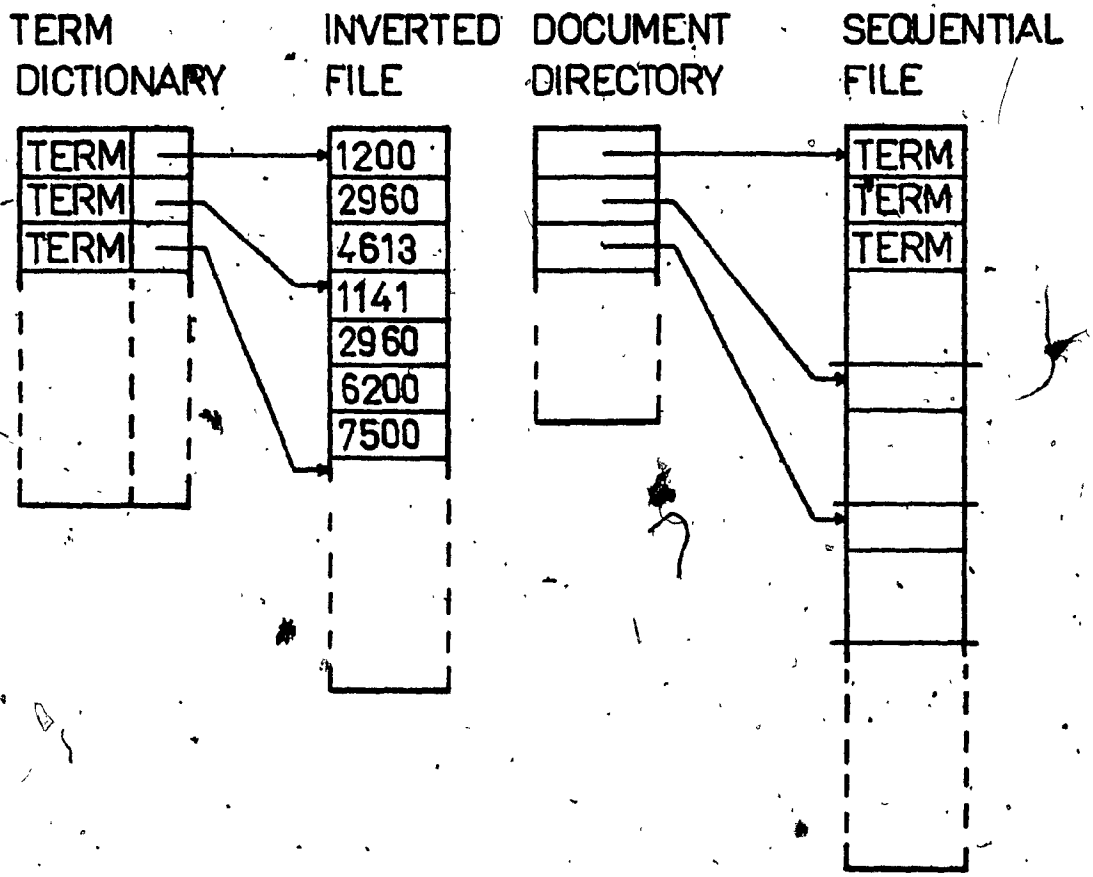

FIG.1.1 : INVERTED FILE ORGANIZATION

of retrieving the records is independent of the address value. This assumption is accurate for a truly random access device such as a core memory. However, few practical applications involve sequential files that can reside entirely in core. In fact the more typical situation is one in which only a small fraction of the main file can be core resident at any time. The rest of the file must be located on a secondary storage device characterized by a much longer access time. Thus the assumption of cost homogeneity should usually be replaced by a cost function in which a sequence of local accesses is far cheaper than a sequence of scattered references. Rothnie and Lozano [6] have considered attribute based file organization in a paged memory environment.

The inverted search procedure is believed to be particulary suitable for online search since the question may be formulated, the expanded question may be examined, and the number of relevant documents may be ascertained. In order to search with respect to a given question each question term must be searched for in the dictionary, and the corresponding lists of the inverted file must be read into main memory. The question logic must then be performed on these lists in order to determine the list of document numbers relevant to the question. Each question term can be specified in untruncated form or truncated form. When specified in truncated form a question term consists of a stem preceded, and/or followed, by an asterisk to indicate that there is no restriction on whether the stem is preceded, and/or followed, by other characters. Truncation is often useful in

document retrieval systems since, for example, a search for compute* as a title term may find compute or computer or computers which otherwise might have to be included separately in a given request.

1.4 FILE COMPRESSION

As described in section 1.3 retrieval can be done efficiently with the use of an inverted file organization, but it increases the amount of storage required to hold the data base since a sequential file is also required to retrieve the details of any stored document item. It is possible to reduce the total storage by coding each key in the key directory and storing a sequential data base that contains the coded representation of each key. The coded sequential file shown in fig. 1.2 occupies less space than the sequential file shown in fig. 1.1, since codes may usually be stored in less space and in fixed fields.

The space required for storage of the four files of fig. 1.2 may be estimated as follows: Suppose R document records in a given sequential data base contain a total of N terms of which only D are different. The required term dictionary then contains D terms, each of which has a pointer whose value is not greater than N, and hence the space required for storage of the dictionary is

$$D[L+CEIL(LOG2\ N)]$$

bits where L indicates average term length in bits. The inverted

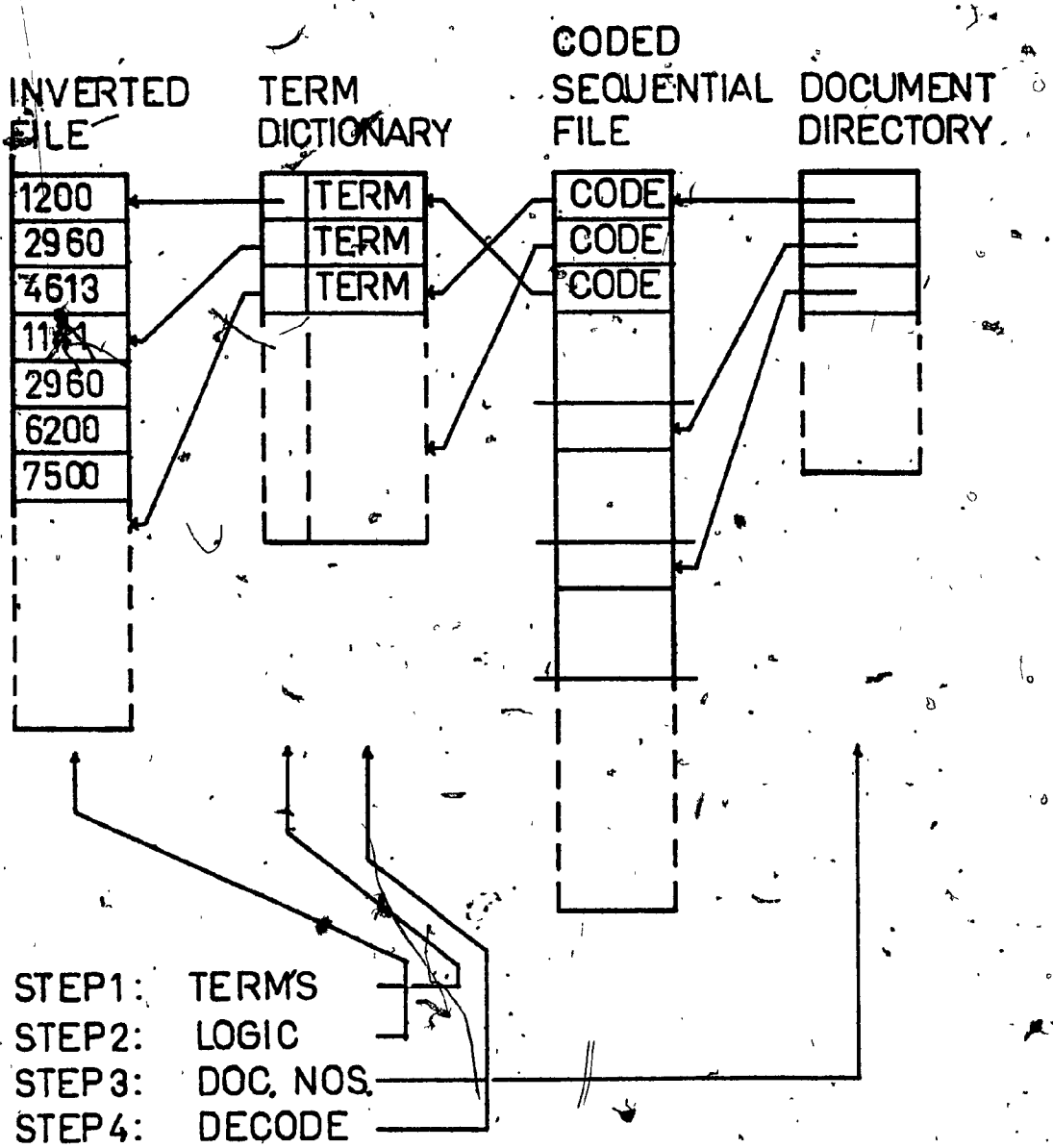

FIG.1.2 : INVERTED FILE ORGANIZATION WITH CODED DATA BASE

file then contains N numbers, each of whose value is not greater than R, and so may be represented in CEIL(LOG2 R) bits. The space required for the inverted file is thus

$$N\ CEIL(LOG2\ R)$$

bits. Finally the space required for storage of the document directory and the sequential file is

$$R\ CEIL(LOG2\ N)+N\ CEIL(LOG2\ D)$$

bits. The total space required to store the four files of fig. 1.2 is therefore

$$DL+(R+D)\ CEIL(LOG2\ N)+N[CEIL(LOG2\ D)+CEIL(LOG2\ R)]$$

In contrast, the space required for storage of N terms in a purely sequentially structured data base is

$$NL$$

bits exclusive of any space required to store tags or delimiters, such as blanks, between terms. The space required for storage of the four files of fig. 1.2 may be less than the space required for sequential storage in uncoded form. However the ratio of the required spaces depends on the characteristics of the data base.

A further characteristic of many document data bases is their similarity with respect to growth of vocabulary. As the data base grows, the number of the dictionary entries increases, and there is no upper bound to the size of the dictionary. This has been proved analytically by Mandelbrot [7], and it may described quantitatively in the following manner. If N is the total number of terms in the data base, and D is the number of different terms in the data base, then

$$D = K*N**B$$

where B and K are constants that depend on the particular text and $0 < B < 1$.

For a large data base only a small fraction of the term dictionary can be stored in core. The rest of the file must be located on a secondary storage device. It is advantageous to store the most frequently referenced terms in core in order to minimize the expected retrieval time. It is also well known, Zipf law [8], that if a sufficiently large sample of general English text is examined, and the distinct words are ranked in descending frequency of occurrence, then their frequency is related to the rank as follows:

$Pr = A/r$

where Pr indicates the probablity of occurrence of the word of rank r, and A is a constant.

If there are D different words the value of A may be determined from the formula

$$A = 1/(1 + 1/2 + 1/3 + 1/4.....+1/D)$$

which may be approximated in the form,

$$A = 1/(LN\ (2D + 1)).$$

As described by Booth [9], one of the consequences of the Zipf law is that it implies that 50% of the different words occur only once in the entire data base, 16% occur only twice and $100/[N(N+1)]$% occur only n times. Of course coding of terms that appear only once in the data base does not save storage space and, in fact, for such terms there is space wasted through

storage of codes. Thus coding the 50% infrequent words tends to increase the size of the dictionary required for coding and decoding of the sequential data base. It is therefore desirable to consider alternative schemes with a view to achieving maximum possible compression of the data base with a dictionary of moderate size.

A greater saving in space would result if terms in the sequential file were replaced by their Huffman codes [10]. Huffman coding, based on statistical characteristic of a file, provides the most compact way to code terms. However, since Huffman codes are of variable lengths, they require relatively complicated procedures for the operations of coding and decoding. Thus it is important to investigate suitable codes to obtain a data compression ratio close to that of the Huffman codes while allowing more rapid decoding of the stored data. A purely fixed length code provides a convenient way of decoding, but has the disadvantage of requiring as much space for storage of frequent term as for infrequent ones.

A coding scheme designed to combine some of the advantages of the Huffman codes with some of the advantages of fixed length codes has been described by Heaps and Thiel [11], and subsequently analyzed by Heaps [2]. This coding scheme may be said to be of restricted variable length, and is based on the frequency of occurrence of words in a document data base. The coding scheme may be described as follows:

Let N1, N2, ..., Nk be a sequence of increasing positive

integers, and suppose the most frequent 2**(N1-1) terms in the dictionary are assigned different codes, each of N1 bits, of which the first bit is always set equal to 1. The next most frequent 2**(N2-2) terms in the dictionary are assigned different codes of length N2 bits in which the first bit is always 0 and the (N+1)th bit is always 1.

Similarly, the next most frequent 2**(N3-3) terms are assigned N3 bit codes in which the first and (N1+1)th bits are always 0 and the (N2+1)th bit is always 1. The codes will be assigned in this fashion and there may be up to

2**(N1-1)+2**(N2-2)+2**(N3-3).....+2**(Nk-k)

such dictionary terms. All terms in the sequential data base may be replaced by their codes assigned as above.

Suppose the frequencies with which terms occur in the sequential data base is according to the Zipf law. It has been shown that three codes of length between 4,8,12..... and 6,12,18..... are the most compact, and have efficiencies of about 92% in comparison to the Huffman codes [10]. Adoption of codes of length 8, 16, and 24 bits is particularly convenient for use with a computer that is oriented toward storage of 8 bit characters. For a computer with a 6 bit representation of characters it is more convenient to use codes of length 6, 12, 18 and 24 bits.

The restricted variable length codes are also preferable to the Huffman codes in that they may be used to address storage

positions in the term dictionary. For example, if the 8 bit codes are used the last 7 bits may be chosen as the dictionary locations of the most frequent 128 terms. Similarly, with the 16 bit codes the bits 2 to 8 and 10 to 16 may be used as the dictionary location of the 2**14 = 16,384 next most frequent terms, and so forth. Huffman codes do not range through all possible values and hence are not convenient for use as dictionary addresses.

The use of restricted variable length codes allows all terms of the same code length to be placed in alphabetic order in several dictionaries. A dictionary that contains the most frequent words may be stored in computer memory, and the dictionaries of very infrequent terms may be stored in secondary storage. In such a storage scheme each question term must first be searched for in the first level, if not present it must then be searched for in the second level, and if still not found it must be searched for in the next storage level, and so forth. Thus efficient compression of the sequential file is gained at the expense of additional time required for search of the dictionary for the presence of question terms. Further improvement could be obtained if in each dictionary all terms of the same length are arranged in alphabetic order in a term string. Such additional segmentation allows saving in storage and reduction in the time required to search the dictionary for question terms specified in untruncated form.

With the dictionary segmented as described above the sequential file may be stored in such a way that is close to the most compact that could be achieved by use of Huffman codes. The scheme also allows the dictionary to be stored without inclusion of delimiting characters between terms.

Dimsdale and Heaps [12] have described the use of restricted variable length codes with a dictionary arranged and accessed by a virtual hash storage scheme.

As mentioned by Heaps [13], although the restricted variable length codes allow good overall compression of data they have several disadvantages. The first disadvantage is that in many document data bases approximately 50% of all different terms occur only once. For such infrequent terms there is no saving in storage and, in fact, space is wasted through storage of the codes.

The second disadvantage is that word-encoding requires a large coding dictionary that is highly data-base dependent and grows with each update. Hence encoding and decoding are slow in operation and expensive in storage requirements. Also, many dictionary words have character strings in common, and it may appear wastful of space to store in their entirety each of a set of terms that contain a common stem.

It is therefore of interest to consider the possibility of coding text by means of text fragments other than complete words or search terms.

Thus, an alternative approach to dictionary storage involves the recording of common character strings as single entities that may be called text fragments. The fragments may be stored in a dictionary with pointers to an inverted file. The sequential data base may then be expressed in terms of the fragments, which are then coded to form a compressed data base [13]. There is a considerable body of literature on data compression by use of fragments.

Colombo and Rush [14] have proposed the use of word fragments as the basic language elements of a data base for information retrieval. Clare et al [15] considered the representation of text by means of equifrequent text fragments in which the fragments may overlap word boundaries rather than be constrained to fragments from within single words. A method of selection of equifrequent text fragments has been described by Schuegraf and Heaps [16].

The use of equifrequent text fragments has several advantages. The first advantage is that each fragment contains the same amount of information content, which implies that maximum compression may be achieved by use of fixed length codes. A further advantage is that the equifrequent fragment oriented inverted file has the same number of entries for each item; this permits organization of the inverted file as a collection of fixed length records.

Mayne and James [17] have discussed a technique for data base compression that treats the more frequent sequences as "common factors". The common factors are then recorded in condensed form. Clearly, if a number of different sequences of several characters occur frequently in the text there is the possiblity of achieving considerable compression by coding such strings. Thus coding of word fragments instead of words allows a spectacular reduction in the size of the dictionary, but leads to less efficient compression and also increases the size of the inverted index.

In use of text fragments each term in the data base is represented as a concatenation of fragments. The same algorithm then should be used for representation of question terms. It is clearly undesirable to include any inverted file entries for single letter fragments. Fragments may also appear in unwanted combinations, but the retrieved unwanted terms may be eliminated by performing a sequential search on the output. This causes an increase in the time required to process each question. Different coding schemes for compression of text in document data bases for information retrieval purposes have been surveyed by Heaps [13].

The compression techniques discussed in this section are all designed to reduce storage space for data files, but the reduction is usually at the expense of increased CPU activity needed for compression and decompression. As CPU time becomes cheaper relative to the cost of external storage divices,

compression appears as an increasingly attractive option for dealing with large files.

1.5 OBJECTIVES OF PRESENT INVESTIGATION

With use of an inverted file, in order to retrieve an item the term dictionary is searched for the keyword, and the corresponding entry in the inverted file is extracted to give the addresss in the sequential file of all the records that satisfy the request. The most time-consuming part of this retrieval process is the search of the dictionary, and several methods have been devised to minimize this. The critical parameter is the number of accesses to secondary storage. One procedure is to narrow the search down to a group (known as a block or bucket or page) of keys, which can be searched rapidly in core. The size of the blocks is selected to be the same as the size of the unit of transfer between primary and secondary storage. The time taken to search the block in core may be assumed to be negligible compared with the time to access the block on disk.

If truncation specifications are allowed in the question then a question term such as compute* groups words such as compute, computer, computers and computerize which otherwise might have to be included separately in a given request. An alphabetic ordering of the terms in the dictionary may prove convenient since it allows use of a binary search [4]. Although the above scheme allows quite fast retrieval of a term through a term

dictionary, it suffers from a number of disadvantages. One of the most serious disadvantages arises from the fact that if truncation is allowed in the specification of question terms, then a binary search of an alphabetically ordered list of terms is generally not convenient since a binary search for a question term such as compute* may be required to range through many terms that begin with c. Left truncation is considerably more difficult to deal with.

A second disadvantage is due to the fact that with an alphabetically ordered term dictionary any insertion of a new term requires an update of the term dictionary and the inverted file. If the sequential file is stored in coded form then the entire set of files, including the sequential file, should be updated at each insertion.

A further disadvantage is that for a large term dictionary, which cannot reside entirely in core, the average number of disk accesses will be increased. Furthermore, perhaps the most important factor affecting the overall performance of an inverted file structure is the manner in which the indexes and associated lists are physically organized and managed in storage. In large, highly inverted data bases the term dictionary becomes a large data base in itself; contrary to the usual assumptions the dictionary should not then be implemented as a sequential file [18]. In the case of the simple sequential organization the dictionary search using a highly inverted large data base with many different key-values requires many I/O accesses, and a

binary search cannot be performed entirely in core. Use of a large dictionary intended to reside in random access external storage devices requires that parts of it be brought into core as needed. It is therefore of interest to consider the possibility of alternative techniques for structuring a term dictionary by means of a hashing function. Such a choice has the advantage that each question term may also be operated on by a hashing function and, if present in the dictionary, the corresponding pointer to the list in the inverted file may be determined readily.

An important problem in a document retrieval system is the structuring of the files in a manner: a) To occupy a minimum of space; b) To minimize the number of file accesses in each question term.

While the use of the hash storage scheme allows random access file organization and rapid access of an entry in main memory, an efficient method of compression coding is also possible through the use of restricted variable length codes. Thus, instead of the codes being chosen to correspond to the addresses at which terms are stored in an alphabetically ordered dictionary, the codes in the sequential file may point directly to the location of terms in the dictionary.

Use of a virtual scatter storage scheme for dictionary look-up, has been described by Murray [19].

Grimson and Stacey [20] have considered two techniques for structuring the term dictionary. A division hash code and a hybrid technique (which is a combination of a hash and a tree) were compared for dictionaries of from 10,000 to 100,000 unique keys. In the final analysis of which of the two methods is preferable they conclude that it would depend on the application environment.

Use of a hash storage scheme, and structuring of a file, for an online library catalog of one million volumes has been described by Dimsdale and Heaps [12].

However, a serious disadvantage of hash storage is that the hash address of a truncated question term bears no relation to the hash address at which the untruncated term is stored. But in a document retrieval system it may be desired to allow left and/or right truncation, and this may complicate the retrieval process. To facilitate processing of a question term that is specified in truncation form a two dimensional hash storage scheme, as described in chapter 3, may be used.

The use of two dimensional hash addressing for dictionary look-up, as proposed in the present thesis, has the following advantages:

1- It allows question terms to be specified in truncated form.

2- It allows fast response to questions.

3- The files are organized in a manner to allow for ease of updating. As further document entries are added it is necessary

to add additional pointers from the inverted index. Also, whenever a new key occurs in a document entry it must be added to the dictionary, assigned a code for storing in the compressed sequential file, and entered into the inverted index.

4- Efficient use of storage space is obtained by use of restricted variable length codes in the sequential file. Furthermore, the method uses a scatter index table in which data records are stored in a separate storage area [5,21], and space for each entry is allocated only as needed.

In storage of the term dictionary two levels of storage are used [22,23], and terms are loaded in order of decreasing frequency of occurrence. Thus the most frequently referenced terms are located in core, the rest of the file being located on a number of fixed sized blocks on secondary storage. This tends to reduce the average number of auxilary storage accesses required to retrieve a key from the dictionary.

In the present thesis the emphasis is on the structuring of a large dictionary in hierarchical storage by means of a two dimensional hash storage scheme.

CHAPTER 2

KEY-TO-ADDRESS TRANSFORMATION TECHNIQUES

2.1 SCATTER STORAGE

In scatter storage tables a key, which is associated with the desired entry, is used to locate the entry in the stored table. This key is transformed into an addresses location, preferably in such a fashion that the addresses are uniformly distributed in the set of available addresses. The resultant address is called a hash address, and the transformation is said to constitute hash coding. Items are entered into a table by use of an index, or address, which is computed for the item by means of some hash transformation. Provided no two items have the same hash code, the searching and insertion are each performed in a single step, regardless of the size of table. When two items have the same hash code a collision is said to exist. There is no method which can be certain to always produce a satisfactory address on the first attempt. Therefore, a method is needed for resolving the collision of keys, and several such methods will be discussed in section 2.4. The process of transforming a record key, or external address, to the corresponding internal address, which gives the location in the file where the record should be found, generally consist of two parts as described in the following sections 2.2 and 2.3.

2.2 KEY TO DIGITS CONVERSIONS

First, the key is changed to a format best suited to the arithmetical capability of a specific computer. Since some of the key sets are alphabetic or alphanumeric, and since nearly all transformation methods operate only on numerical values, it is necessary to encode the alphabetic or alphanumeric keys as numeric keys. The conversion should be effected without causing any loss of information in the key. In the present investigation conversion of the key into digits has been tried by using three different methods as follows:

METHOD A

Each symbol in the descriptor is replaced by the two-digit decimal number that represents its display code. Any leading zeros are deleted, and the digits are concatenated to form a decimal number. For example, for the term COMPUTE the characters have the display codes C=03, O=15, M=13, P=16, U=21, T=20, E=05, and the resulting decimal number is 315131621205.

METHOD B

The key is converted into the number that has the same binary representation in a computer word. Thus the term COMPUTE has the binary representation

000011 001111 001101 010000 010101 010100 000101

which also represents the decimal number 3x2**36+15x2**30+13x2**24+16x2**18+21x2**12+20*2**6+5. This method produces a large key length. Clearly, hashing functions that involve shifting and folding tend to be more appropriate for large keys. It may be noted that if the key contains only one character then both methods produce the same number.

METHOD C

The method that has proved to be the best throughout the experiments is to encode 0,1,....,9,a,b,.....,z as decimal numbers 00,01,....,09,11,12,.....36. For example, KNUTH and HZS630 are encoded as 2124313018 and 183629060300 respectively. The uniqueness of the alphabetic key is preserved by this encoding.

In the external set all keys are distinct, but after a key to digits conversion the distribution of keys over the entire range of keys is a random distribution. The ideal distribution of keys throughout the range of keys is a uniform one in which the difference between any pair of successive keys, taken in ascending order, is constant. There is also a special case of a random distribution in which the uniqueness of the modified keys (after key-to-digits conversion) is preserved although keys are not distributed uniformly. The distribution of the modified keys affects the performance of the subsquent hash transformation. For example, the hash transformation with a uniform distribution

gives the best performance. Its performance for the special case of a random distribution is better than that for the random distribution.

In method C of key-to-digits conversion the uniqueness of the alphabetic key is preserved, while in the two other methods the uniqueness of modified keys is not preserved.

2.3 HASHING FUNCTIONS

The number of digits or bits of the (modified) key is reduced in order to be appropriate to the length and range of the numbers that can be used as file addresses. The best hashing function is one which will map N identifiers into N storage slots and produce no synonyms. However no known method of transforming keys to addresses completely avoids the problem of collision caused by two distinct keys being hashed to the same address. The truth of this fact destroys some of the simplicity of the hashing scheme. In practice, the following properties are desirable for a key-to-address transformation.

1- The algorithm should be simple and computable fast. This leads to a short quickly written, easily understood, and rapidly executable program.

2- It should randomize the assignment of identifiers (i.e., keys) over the entire available memory space.

3- It should distribute the key set uniformly across the address space.

4- The number of collisions should be minimal.

Properties 1, 2, and 3 ensure that the time of a retrieval operation be minimal, which is a principal system design objective.

Several hashing methods have been proposed. The hashing functions fall into two classes: distribution-independent and distribution-dependent functions [24,25]. A distribution-independent hashing function may be applied without specific knowledge of the initial distribution of items. A distribution-dependent hashing function, on the other hand, is obtained by examination of the subset of keys that correspond to known records.

In the present experiment two common distribution-independent functions are used, and are as described in the following two sections.

2.3.1 DIVISION METHOD

A well known, and frequently used, distribution-independent hashing function is the division of the key by a positive integer approximately equal to the number of available addresses. In this method the remainder obtained from the division becomes the address for the key. It has been noted that one of the possible advantages of this method is that any sequential properties possessed by the key set will be preserved through the division,

resulting in more uniform than random distribution of addresses [26,27]. It may be noted that if table sizes of multiples of 2 or 10 are used, then extraction may be superior to division. In the extraction method a value of k bits, or k digits, is obtained by extracting k bits, or k digits, from the key and concatenating them in some manner. Extraction methods are very restrictive because the convenient number of bits or digits extracted is restricted to values of the form 2**k or 10**k respectively, while the division hash function allows the hash table to have almost any length. But, as mentioned in section 2.5, an even divisor gives poor results.

2.3.2 MIDSQUARE METHOD

In application of the midsquare method the key is multiplied by itself, and the address h(x) is obtained by one of the following schemes:

a) Binary truncation

h(x) = a value obtained by truncating bits at both ends of the product until the remaining number of bits is equal to the desired address length. The truncation method is very restrictive because the size of the table is restricted to values of the form 2**k.

b) Decimal truncation

In this method the address is obtained by truncating digits at both ends of the product until the number of remaining digits is equal to the desired address length. In this method the address

ranges are limited to multiples of 10. However if address ranges are not multiples of 10, then the result is adjusted to fit the range of the bucket address. For example if truncation is used to produce 3-digit numbers, and there are 139 buckets, then each number may be multipled by .139 to reduce it to be in the range 000 to 138. However the midsquare method tends to produce a better result when table sizes of multiples of 2 or 10 are used. Since any form of truncation or compression reduces the amount of information it may destroy the uniqueness of the key set and therefore worsen the distribution.

For example suppose that 7 buckets, numbered from 0 to 6, are used. Then one decimal digit should be extracted from the product. It is assumed that all digits from 0 to 9 have equal probabilties of occurring as the extracted digit, then the distribution of addresses through the entire address space will be as follows.

extracted digit	address
0	trunc(0x0.7)=0
1	trunc(1x0.7)=0
2	trunc(2x0.7)=1
3	trunc(3x0.7)=2
4	trunc(4x0.7)=2
5	trunc(5x0.7)=3
6	trunc(6x0.7)=4
7	trunc(7x0.7)=4
8	trunc(8x0.7)=5
9	trunc(9x0.7)=6

Thus addresses 0, 2, and 4 occur 2 times and their probability of occurrence is 1/5, but the probablities of occurrence of other addresses are 1/10. Thus the result will be a non uniform distribution, while a table of size 2**k or 10**k (k is an integer) allows effective use of binary and decimal truncation methods respectively.

Many more methods for choosing hashing functions have been suggested, but none have proved to be superior to the simple division and midsquare methods described above. A survey of some other methods, together with detailed statistics on their performance when applied to actual files, is presented by Lum [27].

2.4 COLLISION RESOLUTION TECHNIQUES

Once an acceptable transformation has been selected in order to map identifiers into home locations in a hash table one must decide on a method for storing and retrieving those records in which the home location is filled with other records. There must be some way of distinguishing an empty slot in the table from a non-empty entry. For instance, if zero can be excluded as a valid key, then a zero key can be used to mean that an entry is empty. Of course the entire table must be initalized to whatever state is used to signal empty slots in the table.

The two basic techniques commonly adopted to accommodate overflow records are: (1) Storage of them in vacancies in another

bucket located in the prime area, and (2) Storage of them in a separate, or independent, overflow area.

Many variations are possible in each basic technique. A possible version of the separate overflow method is through chaining in which the location of the first overflow record from each bucket is listed as the first pointer in a chain of pointers to addresses in other records. An example of the separate overflow technique is the provision of several small areas that can be used to store only overflow records from a particular section of the prime area. Overflow records that cannot be stored there are stored in a larger independent area that is available to all buckets. If overflows are stored in the prime area, rather than in a separate overflow area, the hash algorithm must make a sequence of probes into the table. The sequence terminates when a probe selects either the match for the key or an empty location. Alternatively, at some stage the sequence may recycle and the table be found to be full. The sequence of operations constitutes a search procedure, and the number of entries that appears in a sequence from a particular initial position before any entry is encountered twice is called the period of search. The period of the search should be as large as possible and preferably equal to the table size. Otherwise, the table can appear to be full when there is still space available.

In general a good search procedure should satisfy two requirements:

a) Its computation should be very fast.

b) It should minimize collisions.

Property (a) is somewhat machine-dependent, and property (b) is data dependent. Some search techniques are affected by primary clustering. If H(K) denotes the calculated address for the key K then the primary clustering could be eliminated for every

$H_i(K) = H_j(K')$, $i \neq j$

provided

$H_{i+L}(K) \neq H_{j+L}(K')$

where L= 1, 2, 3,.....

so that the above two expressions do not trace out the same sequence.

However another kind of clustering still remains. If in a search

$H_i(k) = H_i(k')$

then

$H_{i+L}(K) = H_{i+L}(K')$, L = 1, 2, 3,.....

and two keys which are transformed initially to the same location will cause a trace through the same sequence of locations; that is, those keys which hash initially to the same location from a cluster.

There are basically three search techniques; namely linear search, random search, and quadratic search. In the following three sections these search methods are described and the performance of each is discussed.

2.4.1 LINEAR SEARCH METHOD AND ITS EXTENSIONS

The simplest open addressing scheme, known as linear probing, is that in which the collided item is placed by searching linearly forward in the table. Thus if k is the hash code, then locations k+1, k+2,..... (modulo the table size) are examined in that order. Such an approach is slow because the items are clustered to be adjacent. One possible alternative is to use a multiple of an increment a, so that the locations are k+a, k+2a,..... (modulo the table size). This method is also slow, and for the same reason. The condition that a and k are coprime ensures that the period of search for the linear hash method is equal to the table size m:

As is well known, the linear search is badly affected by primary clustering. In a linear search, if $H_i(K1) = H_j(K2)$, $i \neq j$, then $H_{i+L}(K1) = H_{j+L}(K2)$, and the searches for K1 and K2 proceed through the same positions, even though $H_0(K1) \neq H_0(K2)$, where H_0 denotes the first calculated address. Thus the average length of search rapidly increases for high values of the load factor. In the uniform hashing as proposed by Peterson [28] the effect of primary or secondary clustering is ignored. In fact the keys are transformed into random locations of the table, so that each of the m!/(n!(m-n)!) possible configurations of n occupied cells and m-n empty cells is equally likely. This model ignores any affect of primary or secondary clustering; thus the occupancy of any cell in the table is supposed to be essentially independent of

the occupancy of other cells. It has been shown by Knuth [29] that the average number of probes required for uniform hashing is (m+1)/(m-n+1) for 1 <= n < m.

Such uniform hashing does not exist in practice, and the linear search is indeed affected by clustering.

Luccio [30] has presented a linear search procedure for a hash table whose increment step is a function of the key being addressed. It applies to the special instance of a table of size n = 2**r. It allows full table searching and practically eliminates primary clustering at a very low cost.

The formula used by Luccio for computation of a home address is

Hi(k) = H0(k)+Q(k)*i (mod n), i=1, 2 . . .

where Q(k) is an integer function of the key. Then the increment step is generally different for different keys. To ensure that the entire table is searched, then for any k the value of Q(k) must be coprime with n. This may be ensured by setting:

Q(k) =(2.P(k)+1) mod n

where p(k) is an integer constant that is extracted from the key by some hashing method. In particular, the hash address H0(k) can be substituted for P(k), and the increment law then becomes

Hi(k) = [H0(k)+[2H0(k)+1] (mod n*i)] (mod n) i=1, 2, . . .

This formula need be evaulated only once, and any address H(i+1)(k) in the sequence may be computed from the preceding one by a single addition.

Bell and Kaman [31] present a linear quotient hash code. In this method the length of the table is limited to a prime number, and the only difference from the linear search is the change in the offset of successive probes from a constant (1 location) to a variable (q locations), where q represents the quotient created by the division of the key by the table size. The degenerate case for which q is congruent to zero is removed by changing q to 1 when such occurs. In this method every table location will be traced exactly once in the first n (table size) steps. The linear method is one of the simplest method in common use, and in general it is fair to state that the linear quotient is one of the methods that requires the least time per probe and the fewer probes per search.

Brent [32] has described a method for reducing the retrieval time of scatter storage that is applicable in applications where most entries are looked up several times and deletions are rare. The method is preferable to the linear quotient method and is competitive with direct chaining.

2.4.2 RANDOM SEARCH

Elimination of primary clustering has also been achieved by altering the equidistance of successive addresses in the sequence. If the table size is n = 2**k, the random search may be applied (Morris) [21], where the increments Pi of successive probes are generated by an ad hoc pseudorandom number generator.

The following method works well when the table size is a power of two (n = 2**k for some k)

1- Intialize an integer r to 1 when the routine is called.

2- To calculate each Pi, perform the following on the current value of i

a) Set r=r*5

b) Mask out all but the low-order k+2 bits of r and place the result in r

c) Take the value in r, shift it right by 2 bits, and use the result as the value of Pi

This method generates values of Pi that range over every integer from 1 to n-1 exactly once. When the generator runs out of integers, the table is full and the entry cannot be made. This method depends on multiplication and is much slower than the linear search.

2.4.3 THE QUADRATIC SEARCH AND ITS EXTENSIONS

A more interesting method, which is faster than the random methods and yet avoids the clustering problems, is the quadratic search. In effect, this method tries an address k+Ai+Bi**2 (modulo the table size) for I = 1, 2, 3, . . Where k, a and b are fixed. Maurer [33] claims that when the table size is a power of 2 the period of a quadratic search is usually too small for effective use, and when the table size is a prime a quadratic search always covers exactly half of the table. Radke [34] has pointed out that when the length of the table is a prime of the

form 4n+3, where n is an integer, the whole table may be accessed by two quadratic searches plus a separate access for the original entry point. When a collision occurs the search is first made by use of an increment a given by the quadratic function

a) a = (i**2+k) (mod p)

and if this function fails then the search is made by means of the quadratic function.

b) b = p+2*k-(i**2-k) (mod p).

An alternative search method has been presented by Day [35] for a table size of the form 4n+3. This method is computationally simple, and yet accesses all the table in one sweep. The algorithm is given below, where p is the table size and j is the calculated hash address.

(a) Set I to -p

(b) Hash the datum to value j

(c) If location j is not filled, make the entry and stop; otherwise set I = I + 2

(d) Set j = (j+abs(i)) mod p

(e) If I < p return to (c) otherwise the table is full.

In the above method, a test for a location being filled is made only once. The increment abs(i) is never larger than p-2 and it is always positive.

Use of the quadratic search method eliminates the worst "primary" clustering; however it is still affected by secondary clustering. If two elements have the same address, then the same sequence through the table is traced for both. Secondary

clustering can be overcome by means of the quadratic quotient search proposed by Bell [36]. He has modified the quadratic search from

Hi(k) = H0(k)+Ai+Bi**2

to the following

Hi(k) = H0(k)+Ai+B(k)i**2

where the function B(k) denotes the integer value of the quotient (k/n) used to compute H0. Clearly the use of differing quotients provides a good way to distinguish between different keys with the same remainders. Quadratic quotients search through exactly half of the table before returning to the intial hash address.

Hopgood and Davenport [37] have pointed out that, contrary to what is normally assumed by Maurer [33] (1968), in the quadratic hash method a table size equal to a power of two may be used without the usual disadvantage that the period of search is significantly less than the table size. The quadratic hash method avoids primary clustering by defining the ith position in a sequence as:

k + Ai + B*(i**2)

The computation required for the quadratic search method can be reduced by defining

r = A + B and q = 2B so that the algorithm is

1- Calculate k = i(k), j = r

2- If the kth position is empty or contains k then the search is concluded

3- Otherwise set k = k + j, j = j + q and go to 2.

In the special case that q=1 the period of the search is m-r+1 and the complete table is searched. The period of search in such a case is considerably better than the case where m is a prime number. Also, when m is a power of 2 additions modulo the table size can be achieved in most computers by masking off the desired number of bits. The algorithm takes a particularly simple form if C_q in step 3 of the above algorithm is set equal to 1. An unusual property of the method is that the period of search is reached when the ith and i+1th entries are the same.

Ackerman [38] found that the quadratic method can be applied to tables of size p**n for any prime p. It is shown below that rather simple conditions on the coefficients suffice to guarantee that all table locations will be examined once and only once. Specifically, if the equation is

k+Bi**2+Ai mod p**n

where k is the initial hash address and 0 < i < p**n then, if p divides B but not A, a full table search is ensured. These conditions degenerate to a linear search when n=1. In this case one can follow Maurer's [33] suggestion and examine at most half the table positions, or alternatively use Day's [35] technique to cover the whole table. To prevent secondary clustering for the n>1 case, b may of course be varied by choosing it to have the form Pk' where k' is a function of the key and is independent of k'.

Batagelj [39] has extended the use of the quadratic search for various table sizes without the usual disadvantage that the

period of search is significantly less than the table size. He finds that a necessary and sufficient restriction on the table size d for the existence of a full-period quadratic search is that either the factorization of d contains at least one prime power or that d is equal to twice the product of distinct primes.

2.4.4 OTHER COLLISION RESOLUTION TECHNIQUES

Consider a hashing function that stores data "randomly" in a fixed size hash table with no auxiliary storage. If k out of the n locations have previously been filled randomly then the expected number of locations that must be looked at until an empty one is found is 1+k/(n-k+1).

Ullman [40] pointed out that there exist nonrandom hashing functions that are more efficient for certain values of k and n. He gives a new method for determination of probe sequences in which, instead of computing a hash address h(x), each key x is mapped into an entire permutation of {0,1.....,m-1 }, which is chosen as the probe sequence to use for x. Each of the m! permutations is assigned a probablity, and the generalized hash function is designed to select each permutation with that probability. He shows that the expected number C0(k,n) of trials to insert the (k+1)st item into a table of size n can be improved upon for certain k and n. However, C0(k,n) is a bound in the following sense: if there is any insertion algorithm that is superior to random insertion for some value of k, then it is

inferior for some smaller value of k. Finally he has shown that randomness is a sufficient but not necessary condition for attainment of the 1+k/(n-k+1) performace for all k.

Further investigation of the above method and comparison with the linear probing and double hashing of collision resolution has been performed by Guibas [41].

2.5 CHOICE OF A DIVISOR

The divisor is an important factor in the hash transformation. In an attempt to approximate a uniform transform several different divisors have been used. They are as follows:

a) Prime divisor

b) Odd, but not prime, divisor

c) Even divisor

The investigation performed in the present thesis showed that even divisors give very poor results. Since division of an even number by an even divisor results in an even remainder, and division of an odd number by an even divisor results in an odd remainder, a skew distribution of addresses exists after the transformation, and poor performance therefore results. In general, if a large number of keys are congruent modula d, and if D is a multiple of d, then the use of D as a divisor will result in poor performance. This is the basic argument for choosing the divisor, as suggested by Buchholz [26], as the largest prime

number close to, but less than, the size of the address space. But Lum [27] has pointed out that the choice of a divisor is not necessarily limited to prime numbers; selecting a number that does not contain any prime factor below 20, is probably sufficient to assure good performance. Furthermore, choice of the divisor as a prime number of the form 4n+3 (n is an integer) with use of some collision resolution technique gives a better result.

If the divisor is less than, but not sufficiently close to, the table size then some storage space is not accessible through a computed address. However, choice of a divisor greater than the table size, and hence use of a virtual table size, may be made provided the following two conditions are met.

a) There is no prime number close to, but less than, the table size.

b) The virtual size is close enough to the actual size.

In use of a divisor greater than the table size a calculated address that lies outside the address space is regarded as producing a collision and is replaced by some other address [19,21].

It may be noted that when the virtual size is significantly greater than the actual size the probablity of collision, even for the first term, is high. This probablity is given by P=(V-A)/V where P is the probablity of a collision for the first word, V is the virtual size of the table, and A is the actual size of the table.

In the present experiment the descriptors are ranked in decreasing order of frequency of occurrence and are stored in this order. By using a table of virtual size there may be a collision even for the first term. In the instance that A=32 and V=43, the probablity of a collision for the most frequently used word is P=(43-32)/43 which is more than 25 percent. According to the Zipf law [8], if a virtual storage technique is applied in our experiment then when the file is used the collisions will be even more frequent and the expected number of accesses required to get a record will be correspondingly increased.

The present investigation of the results of using different divisors has shown that the prime number close to, but less than, the table size gives the best overall performance.

2.6 TABLE SIZE

There arises the question as what kind of table size is most suitable for key to address transformations, and whether the table size should be limited to be a prime number or a power of two, and so forth.

In general the type of table and its size can be chosen with respect to transformation and collision resolution techniques. Furthermore, if collisions are resolved by storing the data in a prime area then the table size affects the collision resolution technique. Since to store the collided terms the hash algorithm must make a sequence of probes in the table then in the following

investigation the various table sizes are discussed in terms of the transformation and collision resolution aspects.

2.6.1 TABLE SIZE M=POWER OF TWO

Most of the hash-addressing schemes produce a number of random bits to be used as an index, and it is most convenient to use only values of m that are a power of 2. For instance, if the hash transformation generates L bits one can use the L bits as an index in a table that has space for m=2**L entries. Such a table size may be preferable for an efficient use of storage. In addition, generation of the hash addresses generally requires simple calculations, and the operations modulo m that appear in the increment law may be performed very simply by masking a proper number of bits. For such a table size collisions may be handled by any of the four techniques described in section 2.4.

2.6.2 TABLE SIZE M = PRIME NUMBER

For the the prime number table size a division hash coding may be used for the transformation. It is effective but has certain disadvantages. It is convenient to set up a hash table whose length is exactly equal to one or more pages of memory and hence is a power of 2. Of course, any prime number that is sufficiently close to a power of two could be used in conjunction with the division method. However, Lum [27] shows that in the division method the table size is not necessarily limited to be a

prime number; selecting a number that does not contain any prime factor below, say, 20 is probably sufficient to assure good performance. Also h[illegible]laims that the division method is a relatively simple technique and, in general, does provide a good overall performance. For such a table size collisions can be handled effectively by either linear or quadratic techniques.

2.6.3 TABLE SIZE M = A POWER OF 10

For a table size of m = 10**L the address can be obtained by extracting L digits from the key and concatenating them in some order. The range is {0....10**L }. Any hash transformation method available for a table whose size is a power of two could be used for a table size of 10**L. For resolving collisions a linear search or Ullman method [40] may be used, and if L is a prime number then a quadratic search may be applied to give a full table search (Ackerman and Batagetj, [39]).

The three different types of hash tables as explained above could be used effectively in key transformations. However using a table which does not correspond with any of the above table sizes is possible, but as explained in section 2.3.2, it may give a poor result. The division method is also available for any table size but, as indicated in section 2.5, it gives poor results with some table sizes. A linear search may be used for collisions with any table size.

2.7 SCATTER TABLE

It is possible to extend the concept of treating every colliding item as an overflow item by placing all entries in a separate area in which space is allocated for entries only as needed. Such a method of addressing requires the use of two files. One file is the data file in which data records are stored and in which synonyms are resolved. The second file is a scatter table into which the hashing function maps. Each active entry in the scatter table is a pointer into the data file and is set after the identified record has been stored in some convenient manner. However for a volatile set of data the maintenance of the data file may require an impractical amount of updating in the scatter table.

A very valuable characteristic of the scatter table technique is that the physical location of a data record is independent of its hashed address, and therefore of its identifier. It turns out that this property simplifies both the problem of storing variable length data records and the problem of retrieving data records on the basis of a number of different identifiers. A data item often occupies many words, whereas a pointer usually occupies only a part of a single word, so that the unused portion of the scatter table consists of single word items rather than the multiword spaces suitable for storage of data items. Because of this, much of the efficiency of the chaining method may be retained without having to move entries around in storage.

The use of a scatter table has two apparent disadvantages:

1- The additional space that is required for the table.

2- The additional access to the table that is required for every retrieval.

The above are not necessarily serious practical disadvantages. However if records are stored without a scatter table, and with a load factor of less than 1, then the space penalty for an unactive hash entry is high, whereas in use of a scatter table the only penalty is the storage used for the pointer.

CHAPTER 3

DICTIONARY STRUCTURE FOR TWO DIMENSIONAL HASH ADDRESSING

3.1 SPECIFICATION

Consider a two dimensional hash addressing scheme that operates on a set of descriptors. Let h1(term) and h2(term) be hashing functions that operate on the first c1 and last c2 characters respectively of any given descripter, where c1 and c2 are defined in term of descriptor length as follows:

Descriptor length (characters)	c1 and c2
1	1
2	1
3	2
4	3
>4	4

The terms are stored in a dictionary that is accessed through a hash table and index table as shown in fig. 3.1. The dictionary DICT1 consist of a set of 32 term strings, each of which contains 32 terms of the same length. An index table IND1

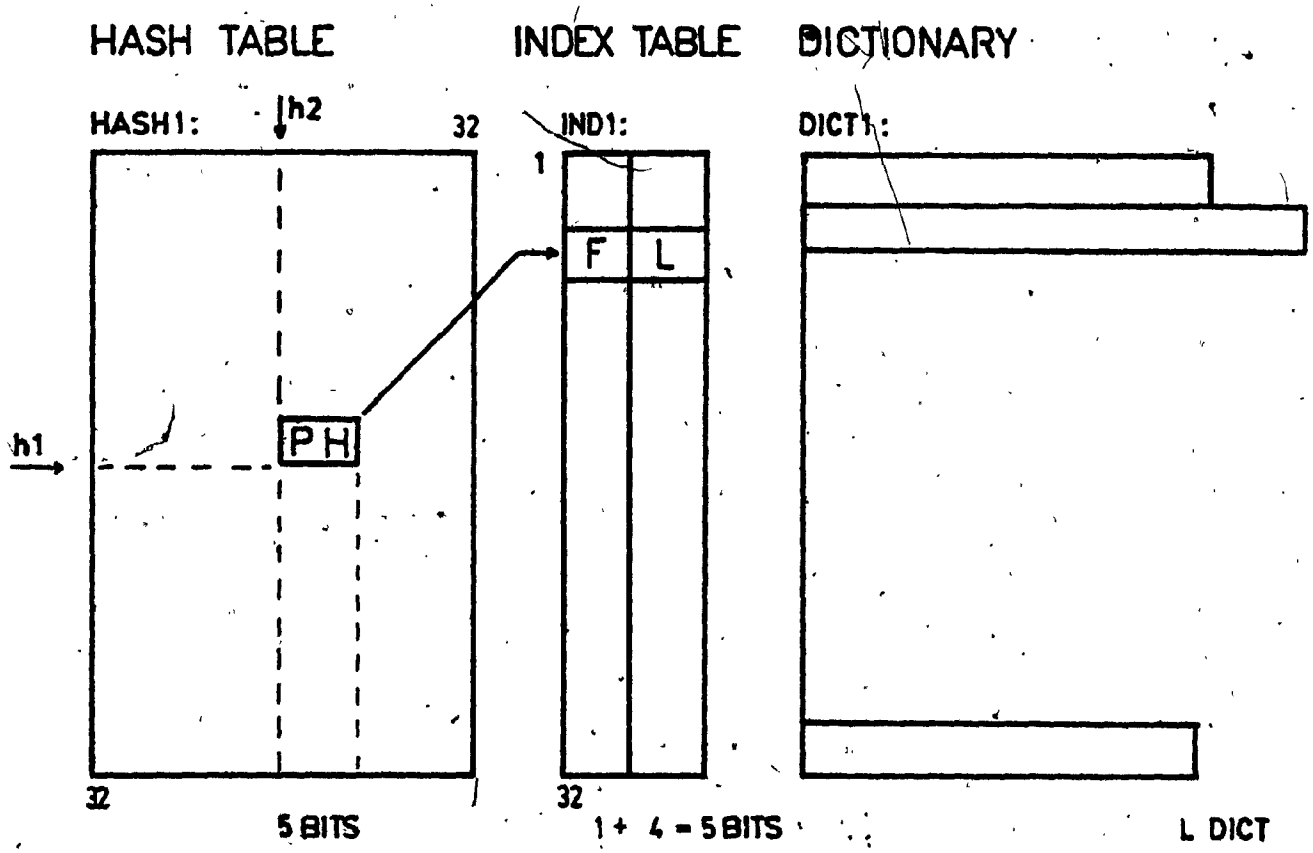

FIG. 3.1 : DICTIONARY STRUCTURE AND ACCESS SCHEME FOR DICT1.

contains 32 blocks of 5-bits each. A 4-bit length L indicates the length, in characters, of the terms in the corresponding term string. There is also a 1 bit flag f that is set to 0 while the corresponding term string is being created, and is set to 1 when the term string contains 32 terms.

The hash table contains a 32x32 array of 5 bit entries, each of which contains a pointer to an element of the index table.

For each term stored in DICT1 the pointer in the h1(term)-th row and h2(term)-th column of the hash table is set to point to the element of the index table that corresponds to the appropriate term string. The term is stored at the h2(term)-th position within the term string. For each pair of values h1, h2 any terms that cannot be stored in dictionary DICT1 because of collisions are stored in a dictionary DICT2 which may be located on disk, but DICT1 is stored in core.

3.2 EXPERIMENTAL ENVIRONMENT

The experiment is aimed at minimizing both the number of accesses required to retrieve a record and the number of collisions created by the hash transformation. There are many factors that affect key to address transformation techniques. In this section some parameters are considered and discussed in the light of the present experiment. The topics presented include:

1-Key to digits conversion.

2-Different hashing functions.

3-Choice of divisor.

4-Different data.

5-Treatment of collisions in the hash table.

6-Treatment of collisions in the dictionary DICT1.

7-Differently dimensioned hash table of the same size.

Two data sets are used in this experiment. The first data set consists of the most frequent terms of general English text as given by Kucera and Francis [42], who examined a total of one million words in which there occurred 50,406 different terms. The text sample covered a wide range of subject categories, and hence may be regarded as representative of terms in general text. In the first data set all words with more than 9 characters, and words that contain any symbol except '0' to '9' and 'a' to 'z', are excluded from consideration. The second data set is selected from a chemical titles tape, words with more than 10 characters are truncated to 10 characters, and duplications are eliminated.

The experiment has been performed on the most frequent 2559 words of the resulting lists organized in such a way that frequently referenced items are loaded first and infrequently referenced ones are loaded last. The loading is thus in decreasing order of frequency. Therefore the most frequently referenced items will tend to be placed in the home buckets of DICT1. By the time that collisions occur they will tend to be with the less frequent terms. Thus at retrieval time the collisions will also be less frequent. For the present experiment the computer program is written in the Fortran

language using the CDC Cyber 172/2 computer system.

3.3 COLLISIONS IN HASH TABLE

In the experiment there may be two different types of collisions, namely collisions in the hash table and collisions in the dictionary. In this section collisions are resolved in a manner that attempts to minimize the expected retrieval time.

There are two alternative ways to store overflows. They may be stored in a separate overflow area or in the prime area. In this experiment collisions in the hash table are resolved by one or more combinations of the following schemes:

S1) The term is stored in DICT2 (as discussed in chapter 5).
S2) A parallel hashing function is used, and the term is stored at a secondary address location in DICT1.
S3) A new scheme is considered in which more than one item may be hashed into each entry of the hash table.
S4) Multiple probes in the hash table are made.

The last three methods resolve collisions by storing entries in the prime area, and by using a limited search instead of a full table search. Although a full table search allows the most frequent terms to be stored in first level store it has two disadvantages. One of the most serious disadvantages arises from the fact that one of the primary objectives is to design a system which allows search to be specified with right truncation of

terms. It will be seen that if a full table search is used then when a question term is specified in truncation form the entire dictionary must be searched regardless of the fact of the existence of the hash table. Thus it does not take advantage of the hashing scheme. A further disadvantage is that the full table search also increases the average search time for question terms specified in untruncated form.

Because of the above considerations a limited table search is preferable. However if a limited search fails then the term will be stored in a dictionary DICT2 in second level store [43].

The following three alternative schemes are considered for resolution of collisions in the hash table by production of a secondary address.

SCHEME S2.1:

In this scheme consideration is given to the use of two different methods of key-to-digits conversion in order to introduce a parallel hashing function. The address of the term is first calculated by using the first method of key-to-digits conversion, in the case of a collision in the hash table a secondary address is determined by using the second method of key-to-digits conversion.

It may be noted that the two methods of key-to-digits conversion are likely to produce different addresses. Therefore

if the first address location is occupied and the second is free the term can be stored in DICT1 instead of in DICT2.

SCHEME S2.2:

In this technique in order to introduce a parallel hashing function two different divisors are used. The first address location is determined by using the first divisor, and in the case of a collision the second divisor is used to calculate the home address. In the first experiment the divisors 43 and 31 were used.

SCHEME S2.3:

One of the simplest ways of solving collisions in two dimensional hash addressing is to exchange the row number with the column number and to store the term in the new address location. This method is only applicable when using a square hash table. Thus if position (h1,h2) in the hash table is occupied then position (h2,h1) can be used (if it is available).

The following two conditions must be satisfied in order to make this scheme attractive.

1- It produces different values for row and column numbers, so that by exchanging row and column number a different address location is produced.

2- Exchanging row and column number produces an unoccupied

address location.

SCHEME S2.4

In this scheme in order to produce a parallel hashing function two different hashing functions are used. The primary address location is calculated by using the first hashing function, and in the case of a collision in the hash table the secondary address location is determined by using the second hashing function [43]. In the first experiment the two hashing functions are the division and midsquare functions. This scheme was found to produce the best overall performance in comparison with the other three schemes (S2.1, S2.2 and S2.3).

The success of any of these four schemes depends on the two following conditions:

1- Secondary address is different from previous address location.

2- Secondary address is unoccupied.

It may be noted that if the row number is equal to the column number the scheme S2.3 always produces the same address location. However, in use of the three other schemes there is still a non-zero probablity of repeating the previous values of the column and row numbers.

The following four alternative schemes are considered for resolution of collisions in the hash table by allowing to hash more than one term into an entry in the hash table.

SCHEME S3.1

As mentioned earlier each occupied entry in the hash table points to a term string in the dictionary DICT1.

Suppose a term with length L hashes to the entry (h1,h2) in the hash table and is stored at the position h2 in term string t. At subsequent time if a collision occurs at entry (h1,h2), the length of the collided term is compared with L. If a match occurs then the term can be stored at any unoccupied position in term string t. If all locations in term string t are occupied then another method must be used for resolution of collisions. In use of this scheme each entry in the hash table also contains a value equal to the number of terms hashed through that entry and stored in the corresponding bucket. The value is greater or equal to zero.

In general, the available bucket space for each entry is dependent on the following factors.

1- Number of collisions in that entry.

2- Probablity that the stored term and the collided term have equal length. This probablity is equal 1/L, where L is the number of different possible word lengths.

3- Probablity of finding an unoccupied space in the corresponding term string.

SCHEME S3.2:

This scheme is similar to scheme S3.1, and it can be used in conjunction with scheme S2.3. In this scheme if a collision occurs at the entry (h1,h2), and if a match occurs in the length of the terms, then the term is stored only at an unoccupied h1-th position in term string t. Thus for each entry in the hash table a maximum of two terms can be hashed and stored in a term string at locations h2 and h1 respectively.

SCHEME S3.3:

In scheme S3.1 each entry in the hash table contains a variable bucket size that ranges from m to n. Since there may be entries through which no terms are hashed then m=0. The value of n is the maximum number of terms that may be hashed through an entry and stored in a term string in the dictionary DICT1.

In scheme S3.1, when a term is hashed through the entry (h1,h2) in the hash table, the corresponding term string t in dictionary DICT1 is examined for an unoccupied location; if term string t is full the collided term must be stored in a dictionary DICT2 in a second level store. However the present scheme extends the scheme S3.1 in that if term string t is full then all terms hashed through entry (h1,h2) are removed from term string t and stored in another term string that also has space available for the collided term. However, in order to remove the above

mentioned terms from term string t, the exact locations of these terms must be recorded previously in an extended storage location of entry (h1,h2). Thus at transformation time each entry in the hash table will include a maximum of n locations. When a term is hashed through an entry in the hash table and stored in a term string, the stored position of the term in that term string is stored in an extended location for that entry. Thus each entry in the hash table contains a record of the positions of all terms hashed through that entry and stored in a unique term string in the dictionary DICT1.

Suppose n terms are hashed through the entry (h1,h2) in the hash table and are stored in the term string t at positions p1,....,pn in the dictionary DICT1. If another term is hashed to the entry (h1,h2), but the term string t is full, then the following algorithm can be used to store the collided term in DICT1.

1- Examine all term strings of appropriate length in DICT1 until either n+1 free locations are found or there are no further term strings.

2- If n+1 free locations are found then go to step 4;

3- If there is space available then create a new term string; otherwise go to step 5;

4- Remove the n terms from term string t and store these terms, and the new collided term, in the new term string and set the pointer PH in the new term string; then exit.

5- The search fails; another collision resolution technique must

be used.

SCHEME S3.4:

A further improvement can be gained from the scheme S3.2 by using the same procedure as in the scheme S3.3.

In fact this scheme can obtained from the scheme S3.3 by replacement of step 4 by the following step.

4- Remove the previous term from term string t, and store this term and the new collided term at locations h2 and h1 respectively in the new term string, and reset the pointer PH to point to the new term string, and exit. It may be noted that this scheme is applicable when the value of n is equal 1.

The following two alternative schemes are considered for resolution of collisions in the hash table by allowing the hash algorithm to make a sequence of probes into the hash table.

SCHEME S4.1:

In general this scheme makes a sequence of probes into a row of the hash table. The sequence terminates when a probe selects an empty location or the row is found to be full. There are many algorithms for producing a sequence of probes, and they are discussed in section 2.4. However in this experiment the Day's technique is used to cover a whole row in the hash table. When

the size of the row is small a linear search is preferable since it needs less time per probe.

As mentioned in chapter 4, a search for a right truncated word requires that all entries in a row be examined. Thus, there is a certain advantage in hashing the collided items into another location in the same row. This algorithm facilitates the operation of the retrieval algorithm when searching for a right truncated term. If the size of the row is sufficiently large then this algorithm is more likely to resolve all collisions in the hash table by storing them in the dictionary DICT1.

This algorithm produces very good results when used in conjunction with algorithm S3.1 or S3.2. Combination of this algorithm with algorithm S3.2 gives the best overall performance.

SCHEME S4.2:

A further improvement over scheme S4.1 could be obtained if the length of the row in the hash table is chosen to be equal to the number of different term lengths in the data base. Then in the hash table h2 may be determined by examination of the length of the word. Also each row in the hash table may contain entries for all possible word lengths. If this scheme is used with scheme S3.2, then it completely eliminates collisions in the hash table.

In conclusion of the discussion of this section it may be noted that the simplest method for resolving collisions is to store all collided terms in a separate overflow area (DICT2), without application of further search technique. In general, however, it is better to apply one or two search techniques as explained above, and if these searches fail then to store the term in DICT2.

3.4 COLLISIONS IN THE DICTIONARY DICT1

Consideration has been given to two different forms of hash table storage and the retrieval algorithm. They differ in the manner in which collisions are resolved. One of the two approaches is to resolve collisions by somehow finding an unoccupied space in DICT1 for those items whose natural home location is already full. The space must, of course, be found in such a way that the item can be retrieved later.

The other approach is to store the term in the dictionary DICT2 whose access requires a disk access.

For storage of the term in DICT1, and solution of the collision problem, the following six alternative algorithms are considered.

ALGORITHM 1:

1. Examine the h2-th location in each term string of appropriate

length in DICT1 until either a free location is found or there are no further term strings.

2. If a free h2-th location is found then use it for storage of the term and go to step 4.

3. If there is space available then create a new term string, and store the term in its h2-th location.
Otherwise go to step 5.

4. Set the pointer PH to indicate the appropriate term string in HASH1 and exit.

5. Store the term in DIGT2.

ALGORITHM 2:

This algorithm is obtained from algorithm 1 by placing the following steps between steps 2 and 3:

2.1 Examine the h1-th location in each term string of appropriate length in DICT1 until either a free location is found or there are no further term strings.

2.2 If a free h1-th location is found then use it for storage of the term and go to step 4.

ALGORITHM 3:

This algorithm is obtained from algorithm 1 by replacement of step 5 by the following 3 steps:

5. Examine the h1-th location in each term string of appropriate length in DICT1 until either a free location is found or there

are no further term strings.

6. If a free location is found then use it for storage of the term, set the pointer PH to indicate the appropriate term string in HASH1, and exit.

7. Store the term in DICT2.

ALGORITHM 4:

This algorithm is obtained from algorithm 1 by replacement of step 5 by

5. Examine all term strings of appropriate length in DICT1 until either a free location is found (at any position) or there are no further term strings.

6. If a free location is found then use it for storage of the term, set the pointer PH to indicate the appropriate term string, and exit.

7. Store the term in DICT2.

It is clear that use of this technique decreases the number of terms that are stored in DICT2 because of collisions in DICT1. On the other hand each time that a collision in DICT1 is resolved by finding an unoccupied space, other than at the h2th position, one location in the hash table will be occupied; this tends to increase the probablity of collision for subsequent terms. However in use of this algorithm the total number of the unresolved collided terms in DICT1 is less than for either of the first 3 algorithms.

ALGORITHM 5:

This algorithm is obtained from algorithm 1 by placing the following steps between step 2 and 3

2.1 Examine all term strings of appropriate length in DICT1 until either a free location is found at any position or there are no further term strings.

2.2 If a free location is found then use it for storage of the term, set the pointer PH to indicate the appropriate term string in HASH1, and exit.

According to this algorithm the collisions in dictionary DICT1 are resolved by finding, if possible, an unoccupied space in the same dictionary. On the other hand by minimizing the number of unresolved collisions in DICT1 it increases the probability of subsequent collisions in the hash table. But the total number of frequent terms required to be stored in DICT2 should be appreciably less than for the other four algorithms. However in use of this technique, since h2th positions will often be unavailable for subsequent terms, then retrieval of terms will often require additional searches in the corresponding term strings.

In the algorithm 4 and 5 collisions are solved by storing terms in another position according to the following algorithm which is obtained from Day's algorithm (section 2.4) with modification in steps c and e. In this algorithm P is the table size and j is the hash address.

a) i=-p

b) hash to value j

c) store at location j+1 unless location filled; if full set i=i+2.

d) j=remainder((j+abs(i))/p).

e) if i<p go to c; if i=p store at location p+1; if it is occupied then the table is full.

This algorithm is designed for application to a table size of 4n+3 (n is an integer). A table (row) size of 31 or 7 is assumed in application of the algorithm, but the real table sizes are 32 and 8 respectively. For a table of size 32, the first 31 locations are filled first according to the above algorithm and the 32nd location is filled last.

ALGORITHM 6:

When a block structure is used for storage of variable length data an obvious improvement of the above storage schemes can be obtained by allocating a block of suitable size for storage of all words of a given length. In order to allocate the storage efficiently the behavior of the particular transformation method must be examined for specific data, and the number of terms that successfully hash through the hash table must be counted for each word length so that space may be allocated for storage of terms of each length.

In contrast, although algorithm 5 allows efficient use of storage, it may happen that only a few terms of a particular length occupy a particular term string while there are many collisions in term strings for other term lengths. However the present algorithm is more conservative in its creation of new term strings and therefore helps to reduce the problem. The algorithm is as follows:

1- Run the program according to algorithm 5, count the number of terms for each word length, the number of empty locations in each term string, and the number of unresolved collisions caused by insufficient space in DICT1.

2- Based on the above statistics an appropriate number of term strings are allocated for each term length. This step may be done manually rather than automatically.

3- Run the program according to the algorithm 5 with term strings allocated for each term length. The exact algorithm for this step is obtained from algorithm 5 by replacement of step 3 by the following step.

If there is space available for the current term length then create a new term string and store the term in its h2-th location; otherwise go to step 5.

This algorithm gives the best overall performance, since in terms of collision resolution it is slightly better than the algorithm 5, and in terms of the expected retrieval time it is better than algorithm 4. However it requires the program to be run twice, but this is not a significant disadvantage since in

data base applications the time required to perform a transformation is less important than the time required for retrieval. This is because each transformation is performed only once whereas retrieval is performed many times.

3.5 DIFFERENTLY DIMENSIONED HASH TABLES OF THE SAME SIZE

In the first experiment a two dimensional hashing function with an index table of 32 entries, and term strings of 32 terms, is used. Therefore a maximum of 1024 terms may be stored in the dictionary. But 1024 storage locations may also be obtained by use of an index table of 128 entries and term strings of 8 terms. It is of interest to determine the relative advantages of these two differently dimensioned tables of the same size.

For these two tables the midsquare method of hashing can be used effectively since numbers 32, 128 and 8 are powers of two. If the division method is used the best choice of a divisor for use with each table should first be determined. As mentioned earlier, the best divisor is a prime number close to, and less than, the table size. But somtimes there is no such prime number for the specified table size. By using a two dimensional hashing function the size of the index table and term string may be adjusted according to the value of the divisor.

If the divisor is less than, but not close enough to, the table size then some locations in the hash table are unused. In the first experiment a divisor of 31 was used for a table size of

32, and hence 63 entries in the hash table were unused. This has the effect of increasing the number of collisions in the hash table. However in order to allow comparison of the two transformation methods, namely division and midsquare, and also the choice of different divisors then fixed dictionary sizes are used. In algorithms 4, 5 and 6 of section 3.4 all 1024 locations in the dictionary are used, but in algorithm 1, 2, and 3 the size of the dictionary DICT1 is chosen to be the same as that of the hash tables.

Suitable divisors that may be chosen for use with an index table of 128 entries and term strings of 8 terms are the divisors 127 and 7. In contrasting the 32x32 with the 128x8 table size using the division method, it may be noted that with divisors 31, 127, and 7 then in the first case the number of entries in the hash table is (31x31) = 961, but in the second case the number of the entries is (127x7) = 889. Thus in order to allow comparison of these two different table sizes using the division method the divisors 139 and 7 are used, instead of 127 and 7, in order to have a table size of 139x7=973 which is sufficiently close to the previous table size of 31x31=961. It is found that, by increasing the size of the index table, the number of collisions in the dictionary is decreased because creation of any new term string may prevent the creation of further more useful term strings of different lengths. Sometimes it happens that only a few terms of a certain length occupy a long term string (32) and yet there are many collisions for other terms of different

lengths. Short term strings reduce the amount of space allocated to almost empty term strings, allow the space allocation to be more efficient, and hence decrease the number of collisions in the dictionary. Thus in the first case (31x31) there will be fewer collisions in the hash table, but there are more collisions in the dictionary. Overall comparison of use of the two tables is presented in chapter 4.

3.6 NEW METHOD FOR ELIMINATING COLLISIONS

In comparison to use of a binary, or similar search [4,5], the use of a scatter storage table greatly reduces the number of probes required to find an item in the table or to enter an item into the table, but the occurrence of collisions destroys some of the simplicity of hashing. In this section consideration is given to a method that attempts to eliminate some of the problems caused by collisions.

Let I be a given set of identifiers. If h transforms the identifiers in I into unique addresses then a single probe is sufficient. Such a transformation is called a perfect hashing function. Knuth [29] defines an amusing puzzle, namely to find a perfect hashing function for a given set I. He points out that a slight modification of I may change h completely, and that the tedious calculations to find h must all be repeated.

Sprugnoli [44] has considered a method that allows retrieval of an item in a static table by means of a single probe. In

practice the method is applicable to a small set I, and may be usefull in applications such as to assemblers. The method is applicable in the case where a particular file or key set is so stable that no change or insertion is likely to occur. Unfortunately, in information retrieval such stability is seldom found, and in practice very large key sets occur.

In the following a method is considered which is more flexible in that it allows changes or insertions, is applicable to large key sets, and yet requires a single probe for retrieval of an item. The method uses a scatter index table with variable bucket sizes. In the scatter table the data records are stored in a separate storage area in which space is allocated for entries only as needed. Suppose that in the scatter table by using a key transformation n terms are hashed into an entry e. Then in the data file n locations are allocated to the entry e.

If n terms are hashed through each entry in the scatter table then the transformation is called a uniform distribution transformation, and this facilitates the retrieval algorithm. However when n is equal to zero (no term hashed through the entry e), the entry e contains the initial value which is not equal to the value of any valid pointer, and no storage is allocated to the entry e in the data file.

If n=1 then entry e is set to point to the location in which the term is stored in the data file.

If n > 1 then all terms hashed through the entry e are stored in sequence in the data file, and the entry e is set to point to the starting addresses of the sequence in the data file. Thus the value of n for each entry should be known in advance so that storage may be allocated in advance for each entry in the data file. In such a scheme the number of entries in the hash table h is not necessarily equal to the number of entries in the data file d. The ratio of h/d indicates the average bucket size in the hash table. However small bucket sizes decrease the retrieval time at the cost of extra storage, and vice versa.

In fig. 3.2, h and d are equal to 10, and so the average bucket size is one. For the entries p7 and p9 in the scatter index table there are 2 terms hashed, thus in the data file 2 locations are allocated for each of these entries. For entries p2 and p8 no terms are hashed in the scatter index table, and so no storage is allocated for entries in the data file. All other entries contain one location in the data file. In fig. 3.2, storage is allocated for each entry in such a manner that consecutive nonempty entries in the index table have consecutive locations in the data file. The data can be stored in any other order, provided that all terms hashed through a single entry in the scatter index table are stored in sequence in the data file. However storing the terms according to fig. 3.2 has certain advantages since subtraction of two consecutive nonempty entries in the scatter index table indicates the number of words hashed to the first entry. An alternative method is to store the terms

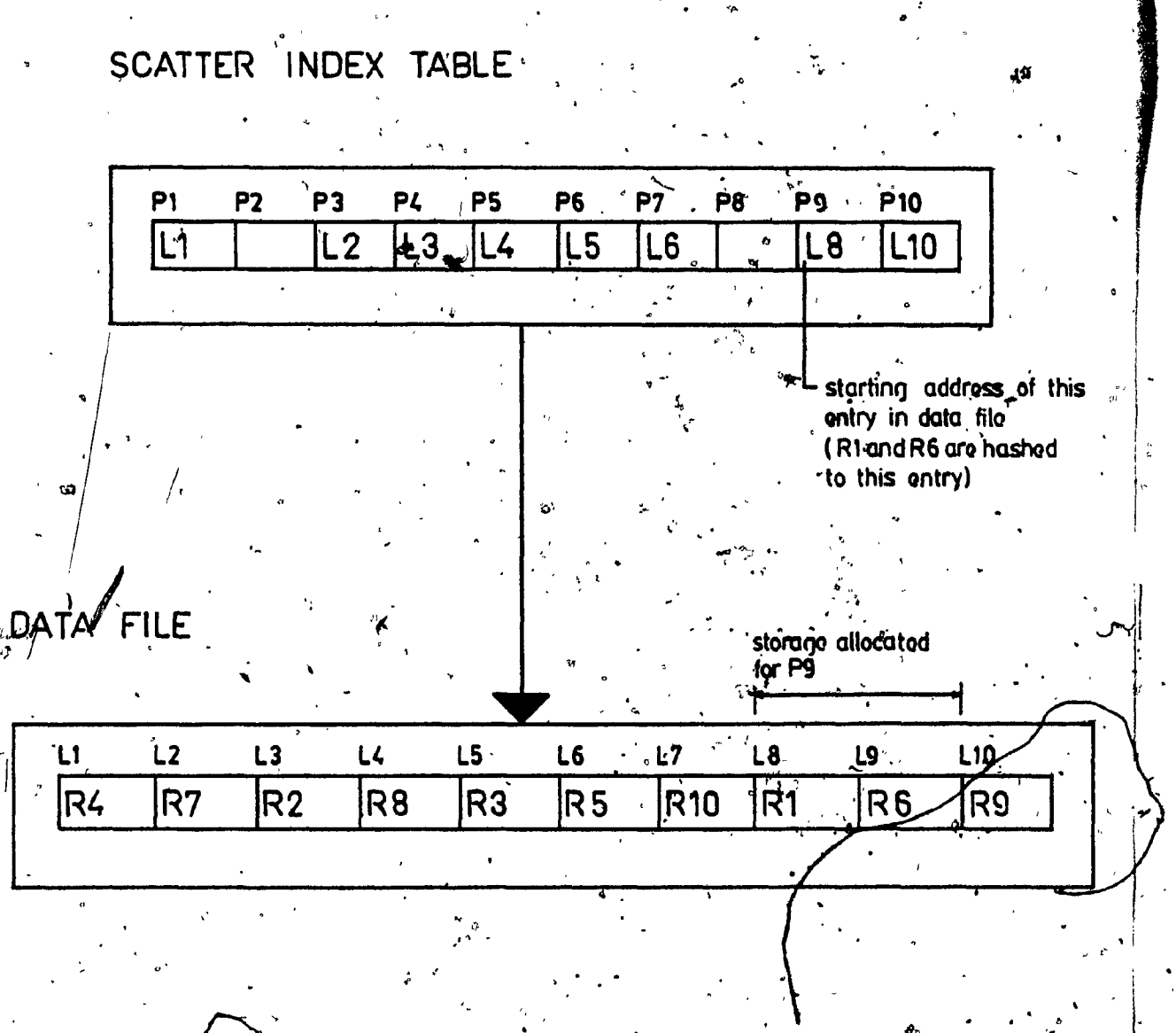

FIG.3.2: ADDRESSING VIA A SCATTER TABLE

in the data file in some other order. Then at retrieval time by knowing c, the maximum number of terms hashed through an entry, a term can be searched for by scanning the starting address of the entry and proceeding through c locations. By using two dimensional addressing, and storing all terms with the same length in a term string, the scheme may be modified in the following manner.

Consider a two dimensional index table of h rows and L columns INDT1, that is addressed by a hashing function h1(k1) and by term length t(k), where k1 is a key derived from the original word, and t(k) indicates the length of the original word. The most frequent terms are stored in a data file DF1, that is accessed through the index table as shown in fig. 3.3.

Let L be the number of different word lengths in the data base. Then data file DF1 consist of L term strings, each of which contains terms of the same length. These term strings are located in a one dimensional array in consecutive storage locations in order of increasing word length.

The index table INDT1 constitutes a hxL array of p-bit entries, each of which may contain a pointer to an element of the data file DF1. For each term stored in DF1 the pointer in the h1(k1)-th row and t(k)-th column of INDT1 is set to point to an element of the appropriate term string. All terms passed through the same entry in INDT1 are stored in the appropriate term string in consecutive locations in DF1, and the pointer in INDT1 is set

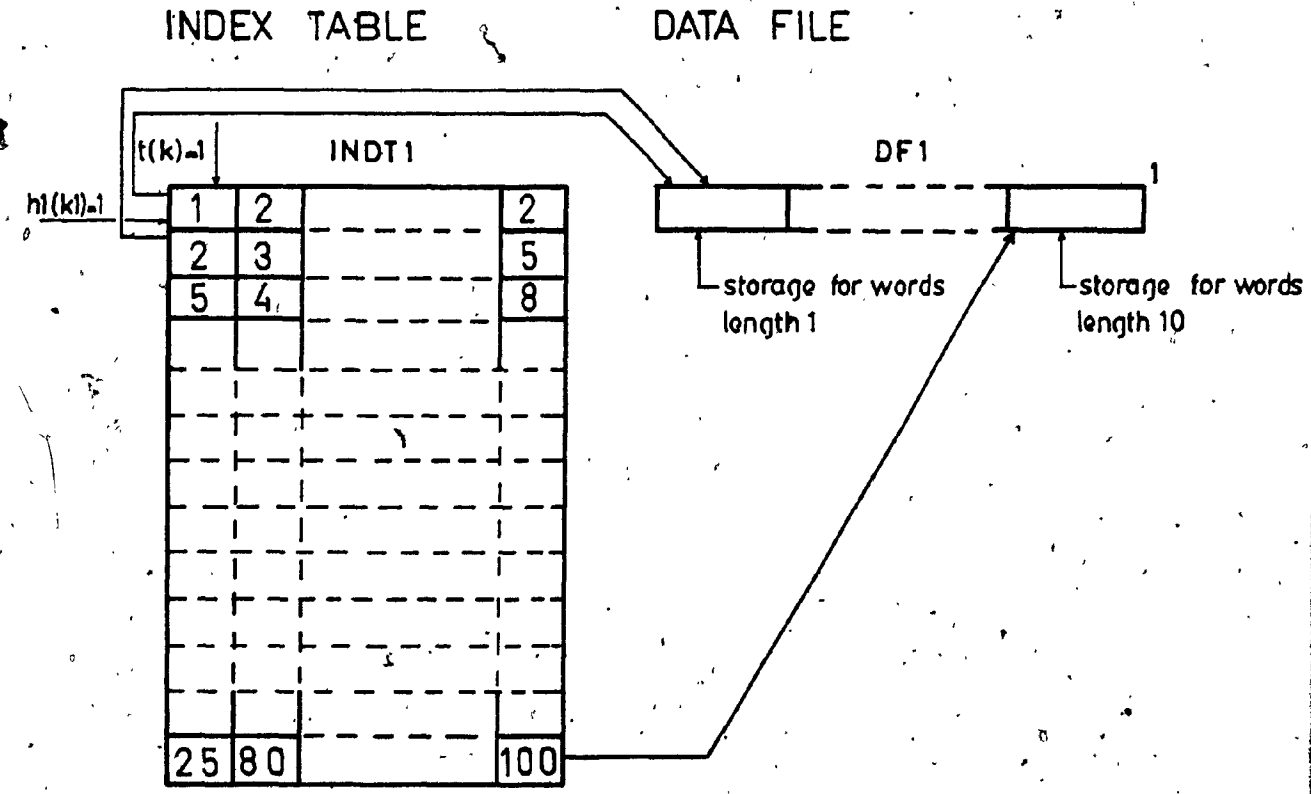

FIG. 3.3 : ADDRESSING THE DATA FILE VIA A TWO DIMENSIONAL INDEX TABLE

to point to the last term passed through that entry. Thus the first term string in DF1 contains all terms of length 1 that are hashed into pointer (1,1) of INDT1, then all terms of length 1 that are hashed into position (2,1) of INDT1, and etc.

The index table INDT1 can be of any size, but before choosing values of h and L it is advantageous to perform some analysis to determine the effect of the choice of the parameters h and L. If the values of h and L are chosen carefully it is possible to improve performance and save storage. In fact by increasing L by 1 the size of INDT1 is increased by h storage locations. In general the value of L is limited to the largest word length among the first N most frequent words that are stored in the first level store. It may happen that in the first hxL most frequent words, there are only a few words of some higher length. It may then prove convenient to choose the first most frequent N terms not greater than a certain length, and to store all words of greater length in a DF2 which may be accessed by a disk access. This leads to a saving of storage in the index table and facilitates execution of the retrieval algorithm.

Once the L is fixed then the remaining decisions are easier. One should decide on the approximate number (hxL) of terms to be stored in core. In a particular case the value of L may be chosen to be 10, and the approximate value of hxL chosen to be 1024, in which case the value of h will be 102.4. Then one should decide what method to use for the transformation. If the division method is used then it is preferable to choose a prime

number close to 102.4. In this example the prime number 103 may be chosen and the exact table size will then be 1030.

The number of bits required to hold the pointer value in the index table INDT1 depends on the vocabulary of the data base. However in the above case the average number of terms in each term string is 103. But there are likely to be some term strings whose numbers of terms are very different from the average, and this leads to a non uniform distribution of terms over the entire address space, and also increases the number of bits required to hold a pointer value. The number of bits should be chosen sufficiently large to address the maximum number of terms of any length. Examination of the most frequently used words in both general English text and the chemical literature indicates that 8 bits will be sufficient for a dictionary of the 1030 most frequent words.

The following algorithm stores the first hxL most frequent words in DF1. The algorithm consists of two passes. In the first pass the storage is allocated for each term string and also for each entry in the appropriate term string. In the second pass the terms are stored in predetermined locations in the DF1. In the following algorithm the counter denotes a temporary storage area that is used only at transformation time, and which has the same size as INDT1.

PASS 1

1- pass the first hxL words of length not greater than L through the index table INDT1 and set the counter[h1(k1),t(k)] = number

of terms hashed into the entry at position (h1(k1),t(k)).

2- From the above result allocate storage for each term string, and for each entry in the INDT1 allocate storage in the appropriate term string so that storage for INDT1[h1(k1)+1,t(k)] preceeds INDT1[h1(k1),t(k)]; set INDT1[h1(k1),t(k)] = starting address of the allocated storage for each entry in the hash table-1.

PASS 2

3- Rewind the data; from the first word to the last word repeat the following step.

4- Pass the term through the INDT1 by calculating h1(k1) and t(k) and set entry INDT1[h1(k1),t(k)] = INDT1[h1(k1),t(k)]+1; store term in location INDT1[h1(k1),t(k)] at the appropriate term string in the DF1.

The above scheme is designed to minimize storage and access time at the expense of updating [32].

A trade-off between space and time in hash coding with allowable errors has been analyzed by Bloom [45].

3.7 RESULT OF EXPERIMENT

In this experiment a number of factors (as already explained) were considered. The results are presented in tabular form in appendix I.

In order to compare the effect of the various factors a standard of measurement must be established. After an investigation of various approaches, the following measurements were used and examined with respect to their dependence on the number of different terms (NTERM).

1- Number of the terms stored in DICT1 (NDICT).

2- Number of unresolved collisions in the hash table. The corresponding terms are stored in DICT2 (NCH).

3- Total number of collisions in DICT1. The corresponding terms are stored either in DICT2 or in an unoccupied space other than at the h2th positon in DICT1 (TCD).

4- Number of unresolved collisions in DICT1. The corresponding terms are stored in DICT2 (NCD).

5- The two data sets used in this experiment consist of terms ranked in decreasing order of frequency.

Suppose D different terms are stored in DICT1 and DICT2. Then the probablity of occurrence pr of a word of rank r should approximately satisfy the equation [8]

$$pr = a / r$$

where $a = 1 / (1+(1/2)+(1/3).....+(1/d))$

which may be approximated by writing

$a = 1 / (\ln(d) + e)$

where e denotes Euler's constant of value 0.5772.

If it is assumed that the d most frequent words are stored in DICT1 then the proportion of the data base text represented in

DICT1 is given by

P1 = (ln(d) + e) / (ln(D) + e)

Thus

P1 = (ln(d) + 0.5772) / (ln(D) + 0.5772)

However in practice the value of P1 is found to be smaller than the value calculated above since the formula does not take into account the probablity of occurrence of unresolved collisions in the hash table. Terms that correspond to unresolved collisions are stored in DICT2.

6- Proportion of data base text represented in DICT2 is given by

P2 = 1 - P1.

7- Ratio of the two probablities (P1/P2).

It may be noted that NTERM=NDICT+NCH+NCD.

Comparsion of the three methods of key to digits conversion shows that method B gives a very poor result when used with an even divisor. In division by an even number the last character in any given term is the most significant in determining the address location because the remaining characters in the term, after conversion to digits, produce a number that is a multiple of 64 and hence divisible by several other even numbers (e.g.32). No significant variation in performance was discovered in method A and C when used with different divisors. However in this experiment division by 31 gives a better result in both methods.

Comparison of four schemes used for solving collisions in the hash table shows that the scheme S4 with incorporation of scheme

S3 gives the best result. As presented in chapter 4, scheme S4 facilitates execution of the retrieval algorithm during the search for truncated terms.

The result of the investigation of the six algorithms used for solving collisions in DICT1 shows that if the location of each term is specified to be only at the h2th position in each term string (algorithm 1) then a number of frequently used terms must be stored in DICT2 because of collisions. However, if position h2 is occupied the term may be stored at the H1-th location in a corresponding term string (algorithm 2 or 3). Algorithms 4, 5 and 6 minimize the number of terms that must be stored in DICT2, but retrieval of the term might require a search through the term strings., Thus there is a trade-off between the time required to search the term string in DICT1 and the time required to access DICT2, which at retrieval time requires a disk access. As a result, the algorithm 6 gives relatively good results in terms of retrieval time.

Investigation of different methods of solving collisions led to a new scheme (4.6) which resolves all collisions by storing them in DICT1. This scheme is preferable, since data is organized in decreasing order of frequency and there is no need for any disk access for the first 1024 most frequent words.

In practice, a combination of factors must be considered. Tables I through LXXXI in appendix I present the effect of different factors on two dimensional hash addressing. By

resolving collisions in the hash table according to schemes S4.1 and S3.3, and solving collisions in DICT1 according to the algorithm 6, we minimize the number of the collisions for which terms are stored in DICT2. Use of method B for key-to-digits conversion with divisor 32 gives very poor performance. Furthermore, in any practical storage and retrieval system many complicated factors exist. The system analyst can exercise a trade-off between saving storage and saving time.

CHAPTER 4

DICTIONARY SEARCH FOR TERMS IN LEVEL 1

4.1 RETRIEVAL OF A TERM IN LEVEL 1

The ability to rapidly retrieve individually identified records from a set of possibly millions of records is an essential function of on-line retrieval systems. Since interactive systems may handle thousands of record retrieval requests each hour, the minimization of the time of a retrieval operation is a principal system design objective.

Consider a two-level memory hierarchy [22,23] that contains a set of entries. Suppose the time required to store or retrieve an entry from the first-level store is less than the time required for the second level.

To retrieve an entry the key is processed by the hashing algorithm, and the storage area in the first-level store is searched for the required entry according to the appropriate algorithm. Suppose the entry is not found. Then either the entry is in the second-level store or it is not in the memory hierarchy at all. Without the hashing scheme the entire first-level store would have to be searched before this could be determined.

In hashing it does not really matter how complicated the transformation algorithm is; simplicity of the retrieval

algorithm is more important since the transformation is performed only once whereas the retrieval algorithm is executed several times [32,46]. In effect when scheme S3 is used with scheme S1, S2 or S4 (as already explained in section 3.3) collisions are resolved in the dictionary DICT1 without introducing any additional expense in the retrieval algorithm. However, a complicated transformation algorithm may require a complicated retrieval algorithm, and vice versa. For instance, when the scheme S1 and the algorithm 1 are used for resolution of collisions in the hash table and the dictionary DICT1 respectively, the retrieval process may be simplified but it may not make effective use of the first-level store. The hashing algorithm may cause an entry to be placed in the second level store while space in other entry of the first-level store is still available. This is undesirable since it takes longer to retrieve an entry from the second-level store.

If the first 1024 most frequent terms are stored in the first level store, and the distribution of the length of the 1024 most frequent terms is uniform, then good performance will be achieved. However the two data sets used in this experiment are typical of data bases is general in that their word lengths are not distributed uniformly. It therefore proves convenient to select the 1024 descriptors as the 1024 most frequent terms of less than a given length. There are two reasons for this. Firstly, in use of a non uniformly distributed data set a few term strings may contain single terms although other term strings

are full. The result is inefficient storage. Second, limiting the stored terms to be not more than a given length facilitates the retrieval algorithm, since if a word is found to be longer than the longest term in DICT1 it need be searched only in the second level store.

The first data set is selected from the Kucera Francis table, and words of less than 10 characters are stored in the first storage level DICT1. The second data set is selected from the chemical titles tape, and words of less than or equal to 10 characters are stored in the first level.

4.1.1 RETRIEVAL ALGORITHM FOR VARIOUS COLLISION SCHEMES IN HASH1

Two quantities that affect the retrieval of a descripter are 1) its length, and 2) its content. If a descripter is not longer than the longest element in DICT1 then it is searched for in DICT1 according to the appropriate algorithm. Otherwise, or if the descripter is not found in DICT1, then it is searched for in the second level store.

In the following section retrieval of a term through the hash table is considered when any of the S1 to S4 schemes are used for resolution of collisions in the hash table.

R1 RETRIEVAL ALGORITHM FOR THE SCHEME S1

Step 1: If the length of the descripter is longer than the longest element in DICT1 go to 7.

Step 2: Calculate H1 and H2 according to the appropriate hashing scheme.

Step 3: If location (H1,H2) in the hash table is empty then go to step 7.

Step 4: Compare the (HASH1(H1,H2),2) location in the index table IND1 with the length of the searched term.

Step 5: If it is not a match go to step 7.

Step 6: Search the dictionary DICT1 according to the scheme used for the collision resolution in DICT1; if the term is found exit.

Step 7: Search the term in dictionary DICT2.

R2 RETRIEVAL ALGORITHM FOR THE SCHEME S2

This algorithm can be obtained from the retrieval algorithm R1 by the following three changes.

a) by initializing the value of I to 0 before step 1.

b) by placing the following step between steps 2 and 3

Step 2.1: If I=1 and the calculated addresses for H1 and H2 are the same as the previous one then go to step 7.

c) by replacement of step 7 by the following step:

Step 7: If I=0 then set I=I+1 and go to the step 1, otherwise search for the term in dictionary DICT2.

R3 RETRIEVAL ALGORITHM FOR THE SCHEME S3

Scheme S3 can be used in conjunction with any of the S1, S2 or S4 schemes. The retrieval algorithm for this scheme is then dependent on the other scheme with which it is used in conjunction. For example, if a combination of schemes S2 and S3 is used then the retrieval algorithm is exactly the same as the one given for scheme S2. This scheme resolves collisions in the hash table by hashing more than one element in each entry without complicating the retrieval algorithm.

This scheme can be used in conjunction with any collision resolution algorithm in DICT1; except with algorithm 1 which does not allow hashing of more than one element into each entry, since the term must be stored only at the H2-th position in the corresponding term string.

R4 RETRIEVAL ALGORITHM FOR THE SCHEME S4

This algorithm can be obtained form the retrieval algorithm R1 by intializing the value of I to -P before step 1, and replacement of step 7 by the following steps. P indicates the size of the row in the hash table, and J is the calculated hash address. H2 is incrimented by 1, because the starting address in each row is one.

Step 7: If I>=P go to step 5, otherwise do the following:

Step 7.1: Set J=(J+abs(I)) mod P.

Step 7.2: Set I=I+2.

Step 7.3: Set H2=J+1 and go to step 3.

Step 8: If H2 ≠ P+1 then set H2=P+1 and go to step 3, otherwise search for the term in dictionary DICT2.

The above algorithm is organized for rows of size 4n+3. If the size of the row is different from 4n+3 then the first 4n+3 positions (4n+3 < size of the row) are filled first, and other locations are filled last.

4.1.2 RETRIEVAL ALGORITHM FOR VARIOUS COLLISION METHODS IN DICT1

The collision resolution technique used in the dictionary DICT1 specifies step 3 of any one of the four retrieval algorithms described in section 4.1.1. In the following, retrieval algorithms are considered for each of the 6 algorithms used for resolution of collisions in DICT1. Any of these 6 retrieval algorithms can be replaced by step 3 of any one of the four algorithms explained in section 4.1.1, except that the algorithm R1 may not be used with scheme S3.

1- RETRIEVAL SCHEME FOR ALGORITHM 1

Step 3.1: If location (HASH1(H1,H2),H2) in DICT1 is empty then the descripter is not in the memory hierarchy.

3.2- Compare location (HASH1(H1,H2),H2) in the dictionary DICT1 with the searched term; if a match occurs exit.

2- RETRIEVAL SCHEME FOR ALGORITHMS 2 AND 3

Step 3.1: If location (HASH1(H1,H2),H2) in DICT1 is empty then the descripter is not in the memory hierarchy.

Step 3.2: Compare location (HASH1(H1,H2),H2) in the dictionary DICT1 with the searched term; if a match is found exit.

Step 3.3: If location (HASH1(H1,H2),H1) in DICT1 is empty then the descripter is not in the memory hierarchy.

Step 3.4: Compare location (HASH1(H1,H2),H1) in the dictionary DICT1 with the searched term; if a match is found exit.

3- RETRIEVAL SCHEME FOR ALGORITHMS 4,5 AND 6

Step 3.1: Set I to -P and set H to H2.

3.2- If location (HASH1(H1,H2),H) in DICT1 is empty then the descripter is not in the memory hierarchy.

Step 3.3: Compare location ((HASH1(H1,H2),H) in the dictionary DICT1 with the searched term; if a match is found exit.

Step 3.4: If I>=P go to step 7.

Step 3.5: Set J=(J+abs(I)) mod P.

Step 3.6: Set I=I+2.

Step 3.7: Set H=J+1 and go to step 1.

Step 3.8: If H2 ≠ P+1 then set H=P+1 and go to step 1.

4.2 RETRIEVAL OF TERMS SPECIFIED IN TRUNCATED FORM.

In a document retrieval system it may be desired to allow left and/or right truncation, and this may complicate the retrieval process. To facilitate use of query terms with truncation the principle of two dimensional hash storage scheme has been introduced. By storing an element in a table (32,32) at an address based on h1(k1) and h2(k2), where k1 and k2 are derived from the original word, there is a tendency to form clusters associated with prefixes or suffixes. For a question term that is specified in the form of at least four characters (as described in section 3.1) with right truncation the pointers from an entries row of HASH1 must be used to locate possible matching terms in DICT1.

For question terms specified in the form of at least four characters preceded by left truncation it is necessary to follow similarly the pointers from entire columns of HASH1. For example, words such as computer, computers, and computation, will be assigned to the same row, and a query that contains comp* will be answered by scanning the row selected by evaluating h1(comp). A similar process is performed to process queries containing "*tion", because most of the words terminating by tion will be stored in the column designated by h2(tion).

It may be noted that for a fixed size of storage an increase in the size of the index table decreases the retrieval times for right truncated words. For a hash table and dictionary of the

size nxm, and a query that contains comp*, all m entries in the h1(comp)-th row in the hash table must be scanned. If each entry in the h1(comp)-th row in the hash table points to a different term string then mxm terms in the dictionary must be scanned. Thus any reduction in the value of m leads to a decrease in the retrieval time for search for a right truncated word. But for queries that contain *tion all n entries in the h2-th column in the hash table must be scanned. If each entry in the h2(tion)-th column in the hash table points to a different term string then mxn terms in the dictionary must be scanned.

Comparison of hash tables and dictionaries of the sizes 128x8 and 32x32 from the viewpoint of searching for a term specified in right truncated form showed that in the first case a search for a term in right truncated form requires inspection of 8 entries (one row) in the hash table and a maximum of 8 term strings (8x8=64 locations) in the dictionary DICT1. To search for a term in left truncated form requires a scan of 128 entries (one column) in the hash table and a maximum of 128 term strings (128x8=1024 locations) in the dictionary DICT1.

In the second case to search for a term in right truncation form requires a scan of 32 entries (one row) in the hash table and a maximum of 32 term strings (32x32=1024 locations) in the dictionary DICT1. To search for a term in left truncated form requires inspection of 32 entries (one column) in the hash table and a maximum of 32 term strings (32x32=1024 locations) in the dictionary DICT1. It may be noted that a table size of 128x8 is

preferable since it requires less time to search for a term in right truncated form. To search for a term in left truncated form requires slightly more than when the table size is 32x32 since the number of entries to be scanned is 128 instead of 32. In either case the maximum number of locations that must be scanned in the dictionary DICT1 is 1024.

As explained in chapter 5, a second level storage may allow search for a term in right or left truncation form but not in both. For instance, if all words with the same suffix are stored in a dictionary DICT2, then the system allows search for a term in left truncation form only. However terms in right truncation form occur more often in queries than do left truncated terms. Thus in the second level storage the search is limited to allow only right truncation.

4.2.1 RETRIEVAL OF TRUNCATED WORDS FOR VARIOUS COLLISION SCHEMES IN HASH1

The following retrieval algorithm may be used to search for a term specified in truncated form:

Step 1: Set K=0 and I=-p.

Step 2: Calculate H1 and H2 according to the hashing scheme used.

Step 3: Set J=(J+abs(I)) mod P.

Step 4: Set I=I+2.

Step 5: Set H2=J+1.

Step 6: If K>0 then for I=1 to K repeat the following steps.

Step 7: If TEMP(K)=HASH1(H1,H2) go to step 9.

Step 8: K=K+1; TEMP(K)=HASH1(H1,H2).

Step 9: Search the HASH1(H1,H2)-th row in the dictionary DICT1 according to the collision resolution technique used.

Step 10: If I>=P go to 2.

Step 11: If H2 ≠ P+1 then set H2=P+1 and go to 3.

The above algorithm can be used to search for a term specified in truncated form when scheme S1 or S4 is used for resolution of the collisions in the hash table.

When S2 is used for resolution of collisions in the hash table the appropriate retrieval algorithm may be obtained from the above algorithm by initializing L=0 before step 1, and by placing the following step after step 11.

Step 12: Set L=L+1; if L=1 go to step 3.

Thus in step 3 the first address location is calculated according to the first hashing scheme; then in the second iteration the secondary address location is calculated according to the second hashing scheme.

Scheme S3 for collision resolution in the hash table can be used in conjunction with scheme S1, S2 or S4. When this scheme is used in conjunction with any scheme, say S2, the retrieval algorithm is the same as the one given for that scheme.

4.2.2 RETRIEVAL OF TRUNCATED WORDS FOR VARIOUS COLLISION METHODS IN DICT1

Step 9 of any one of the two retrieval algorithms in section 4.2.1 depends on the particular scheme used for collision resolution in the dictionary DICT1. Suppose L is the length of the truncated term; then the retrieval scheme for each collision resolution in the dictionary DICT1 will be as follows:

1- RETRIEVAL SCHEME FOR ALGORITHM 1

Step 9.1: If location (HASH1(H2,H1),H2) in DICT1 is empty then the descripter is not in memory hierarchy.

Step 9.2: Compare the first L characters of location (HASH1(H1,H2),H2) in the dictionary DICT1 with the truncated word; if a match is found then store the term in a temporary buffer.

Step 9.3: the result is in the temporary buffer.

2- RETRIEVAL SCHEME FOR ALGORITHMS 2 AND 3

Step 9.1: If location (HASH1(H1,H2),H2) in DICT1 is empty then the descripter is not in the memory hierarchy.

Step 9.2: Compare the first L characters of location (HASH1(H1,H2),H2) in the dictionary DICT1 with the truncated word; if a match is found then store the term in a temporary buffer.

Step 9.3: If location (HASH1(H1,H2),H1) in DICT1 is empty then

the descriptor is not in the memory hierarchy.

Step 9.4: Compare the first L characters of location (HASH1(H1,H2),H1) in the dictionary DICT1 with the truncated word; if a match is found store the term in a temporary buffer.

Step 9.5: The result will be in the temporary buffer.

3- RETRIEVAL SCHEME FOR ALGORITHMS 4,5 AND 6

This scheme is obtained from the scheme 3 in section 4.1.2 by replacement of step 3.3 by the following step:

Step 3.3: Compare the first L characters of location (HASH1(H1,H2),H2) in the dictionary DICT1 with the truncated term, if a match is found then store the term in the temporary buffer.

It may be noted that after execution of this algorithm the result will be in the temporary buffer.

4.3. RETRIEVAL ALGORITHM FOR NEW COLLISION SCHEME

This method is particularly applicable to a stable file or key set in which changes or insertions do not often arise. Thus it is very important to minimize the number of probes required to look up keys that are already in the table. The number of probes required to look up (and perhaps insert) keys that are not already there is not so important.

The basis of this method, which is described in detail in section 3.6, is that considerable care be taken when keys are inserted in order to attempt to reduce the number of probes required for subsequent look-ups.

In this method each entry in the hash table has a variable bucket size. When the bucket size is greater than one it is advantageous to first enter the keys that occur with high frequency in the search queries. They will then tend to appear near the beginnings of the bucket and hence require little CPU time for their subsequent look-up.

4.3.1 RETRIEVAL ALGORITHM TO SEARCH FOR A TERM IN DICT1

In the following algorithm ITEMP is a temporary location. ISTART and IEND indicate the start and end of a substring that may contains the search term. If the term is not in that substring then it is not in the first level store. LROW indicates the last row in the hash table and is a constant.

Step 1: If the length of the descripter is longer than the longest element in DICT1 go to step 8.

Step 2: Specify word length L, and calculate H1 according to the hashing scheme used.

Step 3: If location (H1,L) in the entry table is empty then go to step 7.

Step 4; Set ITEMP=0.

Step 5: For I=1 to L-1 do ITEMP=ITEMP+HASH1(LROW,I)*I.

Step 6: If H1=1 then ISTART=ITEMP, else ISTART=ITEMP+HASH1(H1-1,L)*L; set IEND=ITEMP+HASH1(H1,L)*L.

Step 7: Starting from ISTART to IEND compare every L characters with the search term. If there is a match then exit.

Step 8: Search for the term in dictionary DICT2.

In the above algorithm it is assumed that all words are stored in a linear array in such a way that all words of equal length are stored in a set of adjacent words. No blanks are placed between adjacent words. Words of greater length are stored immediately after words of lower length, and are stored in the same fashion. No blanks are used to separate words of different lengths.

The above algorithm does not require an index table at retrieval time since the hash table is organized in such a way that it contains all information required for retrieval. However, use of an index table, which indicates the starting position of each word length, facilitates execution of the retrieval algorithm since steps 3 and 4 of the above algorithm may then be eliminated, and in step 6 the first value of ITEMP may be extracted from the index table.

In use of this method the average number of locations examined in order to retrieve a term is approximately equal to that needed in the separate chaining method. But no space is wasted for pointer fields except for the pointer used in the entry table to indicate the starting location of this entry in

the dictionary. Also there is no need to calculate the address of the location of the next entry since all items are stored in sequential order.

4.3.2 RETRIEVAL ALGORITHM TO SEARCH FOR TRUNCATED TERMS IN DICT1

The algorithm to search for terms specified in truncated form is obtained from the retrieval algorithm in section 4.3.1 by the following two changes.

a) By replacement of step 2 by steps:

Step 2.1: Calculate H1 according to the hashing scheme used.

Step 2.2: From L=1 to longest term in the first level repeat the following four steps.

b) By the replacement of step 7 by the following step:

Step 7: Starting from ISTART compare the first L1 (length of word specified in truncated form) characters with the search term, if it is a match then exit; otherwise, advance ISTART by L characters and repeat this step until either a match occurs or ISTART is equal to IEND.

4.4 ESTIMATED TIMES FOR COMPUTATION AND DISK ACCESSES

In this section consideration is given to the average number of disk accesses required for various collision schemes.

Minimizing the average number of disk accesses per look-up is a traditional goal of many search techniques. Unfortunately, we

were unable to treat this problem analytically for various collision schemes. This is because in the present experiment there may be two different types of collisions (as already explained in sections 3.3 and 3.4) which makes it difficult to analyze the average number of disk accesses.

Tables I-LXXXI in appendix I present the results obtained for various collision schemes and for different numbers of terms. The columns NCH and NCD of these tables refer to collisions in the hash table and the dictionary respectively. The sum of NCD and NCH indicates the number of unresolved collisions in DICT1, and each such collision requires a disk access.

The time T required to retrieve one element from the first level of storage is equal to T1+T2, where T1 is the time required to compute the hashing function, and T2 is the time required to verify the presence of an entry in the table. The total time T is different for different collision schemes, but is of the order of only a few microseconds in all cases.

The second level store contains a set of entries. These entries are all of the same size of 640 characters. When the disk and channel are idle the average time to access a bucket is the sum of the average rotational delay and the record transmission time. For storage in a CDC Cyber 172/2 computer system, which can transfer 462 characters per millisecond, the average bucket access time is therefore 30+640/462=31.38 milliseconds. After transmission of a bucket content to the main

an element is also equal to T1+T2. This time is calculated in terms of microseconds, and is negligible. However the time required to retrieve an element in the first level according to any collision resolution scheme is much less than the time required to retrieve an element in the second level.

To search for terms specified in truncated form by use of any collision resolution scheme requires at least one disk access to make sure that all terms are retrieved. The number of disk accesses required to retrieve terms in truncated form is the same for all collision resolution schemes. This is a possible advantage of use of scheme S1 for solving collisions in the hash table and algorithm 1 for solving collisions in the dictionary. When scheme S1 is used in conjunction with algorithm 1 it requires a relatively simple retrieval algorithm in which all collisions are resolved by storage in second level storage even if storage is available in the first level.

CHAPTER 5

ORGANIZATION OF LEVEL 2

5.1 SPECIFICATION

In a file of records with no inherent ordering property the access to any random record is fastest when a random access method is used. In such a method each record has a key that is transformed to an address location within the file. If the file is maintained on a disk it is convenient for each track to be an addressable location, since it can then be searched by one channel command. Since each addressable location will probably contain more than one record it is called a bucket (storage area), and the number of records it can contain determines the bucket size.

In the CDC Cyber 172/2 the smallest physical disk file has space for 640 characters. Thus in the second level a storage area (bucket) size of 640 characters is used.

The second level of storage is subdivided into 1024 sublevels (buckets). Each SUBLEVEL(I) is of the same form as in LEVEL1 except that the hash table HASH2 has 8 rows and 8 columns, and hence a total storage for 65536 different terms. For each pair of values H1, H2 the terms that cannot be stored in DICT1 because of collisions are stored in a dictionary DICT2 of SUBLEVEL(I). The value of I is determined by a hashing function H(K1), where

K1 is derived from the original in the same manner as for LEVEL1.

The key K1 is processed by a hashing algorithm to yeild an integer between 1 and 1024. This integer designates one of 1024 SUBLEVEL(I). Thus there is a tendency to form clusters associated with prefixes. For example, words such as compute, computer, and computers, will be assigned to the same dictionary DICT2, and a query that contains "comp*" will be answered by scanning the dictionary DICT2 selected by evaluating H(COMP). It is to be noted that queries that contain words in left truncation form cannot be answered in the second level store since the organization of the second level store is for the capabability of answering queries specified in right or left truncation, but not the both. In order to limit queries in left trucation form the hashing function H(K2) should be used instead of H(K1), where K2 is derived from the original word by the same method as in the LEVEL1. It may be expected that right truncation will occur more frequently in queries than will left truncation and so the second level store is designed to allow queries to be specified in right truncation form.

Each SUBLEVEL(I) contains an index table of 8 entries and term strings of 8 terms. Thus a maximum of 64 terms can be stored in each SUBLEVEL(I), but 64 storage locations may also be obtained by use of an index table of 16 entries and term strings of 4 terms. In contrast to a dictionary DICT2 of the form 8x8 or 16x4, a DICT2 of the form 16x4 is preferable since it gives less collisions for the same reason that was mentioned in the case of

DICT1 (section 3.6).

5.2 HASHING SCHEME

It is necessary to convert the key H(K1) to fit in the range of the available address space (1 to 1024).

First the key is converted to digits. In the choice of the key-to-digits conversion, method B of section 2.2 may be preferable since it produces a large key length, and so is more appropriate for large address spaces.

Second, the number of digits or bits of the (modified) key is reduced to the range of the address space by using a hashing function. The midsquare method may be better than the division method since, although the division method gives good performance with a small address space [26], it is inferior for a large address space. Furthermore it is convenient to set up a hash table whose length is a power of two. Such a table size is desirable in order to allow efficient use of storage.

Similar to addressing LEVEL1, two hashing functions H3(K1) and H4(K2) may be used to address an element in a SUBLEVEL(I), where K1 and K2 are derived from the original word in a similar manner to that used in LEVEL1. If hash tables of the size 16x4 are used then H3(K1) and H4(K1) will have hash values in the range 1 to 16 and 1 to 4 respectively.

In each SUBLEVEL(I), similar to the LEVEL1, there may be two different types of collisions, namely collisions in the hash table and collisions in the dictionary.

The first step in the attempt to resolve collisions in each SUBLEVEL(I), is to find an unoccupied location in the same storage area. All collision resolution schemes proposed for use at the first level may also be used in each SUBLEVEL(I). However, a strong attempt should be made to minimize the number of unresolved collisions in each SUBLEVEL(I). Any unresolved collisions in a SUBLEVEL(I) may be resolved by storing in another SUBLEVEL(I), but this requires another disk access. Thus a combination of schemes S3 and S4 is recommended for resolution of collisions in each hash table HASH2, and algorithm 5 or 6 is recommended for resolution of collisions in each dictionary DICT2. Furthermore, when schemes S3 and S4 are used for resolution of collisions then in each dictionary DICT2 the location of terms in each term string will be purely sequential.

5.3 COLLISION RESOLUTION SCHEME

For each term that is not already stored in the first assigned SUBLEVEL(I) a location must be found in another SUBLEVEL(I). This determines the search method. A common method is through open addressing. Here, if no location is found in the first SUBLEVEL(I), according to the hashing algorithm used, each successive I+1, I+2,.... SUBLVEL2 is searched until a vacant

position is found. If the end of the file is reached the search continues from the beginning until reaching location I-1.

The open addressing method is to be prefered over other collision handling methods (chaining and quadratic search) since in minimizing the retrieval time the real concern is, not with identifier comparisons, but with secondary memory accesses. The successive records inspected by the open method are likely to lie on a single track and so this method may require less total time than the other methods. Both chaining algorithms could, of course, be implemented so that, whenever possible, overflow records are stored on the same track as their home address.

Because of its random probing of secondary memory, the quadratic technique of collision resolution is likely to result in a relatively poor retrieval time. In comparison to the open and chaining techniques it may be remarked that with a chained technique the pointer required in each bucket constitutes some overhead in storage but its amount is small in comparison to the size of the typical bucket. More significant storage overheads are the unutilized storage slots that occur in buckets with a low loading factor. When the maximum utilization of available storage space is a primary design consideration a technique of chaining collision resolution is likely to be the best alternative. When speed of retrieval is critical the open overflow method is best; as space becomes a dominant concern its advantages quickly disminish and only at a very high load factor does the open overflow method compare favorably with chained

overflow. While storage requirements are approximately equal for the open and chained techniques, access to overflow buckets is fastest with the open method [47]. However, in an attempt to minimize the number of disk accesses, a load factor not greater than, say, .85 or .80 could be used. Because of the large storage area in each SUBLEVEL(I) (64 terms) collisions in each SUBLEVEL(I) can be resolved by finding an unoccupied location in the same sublevel. Then when the load factor is low one disk access might be sufficient for retrieval of a term from the second level storage.

5.4 ELIMINATING THE HASH TABLES IN THE SECOND LEVEL STORE

Further improvement in retrieval time can be obtained by eliminating the hash table in the second level store. Even if a combination of schemes S3 and S4 is used for resolution of collisions in the hash table, some terms should still be stored in another SUBLEVEL(I), even although space is available in the first assigned SUBLEVEL(I). However it is possible to extend the scheme S4 of collision resolution and to search the entire hash table for an empty slot. But this requires a complicated retrieval algorithm and increases retrieval time if there is a high load factor.

Since in the second level each hash table contains only 64 entries, it may be reasonable to eliminate the hash table and to search for terms according to term length. In this case

organization of each DICT2 will be similar to the DICT1 as described in section 3.6. In this scheme an index table IND2 contains L pointers PI2 that indicate the address of the term strings, where L is the number of different word lengths in that SUBLEVEL(I). Associated with each pointer PI2 is a field that indicates the length of the corresponding terms in characters. Similar to the scheme presented in section 3.6, the program is executed twice. At the first phase storage is allocated for each term length in the dictionary. In the second phase the DICT2 are organized in such a way that words of greater length preceed words of smaller length. Thus the IND2 organization is in ascending order of word lengths.

There are two reasons for the use of this scheme: The first reason is to save storage. Since storage of hash tables HASH2 is not required in use of this scheme it may allow an increase in the number of entries in each SUBLEVEL(I). Second, because of the small address space for each SUBLEVEL(I), and storage of terms with the same length in a term string, the hash table may give rise to many collisions. This results in low utilization of storage. If all collisions are resolved by finding somehow an unoccupied location in the same SUBLEVEL(I), the average search length will be increased. It is clear that in use of hash tables in the second level the retrieval time may not be improved, since it requires calculation of hashing functions H3(K1) and H4(K2) and also extra access to the hash table.

5.5 DICTIONARY SEARCH FOR TERMS IN DICT2

When any proposed collision scheme for DICT1 is used for solving collisions in a SUBLEVEL(I) then the retrieval algorithm will be similar to the corresponding retrieval algorithm for LEVEL1. The only difference is when the search is a failure in the first assigned SUBLEVEL(I), in which event step 4 of the algorithm described below could be used.

When hash tables are not used in the second level store the following retrieval algorithm can be used to retrieve a term in a dictionary DICT2.

Step 1: Search the index table IND2 for the required length; if a match is not found go to step 4.

Step 2: Extract the starting and last positions of the appropriate term string from the index table IND2.

Step 3: Search the term string for the required descripter; if a match is found then exit.

Step 4: If storage is still available for a word of the required length then the descripter is not in the memory hierarchy; otherwise, according to the collision proccess, another SUBLEVEL2(I) may be scanned for the required descripter.

The above algorithm gives good performance for a successful search. When open addressing is used one of the following methods should be chosen to reduce the number of disk accesses for an unsuccessful search. The first method is to limit the number of disk accesses to a value M, which indicates the longest

search length. Then if M disk accesses do not find the descripter it is known to be not present in the memory hierarchy. The second method is to include in each SUBLEVEL(I) one field that indicates the search length in that sublevel. This method may give good performance with low load factors. However, there is a trade-off between the value of the load factor and the average number of disk accesses required to insure the presence of a descripter. If in each SUBLEVEL(I) all collisions are resolved by storing in unoccupied locations in the same SUBLEVEL(I) then a search for terms in truncated form may require that the entire dictionary DICT2 be inspected without any attention to the hash table. There is, of course, some advantage in a scheme that does not require any hash table. Furthermore, by eliminating the hash table in each SUBLEVEL(I), the unused space in the hash table may be used to store more terms in each SUBLEVEL(I).

CHAPTER 6

CONCLUSIONS

6.1 CONCLUSIONS AND FINAL REMARKS

The results reported in the presented thesis indicate that a satisfactory structure for a large term dictionary may be designed to make use of a two level storage hierarchy with two dimensional hash addressing. A two dimensional hash storage scheme allows rapid search for question terms that are specified in truncated form. A method of random access file organization (with reasonable load factors) allows a record generally to be retrieved by a single memory access. Addressing via such a randomizing transformation therefore compares favorably with other techniques such as scanning, binary searching, and various indexing schemes, which all require multiple memory accesses when used for data sets of a realistic size.

A small hash table has been used to address the most frequently accessed items, which are placed in core, and a number of fixed hash tables have been used to address all other terms in secondary memory.

Several factors affecting performance of the two dimensional hashing scheme have been identified. For each of these factors several algorithms have been designed, implemented, and studied. The performance and pecularities of each method were discussed,

and retrieval algorithms given.

The study reported in this thesis provides practical guidelines to design an appropriate dictionary structure for use with an inverted file organization. The results contained in appendex I provide an indication of the effect of varying the major parameters, and may be used for design guidelines. The study covered a wide range and the results may be applied to satisfy many design objectives for files that vary from static key sets to dynamic environments, and which require different degrees of time and space trade-offs. After an appropriate dictionary structure is selected then corresponding transformation and retrieval algorithms, as described in chapters 3 and 4 respectively, may be used in the implementation.

In the first attempt to improve the performance of the two dimensional hash addressing the following topics were investigated.

1- Key to digits conversion

2- Different hashing functions

3- Choice of divisor

The above topics have been considered by several other authors [25,26,27] in the single hashing environment, but in this thesis the effects of these parameters in two dimensional hashing techniques have been examined. One major difference is that in the two dimensional hashing scheme used in this thesis not more than four characters are derived from the original word and used

as a key.

Three different methods of key-to-digits conversion have been applied throughout the experiments, and each method has been examined with different hashing schemes and different divisors. It was noticed that method B for key-to-digits conversion produces a large key length, which is more appropriate for some hashing functions, while method A and C produce relatively small key lengths.

Prime, odd but not prime, and even numbers were chosen as divisors. It is observed that even divisors give very poor results when used with method B for key-to-digits conversion, and produce relatively good results when used with the two other key-to-digits conversion methods. No significant variation in performance was discovered among method A and C for key-to-digits conversion using different divisors, although in method C the uniqueness of the alphabetic key is preserved. The investigation of best choice of divisor indicates that the results are generally in agreement with Buchholz [26], who recommends the use of prime divisors slightly smaller than the table size. Prime divisors are more advantageous when collided terms are to be stored in a prime area. In particular, the divisor as a prime number of the form $4n+3$ (n is integer) proves convenient for quadratic search.

It is observed that when terms are loaded in decreasing order of frequency in two levels of storage hierarchy there is no

advantage in use of a virtual hash table since there is a non zero probability of having collisions even for the most frequent terms. This tends to increase the expected retrieval time since some frequently used terms are stored in a number of sublevels, SUBLEVEL(I), that are located by a disk access.

Examination of the literature shows that the hashing functions Division and Midsquare have proved superior to the other transformation methods, and they are used in the present thesis. Comparison of these two methods showes that for a large address space the midsquare method is preferable. For a small address space the division method performs slightly better than the midsquare. The midsquare method uses a bits oriented transformation and allows efficient use of storage. In the division method when the divisor is not sufficiently close to the table size some storage locations are not accessible through a computed address. Furthermore in two dimensional hash addressing, even when the divisor is sufficiently close to the table size, there is still a considerable amount of unaccessible storage. For instance, when a divisor of 31 is used for a table of size 32x32 there are still (32x32)-(31x31)=62 unused locations. This is undesirable in a two dimensional hashing scheme and is a possible disadvantage of the division method. However, several differently dimensioned hash tables are possible for a given table size. Thus when using the division method the table size that is most suitable for a given pair of divisors should be chosen.

Due to the special organization chosen for the two dimensional hashing there are two kinds of collision, namely collisions in the hash table and collisions in the dictionary.

Four basic schemes were designed and implemented for resolution of collisions in the hash table. They differ in the degree of their attempt to store the collided term in the prime area. However if a scheme fails then the corresponding term will be stored in a SUBLEVEL(I). In scheme S1 no attempt is made to store collided terms in the prime area, whereas schemes S2, S3, and S4 involve progressively increasing attempts to store the term in a prime area. In scheme S3 for collision resolution in the hash table more than one item may be hashed into each entry of the hash table. This resolves collisions in the hash table without any expense of complication of the retrieval algorithm. Finally, in scheme S4 for storage of a collided term all locations in the corresponding row of the hash table are examined for an unoccupied location. It may be noted that in all these schemes a limited search is used and the term may, in fact, be stored in the second level while storage is still available in the first level. In general, the time taken to search the LEVEL1 in core may be assumed to be negligible in comparison to the time required to access a SUBLEVEL(I) on disk. However, if a full table search is used for storage of collided terms it increases the search length in any subsquent look-up, and also require a full table search when question terms are specified in truncated form.

Six algorithms have been developed and implemented for resolution of collisions in the dictionary. They vary in the extents to which the location of terms in each term string are specified by H2(K2), and to what extent the other locations in the appropriate term string need to be scanned for an unoccupied location. Finally, in algorithm 6 the retrieval time is decreased at the expense of increased transformation time. This method is therefore recommended for use with a stable file in which updates occur very seldom.

When scheme S1 and algorithm 1 are used in resolution of collisions the retrieval algorithm can be simplified, but it may not make effective use of LEVEL1. However this may be preferable in a situation in which question terms are specifed in truncated form. For the use of any collision scheme a minimum of one disk access is required to ensure that all terms have been retrieved. Investigation of different methods of solving collisions led to a new scheme (section 3.6) that resolves all collisions by storing them in LEVEL1 and yet requires only one probe per look-up. In use of this scheme no disk accesses are required for the first 1024 most frequent words. However, this scheme is designed to simplify the retrieval algorithm at the expense of complicating the transformation algorithm [32,46]. The approach is justified by considering that the transformation is performed only once, but the retrieval is performed many times. In this scheme, similar to the scheme S3, a hash table and dictionary of different size could be used since each entry in the hash table

is treated as a variable size of bucket. Any increases in the size of the hash table decrease the average bucket size and expected retrieval time. However, a trade off between storage cost and rapid access should be expected.

The effects of differently dimensioned hash tables MxN of the same size have been examined. It is found that any increases in M (which decreases N) decrease the retrieval time needed to search for question terms in right truncated form. Also any increases in M allow efficient use of storage in the dictionary at the expense of the storage overhead in the index table.

In order to make the results of the experiment more reliable, the effect of the above factors has been examined with respect to two data sets. More than 80 experiments have been performed, and the results are presented in tabulated form in appendix I with respect to the measures indicated in section 3.7.

In the second level store there are 1024 different sublevels. The size of these sublevels is selected to be the same as the size of the unit of transfer between primary and secondary storage which is 640 characters (64 terms) for the CDC Cyber 172/2. Each sublevel in the second level has 64 entries, and hence a total storage for 65,536 different terms. Thus a maximum of 65536+1024=66560 terms could be stored in a two level storage hierarchy. However, in order to decrease the number of disk accesses the load factor should not be too high.

In the second level of storage several methods were proposed for resolution of collisions. It is concluded that it is advantageous to eliminate the hash table in each SUBLEVEL(I), and to store terms in the dictionary according to the word length with use of an appropriate index table. This has the advantage of simplyfing the retrieval algorithm in the second level, and of decreasing the number of disk access. Also unused space in the hash table could be used to store more terms in each SUBLEVEL(I). Furthermore, because of the small size of each SUBLEVEL(I), the time taken to search a SUBLEVEL(I) is negligible in comparison to the time taken to access another SUBLEVEL(I) on disk.

Suppose D different terms are stored in decreasing order of frequency within a two level storage hierarchy LEVEL1 and LEVEL2. For different sizes of LEVEL1 the proportion of two level storage has been computed according to the Zipf law and is shown in table 6.1 for values of D between 33280 and 66530. The corresponding values of load factor are also given. It was observed that increases in the size of the dictionary DICT1 increase the proportion of two level storage and result in rapid retrieval. However, because of collision problems, in practice this proportion will be lower than the one given in table 6.1.

TABLE 6.1

D	LOAD FACTOR	DICT1									
		1024	2048	3072	4096	5120	6144	7168	8192	9216	10240
33280	.50	.683	.746	.783	.809	.830	.846	.860	.872	.883	.893
36600	.55	.677	.740	.776	.802	.823	.839	.853	.865	.876	.885
39930	.60	.672	.734	.770	.796	.816	.832	.846	.858	.869	.878
43260	.65	.667	.729	.765	.791	.810	.827	.840	.852	.863	.872
46590	.70	.663	.724	.760	.785	.805	.821	.835	.847	.857	.866
49920	.75	.659	.720	.755	.781	.800	.816	.830	.841	.852	.861
53240	.80	.655	.716	.751	.776	.796	.812	.825	.837	.847	.856
56570	.85	.652	.712	.747	.772	.791	.807	.821	.832	.842	.852
59900	.90	.649	.708	.743	.768	.788	.803	.817	.828	.838	.847
63230	.95	.646	.705	.740	.765	.784	.800	.813	.824	.834	.843
66560	1.00	.643	.702	.737	.761	.780	.796	.809	.821	.831	.840

The proportion of data base terms contained in DICT1 for different values of LEVEL1 and D.

TABLE 6.1

D	LOAD FACTOR	DICT1									
		1024	2048	3072	4096	5120	6144	7168	8192	9216	10240
33280	.50	.683	.746	.783	.809	.830	.846	.860	.872	.883	.893
36600	.55	.677	.740	.776	.802	.823	.839	.853	.865	.876	.885
39930	.60	.672	.734	.770	.796	.816	.832	.846	.858	.869	.878
43260	.65	.667	.729	.765	.791	.810	.827	.840	.852	.863	.872
46590	.70	.663	.724	.760	.785	.805	.821	.835	.847	.857	.866
49920	.75	.659	.720	.755	.781	.800	.816	.830	.841	.852	.861
53240	.80	.655	.716	.751	.776	.796	.812	.825	.837	.847	.856
56570	.85	.652	.712	.747	.772	.791	.807	.821	.832	.842	.852
59900	.90	.649	.708	.743	.768	.788	.803	.817	.828	.838	.847
63230	.95	.646	.705	.740	.765	.784	.800	.813	.824	.834	.843
66560	1.00	.643	.702	.737	.761	.780	.796	.809	.821	.831	.840

The proportion of data base terms contained in DICT1 for different values of LEVEL1 and D.

The optimum sizes of LEVEL1, LEVEL2, and so forth, depend on the size of the data base and the available storage devices. It is clearly advantageous to store the most frequently used terms in main memory, but the advantage of rapid access must be weighed against the storage cost.

The two level storage hierarchies as described in this thesis are well suited for structuring term dictionaries since they allow both economy of storage and rapid look-up. The space savings result from the fact that the dictionary structure allows the sequential file to be stored in a compressed form using restricted variable length codes that give a compression ratio close to the optimum achievable through the use of Huffman codes. Also the dictionary is stored without inclusion of delimiting characters between terms.

The rapid look-up is due to the fact that the most frequently referenced items are stored in core and the less frequent items are stored in a number of fixed size blocks in a random access storage device. This tends to minimize the average search time.

Further investigation of the term dictionary indicates that the terms such as

THE, OF, AND, TO, A, IN, THAT, IS, WAS, HE, FOR, IT.

and so forth, which occur very frequently in titles and which are considered unimportant in formulation of question, may be excluded from LEVEL1. Also the inverted index may contain no entries for these terms. However, if restricted variable length

codes are used for compression coding, then the above most frequently occurring terms may be stored in fast storage in LEVEL0 for coding purposes. Three levels of storage improve the efficiency of coding when restricted variable length codes are used [48].

In this thesis some methods of structuring a term dictionary in two levels of storage have been presented. A possible extension of this thesis is to structure the sequential file on a random access storage device in such a way that accesses will be minimized with respect to a given mix of expected retrieval requests.

General guidelines for structuring a term dictionary are difficult to establish since various organizations have different performance criteria. However, before any decision is made regarding the organization of a term dictionary the following aspects must be considered.

1- Number of keys
2- Storage device specification
3- Machine cost per second
4- Frequency of update
5- Frequency of query
6- Frequency of occurrence of question terms in truncated form.
7- Relative importance of retrieval speed to percent storage overhead.

In the present investigation the performance of two dimensional hashing has been analyzed, and several methods have been developed and examined with respect to various factors. The advantages and disadvantages of each method have been described. On this basis an appropriate structure for a term dictionary could be selected. For instance, a combination of the schemes of sections 3.6 and 5.4 leads to the most appropriate dictionary structure for minimization of storage and access time. But updating is expensive and should preferably be done in off-time. Thus this dictionary structure is most useful in a stable file in which the data base is not updated frequently.

For any particular application, the two dimensional hash storage scheme may be varied to suit the different needs and to store the different information. However the advantages of two dimensional hash addressing, and use of two levels of storage utilization, will still apply.

REFERENCES

1. Hollaar, L.R., Stellhorn, W.H., "A Specialized Architecture for Textual Information Retrieval", National Computer Conference, 697-702, 1977.
2. Heaps, H.S., "Storage Analysis of a Compression Coding for Document Data Bases", INFOR, Vol. 10, No. 1, 47-61, Feb. 1972.
3. Cardenas, A.F., "Evaluation and Selection of File Organization", Comm. ACM, Vol. 16, No. 9, 540-548, Sept. 1973.
4. Price, C.E., "Table Lookup Techniques", Computing Surveys, Vol. 3, No. 2, 49-65, June 1971.
5. Service, D.G., "Identifier Search Machanisms: A Survey and Generalized Model", Computing Surveys, Vol. 6, No. 3, 175-194, Sept. 1974.
6. Rothnie, J.B.Jr., Lozano, T., "Attribute Based File Organization in a Paged Memory Environment", Comm. ACM, Vol. 17, No. 2, 63-69, Feb. 1974.
7. Mandelbrot, B.M., "An Informational Theory of the Statistical Structure of Language", Proc. Symp. Appl. Communication theory, W. Jackson, Ed., Butterworth, London, 1953.
8. Zipf, G.K., "Human Behaviour and the Principle of Least Effort", Addison-Wesley, Cambridge, Massachusetts, 1949.
9. Booth, A.d., "A 'Law' of Occurrence for Words of Low Frequency", Inf. And Contr., Vol. 10, 386-393, 1967.
10. Huffman, D.A., "A Method for Construction of Minimum Redundancy Codes", Proc. IRE, Vol. 40, 1098-1101, 1952.

11. Heaps, H.S., Thiel, L.H., "Optimum Procedures for Economic Information Retrieval", Inf. Stor. Retr., Vol. 6, 137-153, 1970.

12. Dimsdale, J.J., Heaps, H.S., "File Structure for an On-Line Catalog of One Million Titles", J. Library Automation, Vol. 6/1, 37-55, March 1973.

13. Heaps, H.S., "Data Compression of Large Document Data Bases", J. Chem. Inf. And Computer Sciences, Vol. 15, No. 1, 32-39, 1975.

14. Columbo, D.S., Rush, J.E., "Use of Words Fragments in Computer Based Retrieval Systems", J. Chem. Doc. 9, 47-50, 1969.

15. Clare, A.C., Cook, E. M., Lynch, M.F., "The Identification of Variable-Length, Equifrequent Character Strings in a Natural Language Data Base", Computer J., Vol. 15, 259-262, 1972.

16. Schuegraf, E.J., "Heaps, H.S., "Selection of Equifrequent Word Fragments for Information Retrieval", Inf. Stor. Retr, Vol. 9, 697-711, 1973.

17. Mayne, A., James, E.B., "Informaton Compression by Factorising Common Strings", Computer J. Vol. 18, 157-160, 1975.

18. Cardenas, A.F., "Analysis and Performance of Inverted Data Base Structures", Comm. ACM, Vol. 18, No. 5, 253-263, May 1975.

19. Murray, D.M., "A Scatter Storage Scheme for Dictionary Lookups", J. Library Automation, Vol. 3/3, 173-201, Sept. 1970.

20. Grimson, J.B., Stacey, G.M., "A Performance Study of Some Directory Structures for Large Files", Inf. Stor. Retr., Vol. 10, 357-364, 1974.

21. Morris, R. "Scatter Storage Techniques", Comm. ACM, Vol. 11, No. 1, 38-44, Jan. 1968.

22. Williams, J.G., "Storage Utilization in a Memory Hierarchy When Storage Assignment is Performed by a Hashing Algorithm", Comm. ACM, Vol. 14, No. 3, 172-175, March 1971.

23. Salasin, J., "Hierarchical Storage in Information Retrieval", Comm. ACM, Vol. 16, No. 5, 291-295, May 1973.

24. Knott, G.D., "Hashing Function", Computer J., Vol. 18, No. 3, 265-278, August 1975.

25. Sorenon, P.g., Tremblay, J.P., Deutscher, R.F., "Key-To-Address Transformation Techniques", INFOR, Vol. 16, No. 1, 1-34, October 1977.

26. Buchholz, W., "File Organization and Addressing", IBM Systems J., 86-111, June 1963.

27. Lum, V.Y., Yuen, P.S.T., Dodd, M., "Key-to-Address Transform Techniques: A Fundamental Performance Study on Large Existing Formatted Files", Comm. ACM, Vol. 14, No. 4, 228-239, April 1971.

28. Peterson, W.W., "Addressing for Random-Access Storage", IBM J. Res. Develop., 130-146, April 1957.

29. Knuth, D.E., Sorting and Searching, The Art of Computer Programming, Vol. 3, Addison-Wesley, Reading Mass., 506-549, 1973.

30. Luccio, F., "Weighted Increment Linear Search for Scatter Tables", Comm. ACM, Vol. 15, No. 12, 1045-1047, Dec. 1972.

31. Bell, J.R., Kamman, C.H., "The Linear Quotient Hash Code", Comm. ACM, Vol 13, No. 1, 675-677, Nov. 1970.

32. Brent, R.P., "Reducing the Retrieval Time of Scatter Storage Techniques", Comm. ACM, Vol. 16, No. 2, 105-109, Feb. 1973.

33. Maurer, W.D., "An Improved Hash Code for Scatter Storage", Comm. ACM, Vol. 11, No. 1, 35-38, Jan. 1968.

34. Radke, C.E., "The Use of Quadratic Residue Research" Comm. ACM, Vol. 13, No. 2, 103-105, Feb. 1970.

35. Day, A.C., "Full Table Quadratic Searching for Scatter Storage", Comm. ACM, Vol. 13, No. 8, 481-482, August 1970.

36. Bell, J.R., "The Quadratic Method: A Hash Code Eliminating Secondary Clustering", Comm. ACM, Vol. 13, No. 2, 107-109, Feb. 1970.

37. Hoppgood, F.R., Davenport, J., "The Quadratic Hash Method When the Table Size is a Power of 2", Computer J., Vol. 15, No. 4, 314-315, 1972.

38. Ackerman, A.F., "Quadratic Search for Hash Table of Size P**n", Comm. ACM, Vol. 17, no. 3, 164, March 1970.

39. Batagelj, V., "The Quadratic Hash Method When the Table Size is not a Prime Number", Comm. ACM, Vol. 18, No. 4, 216-217, April 1975.

40. Ullman, J.D., "A Note on the Efficiency of Hashing Functions", JACM, Vol. 19, No. 3, 569-575, July 1972.

41. Guibas, L.J., "Hashing Techniques that Exhibit Secondary or Tertiary Clustering", Second USA-JAPAN Computer Conference, 324-328, 1975.

42. Kucera, H, Francis, W,N., "Computational Analysis of Present-day American English", Brown University Press Providence, Rhode Island, 1967.

43. Bookstein, A., "Double Hashing", J. ASSIS, 402-405, Nov.-Dec. 1972.

44. Sprugnoli, R., "Perfect Hashing Functions: A Single Probe Retrieving Method for Static Sets", Comm. ACM, Vol. 20, No. 11, 841-850, Nov. 1977.

45. Bloom, B.H., "Space/Time Trade-Offs in Hash Coding with Allowable Errors", Comm. ACM, Vol. 13, No. 7, 422-426, July 1970.

46. Clapson, P., "Improving the Access Time for Random Access Files", Comm. ACM, Vol. 20, No. 3, 127-135, March 1977.

47. Severance, D., Duhne, R., "A Practitioner's Guide to Addressing Algorithms", Comm. ACM, Vol. 19, No. 6, 314-326, June 1976.

48. Kernizan, R., "Compact File Organization for Information Retrieval", Master Thesis, Department of Computer Science, Concordia University, May 1977.

APPENDIX I

TABLE OF TABLES

TABLE NO.	DATA SET	KEY-DIGITS CONVERISON (SEC. 2.2)	HASHING & DIVISOR	HASH1 SIZE	DICT1 SIZE	COLLISION RESOLUTION	
						HASH1 (SECTION 3.3)	DICT1 (3.4)
I	1	A	DIV 31,31	31X31	31X31	S1	1
II	2	A	DIV 31,31	31X31	31X31	S1	1
III	1	A	DIV 32,32	32X32	32X32	S1	1
IV	1	A	DIV 43,43	32X32	32X32	S1	1
V	1	B	DIV 31,31	31X31	31X31	S1	1
VI	1	B	MIDSQUARE	31X31	31X31	S1	1
VII	2	B	DIV 31,31	31X31	31X31	S1	1
VIII	2	B	MIDSQUARE	31X31	31X31	S1	1
IX	1	B	DIV 32,32	32X32	32X32	S1	1
X	1	B	DIV 43,43	32X32	32X32	S1	1
XI	1	B	MIDSQUARE	32X32	32X32	S1	1
XII	2	B	MIDSQUARE	32X32	32X32	S1	1
XIII	1	C	DIV 31,31	31X31	31X31	S1	1
XIV	2	C	DIV 31,31	31X31	31X31	S1	1
XV	1	A	DIV 31,31	31X31	32X32	S1	4
XVI	1	A	DIV 31,31	31X31	32X32	S1	5
XVII	1	B	DIV 31,31	31X31	32X32	S1	4
XVIII	1	B	DIV 31,31	31X31	32X32	S1	5
XIX	1	A,B	DIV 31,31	31X31	31X31	S2.1,S1	1
XX	1	A,B	DIV 33,33	32X32	32X32	S2.1,S1	1
XXI	1	B	DIV 31,31	31X31	31X31	S2.3,S1	2
XXII	1	B	DIV 31,31	31X31	31X31	S2.3,S1	3

TABLE OF TABLES

TABLE NO.	DATA SET	KEY-DIGITS CONVERISON (SEC. 2.2)	HASHING & DIVISOR	HASH1 SIZE	DICT1 SIZE	COLLISION RESOLUTION	
						HASH1 (SECTION 3.3)	DICT1 (3.4)
XXIII	2	B	DIV 31,31	31X31	31X31	S2.3,S1	3
XXIV	1	B	DIV 31,31	31X31	31X31	S2.3,S1	4
XXV	1	A	DIV 31,31	31X31	31X31	S2.3,S1	4
XXVI	1	A,B	DIV 31,31	31X31	32X32	S2.1,S1	4
XXVII	1	A,B	DIV 31,31	31X31	32X32	S2.1,S1	5
XXVIII	1	A,B	DIV 43,43	32X32	32X32	S2.1,S1	5
XXIX	1	B	DIV 31,31	31X31	32X32	S2.3,S1	5
XXX	1	A	DIV 31,31	31X31	32X32	S2.3,S1	5
XXXI	1	A	DIV 43,43 & 31,31	32X32	32X32	S2.2,S1	5
XXXII	1	B	DIV 43,43 & 31,31	32X32	32X32	S2.2,S1	5
XXXIII	1	B	DIV 31,31	31X31	31X31	S2.3,S3.2,S1	2
XXXIV	1	B	DIV 31,31	31X31	31X31	S2.3,S3.2,S1	3
XXXV	2	B	DIV 31,31	31X31	31X31	S2.3,S3.2,S1	3
XXXVI	1	A,B	DIV 31,31	31X31	32X32	S2.1,S3.1,S1	4
XXXVII	1	A,B	DIV 31,31	31X31	32X32	S2.1,S3.1,S1	5
XXXVIII	1	B	DIV 31,31	31X31	32X32	S2.3,S3.1,S1	5
XIL	1	A	DIV 31,31	31X31	32X32	S2.3,S3.1,S1	4
XL	1	A	DIV 31,31	31X31	32X32	S2.3,S3.1,S1	5
XLI	1	A	DIV 43,43 & 31,31	32X32	32X32	S2.2,S3.1,S1	5
XLII	1	B	DIV 43,43 & 31,31	32X32	32X32	S2.2,S3.1,S1	5
XLIII	1	A,B	DIV 31,31	31X31	32X32	S2.1,S3.3,S1	4
XLIV	1	A,B	DIV 31,31	31X31	32X32	S2.1,S3.3,S1	5
XLV	2	A,B	DIV 31,31 MIDSQUARE	31X31	32X32	S2.1,S3.3,S1	4

TABLE OF TABLES

TABLE NO.	DATA SET	KEY-DIGITS CONVERISON (SEC. 2.2)	HASHING & DIVISOR	HASH1 SIZE	DICT1 SIZE	COLLISION RESOLUTION	
						HASH1 (SECTION 3.3)	DICT1 (3.4)
XLVI	2	A,B	DIV 31,31	31X31	32X32	S2.1,S3.3,S1	5
XLVII	2	B	DIV 31,31	31X31	32X32	S2.4,S3.3,S1	5
XLVIII	2	B	DIV 31,31 MIDSQUARE	31X31	32X32	S2.4,S3.3,S1	6
XLIX	1	B	DIV 31,31 MIDSQUARE	31X31	32X32	S2.4,S3.3,S1	5
L	1	B	DIV 31,31 MIDSQUARE	31X31	32X32	S2.4,S3.3,S1	6
LI	1	A	DIV 139,7	139X7	139X7	S1	1
LII	1	A	MIDSQUARE	139X7	139X7	S1	1
LIII	2	A	DIV 139,7	139X7	139X7	S1	1
LIV	1	B	DIV 139,7	139X7	139X7	S1	1
LV	2	B	DIV 139,7	139X7	139X7	S1	1
LVI	1	B	MIDSQUARE	139X7	139X7	S1	1
LVII	1	B	MIDSQUARE	128X8	128X8	S1	1
LVIII	2	B	MIDSQUARE	128X8	128X8	S1	1
LIX	1	C	DIV 139,7	139X7	139X7	S1	1
LX	2	C	DIV 139,7	139X7	139X7	S1	1
LXI	1	C	MIDSQUARE	139X7	139X7	S1	1
LXII	2	C	MIDSQUARE	139X7	139X7	S1	1
LXIII	1	A,B	DIV 139,7	139X7	128X8	S2.1,S1	4
LXIV	1	A,B	DIV 139,7	139X7	128X8	S2.1,S1	5
LXV	1	A,B	DIV 139,7	139X7	128X8	S2.1,S3.1,S1	4
LXVI	1	A,B	DIV 139,7	139X7	128X8	S2.1,S3.1,S1	5
LXVII	1	A,B	DIV 139,7	139X7	128X8	S2.1,S3.3,S1	4
LXVIII	1	A,B	DIV 139,7	139X7	128X8	S2.1,S3.3,S1	5

TABLE OF TABLES

TABLE NO.	DATA SET	KEY-DIGITS CONVERISON (SEC. 2.2)	HASHING & DIVISOR	HASH1 SIZE	DICT1 SIZE	COLLISION RESOLUTION	
						HASH1 (SECTION 3.3)	DICT1 (3.4)
LXIX	2	A,B	DIV 139,7	139X7	128X8	S2.1,S3.3,S1	4
LXX	2	A,B	DIV 139,7	139X7	128X8	S2.1,S3.3,S1	5
LXXI	2	A,B	DIV 139,7	139X7	128X8	S2.1,S3.3,S1	6
LXXII	1	B	DIV 139,7 MIDSQUARE	139X7	128X8	S2.4,S3.3,S1	4
LXXIII	1	B	DIV 139,7 MIDSQUARE	139X7	128X8	S2.4,S3.3,S1	5
LXXIV	1	B	DIV 139,7 MIDSQUARE	139X7	128X8	S2.4,S3.3,S1	6
LXXV	2	B	DIV 139,7 MIDSQUARE	139X7	128X8	S2.4,S3.3,S1	4
LXXVI	2	B	DIV 139,7 MIDSQUARE	139X7	128X8	S2.4,S3.3,S1	5
LXXVII	2	B	DIV 139,7 MIDSQUARE	139X7	128X8	S2.4,S3.3,S1	6
LXXVIII	1	B	DIV 139,7	139X7	128X8	S3.3,S4.1,S1	5
LXXIX	1	B	DIV 139,7	139X7	128X8	S3.3,S4.1,S1	6
LXXX	2	B	DIV 139,7	139X7	128X8	S3.3,S4.1,S1	5
LXXXI	2	B	DIV 139,7	139X7	128X8	S3.3,S4.1,S1	6

TABLE I

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	97	5	0	0	.98971	.01123	88.128
204	181	23	0	0	.97132	.02910	33.380
307	264	43	0	0	.96038	.03988	24.082
409	328	77	0	4	.94615	.05404	17.509
511	380	120	0	11	.93191	.06823	13.658
614	422	172	0	20	.91814	.08198	11.200
716	461	220	0	35	.90664	.09346	9.701
819	486	272	0	61	.89436	.10572	8.459
921	514	329	0	78	.88456	.11552	7.657
1023	538	383	0	102	.87548	.12459	7.027
1126	560	443	0	123	.86712	.13294	6.523
1228	576	510	0	142	.85912	.14094	6.096
1331	590	580	0	161	.85162	.14843	5.738
1433	611	639	0	183	.84555	.15450	5.473
1535	619	704	0	212	.83887	.16117	5.205
1638	630	773	0	235	.83292	.16712	4.984
1740	643	834	0	263	.82761	.17243	4.800
1843	648	911	0	284	.82207	.17796	4.619
1945	659	970	0	316	.81735	.18268	4.474
2047	662	1043	0	342	.81244	.18759	4.331
2150	669	1118	0	363	.80802	.19201	4.208
2252	679	1183	0	390	.80405	.19598	4.103
2355	684	1249	0	422	.80000	.20003	3.999
2457	689	1321	0	447	.79620	.20383	3.906
2559	692	1387	0	480	.79250	.20753	3.819

TABLE II

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	94	8	0	0	.96988	.03106	31.225
204	181	23	0	0	.95688	.04354	21.976
307	267	36	0	4	.94920	.05106	18.591
409	323	74	0	12	.93182	.06837	13.629
511	376	115	0	20	.91829	.08185	11.219
614	418	163	0	35	.90439	.09572	9.448
716	458	207	0	51	.89382	.10628	8.410
819	493	261	0	65	.88360	.11648	7.586
921	522	318	0	81	.87408	.12599	6.937
1023	549	374	0	100	.86555	.13452	6.434
1126	576	437	0	113	.85793	.14213	6.036
1228	595	503	0	130	.85036	.14970	5.680
1331	613	563	0	155	.84336	.15669	5.382
1433	628	639	0	166	.83681	.16324	5.126
1535	647	705	0	183	.83114	.16890	4.921
1638	662	781	0	195	.82556	.17448	4.732
1740	672	855	0	213	.82009	.17995	4.557
1843	686	931	0	226	.81523	.18481	4.411
1945	697	1004	0	244	.81055	.18948	4.278
2047	702	1091	0	254	.80581	.19422	4.149
2150	711	1168	0	271	.80153	.19850	4.038
2252	717	1249	0	286	.79738	.20265	3.935
2355	725	1327	0	303	.79352	.20651	3.843
2457	733	1411	0	313	.78991	.21012	3.759
2559	741	1488	0	330	.78647	.21355	3.683

TABLE III

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	98	4	0	0	.98827	.01267	77.984
204	180	24	0	0	.96639	.03402	28.403
307	250	57	0	0	.94739	.05287	17.918
409	301	103	0	5	.92806	.07213	12.867
511	351	153	0	7	.91386	.08629	10.590
614	385	215	0	14	.89859	.10153	8.851
716	417	277	0	22	.88611	.11399	7.773
819	452	331	0	36	.87601	.12407	7.060
921	476	394	0	51	.86589	.13419	6.453
1023	497	468	0	58	.85666	.14341	5.974
1126	519	534	0	73	.84853	.15153	5.600
1228	539	598	0	91	.84115	.15890	5.294
1331	557	666	0	108	.83425	.16580	5.032
1433	574	735	0	124	.82796	.17209	4.811
1535	584	813	0	138	.82162	.17842	4.605
1638	592	896	0	150	.81557	.18447	4.421
1740	607	961	0	172	.81054	.18949	4.277
1843	618	1038	0	187	.80554	.19449	4.142
1945	625	1119	0	201	.80067	.19936	4.016
2047	631	1197	0	219	.79605	.20398	3.903
2150	634	1279	0	237	.79148	.20855	3.795
2252	644	1351	0	257	.78761	.21242	3.708
2355	648	1439	0	268	.78360	.21643	3.621
2457	654	1513	0	290	.77994	.22009	3.544
2559	660	1587	0	312	.77645	.22357	3.473

TABLE IV

NTERM	NDIGT	NCH	TCD	NCD	P1	P2	P1/P2
102	64	38	0	0	.62874	.37221	1.689
204	116	88	0	0	.61380	.38661	1.588
307	165	142	0	0	.60445	.39581	1.527
409	210	199	0	0	.59720	.40299	1.482
511	248	263	0	0	.58995	.41020	1.438
614	290	323	0	1	.58514	.41498	1.410
716	313	398	0	5	.57739	.42271	1.366
819	353	456	0	10	.57392	.42617	1.347
921	383	521	0	17	.56950	.43057	1.323
1023	406	596	0	21	.56469	.43538	1.297
1126	425	674	0	27	.55989	.44018	1.272
1228	447	749	0	32	.55601	.44405	1.252
1331	472	817	0	42	.55278	.44727	1.236
1433	486	892	0	55	.54886	.45118	1.217
1535	499	969	0	67	.54519	.45485	1.199
1638	514	1046	0	78	.54193	.45811	1.183
1740	530	1117	0	93	.53903	.46101	1.169
1843	543	1194	0	106	.53611	.46393	1.156
1945	562	1269	0	114	.53380	.46623	1.145
2047	572	1345	0	130	.53109	.46895	1.133
2150	582	1431	0	137	.52850	.47153	1.121
2252	594	1507	0	151	.52621	.47382	1.111
2355	603	1588	0	164	.52385	.47617	1.100
2457	610	1674	0	173	.52155	.47847	1.090
2559	615	1754	0	190	.51927	.48075	1.080

TABLE V

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	91	11	0	0	.96498	.03596	26.836
204	173	31	0	0	.94819	.05223	18.156
307	249	58	0	0	.93447	.06579	14.204
409	315	90	0	4	.92208	.07811	11.805
511	376	126	0	9	.91146	.08868	10.278
614	429	166	0	19	.90111	.09900	9.102
716	475	216	0	25	.89143	.10867	8.203
819	513	273	0	33	.88182	.11826	7.456
921	549	331	0	41	.87339	.12668	6.894
1023	581	395	0	47	.86554	.13453	6.434
1126	603	461	0	62	.85731	.14276	6.005
1228	624	528	0	76	.84995	.15010	5.662
1331	645	592	0	94	.84325	.15680	5.378
1433	661	665	0	107	.83678	.16327	5.125
1535	672	741	0	122	.83044	.16960	4.896
1638	680	821	0	137	.82431	.17573	4.691
1740	691	901	0	148	.81893	.18111	4.522
1843	703	971	0	169	.81394	.18609	4.374
1945	713	1051	0	181	.80921	.19082	4.241
2047	726	1127	0	194	.80497	.19506	4.127
2150	733	1209	0	208	.80058	.19945	4.014
2252	739	1295	0	218	.79644	.20359	3.912
2355	745	1375	0	235	.79248	.20755	3.818
2457	749	1449	0	259	.78867	.21136	3.731
2559	758	1527	0	274	.78529	.21474	3.657

TABLE VI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	97	5	0	0	.98409	.01686	58.381
204	185	19	0	0	.96995	.03047	31.833
307	262	44	0	1	.95601	.04425	21.603
409	325	79	0	5	.94134	.05865	16.054
511	372	124	0	15	.92582	.07433	12.456
614	423	164	0	27	.91440	.08572	10.668
716	456	220	0	40	.90175	.09835	9.168
819	479	280	0	60	.88919	.11090	8.018
921	508	335	0	78	.87964	.12044	7.304
1023	526	398	0	99	.86981	.13026	6.678
1126	553	459	0	114	.86212	.13794	6.250
1228	567	528	0	133	.85396	.14610	5.845
1331	590	594	0	147	.84743	.15262	5.552
1433	609	658	0	166	.84121	.15884	5.296
1535	622	722	0	191	.83501	.16504	5.060
1638	633	793	0	212	.82908	.17096	4.850
1740	637	867	0	236	.82314	.17690	4.653
1843	647	941	0	255	.81798	.18205	4.493
1945	659	1014	0	272	.81336	.18668	4.357
2047	666	1080	0	301	.80871	.19132	4.227
2150	676	1154	0	320	.80448	.19555	4.114
2252	686	1230	0	336	.80054	.19949	4.013
2355	691	1308	0	356	.79651	.20352	3.914
2457	698	1381	0	378	.79282	.20720	3.826
2559	700	1462	0	397	.78909	.21093	3.741

TABLE VII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	96	6	0	0	.98553	.01541	63.939
204	184	20	0	0	.97130	.02912	33.356
307	259	48	0	0	.95507	.04519	21.133
409	322	83	0	4	.94033	.05980	15.710
511	368	130	0	13	.92446	.07569	12.214
614	415	178	0	21	.91216	.08796	10.370
716	467	223	0	26	.90365	.09645	9.369
819	511	268	0	40	.89486	.10523	8.504
921	552	315	0	54	.88704	.11304	7.847
1023	589	377	0	67	.87831	.12175	7.214
1126	602	445	0	79	.87007	.12999	6.693
1228	615	518	0	95	.86170	.13836	6.228
1331	633	592	0	106	.85459	.14546	5.875
1433	650	662	0	121	.84811	.15194	5.582
1535	663	731	0	141	.84184	.15820	5.321
1638	672	815	0	151	.83570	.16434	5.085
1740	685	890	0	165	.83038	.16965	4.895
1843	701	965	0	177	.82559	.17445	4.733
1945	712	1040	0	193	.82084	.17919	4.581
2047	723	1119	0	205	.81640	.18363	4.446
2150	728	1202	0	220	.81184	.18819	4.314
2252	735	1279	0	238	.80768	.19235	4.199
2355	742	1358	0	255	.80372	.19631	4.094
2457	752	1440	0	265	.80015	.19988	4.003
2559	759	1518	0	282	.79662	.20341	3.916

TABLE VIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	96	6	0	0	.98330	.01764	55.744
204	178	26	0	0	.96328	.03713	25.941
307	258	46	0	3	.95144	.04882	19.490
409	320	78	0	11	.93640	.06379	14.679
511	366	122	0	23	.92074	.07940	11.596
614	418	162	0	34	.90992	.09020	10.088
716	457	207	0	52	.89846	.10164	8.840
819	494	260	0	65	.88858	.11151	7.969
921	519	324	0	78	.87840	.12167	7.219
1023	539	385	0	99	.86886	.13120	6.622
1126	569	445	0	112	.86158	.13848	6.222
1228	587	515	0	126	.85386	.14619	5.841
1331	605	583	0	143	.84680	.15325	5.526
1433	621	655	0	157	.84031	.15974	5.261
1535	637	722	0	176	.83437	.16567	5.036
1638	652	788	0	198	.82877	.17127	4.839
1740	662	864	0	214	.82327	.17677	4.657
1843	672	937	0	234	.81812	.18192	4.497
1945	684	1003	0	258	.81349	.18654	4.361
2047	696	1087	0	264	.80916	.19087	4.239
2150	710	1165	0	275	.80516	.19487	4.132
2252	718	1242	0	292	.80110	.19893	4.027
2355	725	1322	0	308	.79717	.20286	3.930
2457	735	1398	0	324	.79363	.20639	3.845
2559	742	1479	0	338	.79013	.20989	3.764

TABLE IX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	63	39	0	0	.82858	.17236	4.807
204	105	99	0	0	.77879	.22163	3.514
307	132	175	0	0	.74522	.25504	2.922
409	153	255	0	1	.72191	.27828	2.594
511	168	340	0	3	.70320	.29695	2.368
614	178	430	0	6	.68731	.31281	2.197
716	193	513	0	10	.67570	.32440	2.083
819	206	601	0	12	.66557	.33452	1.990
921	217	687	0	17	.65672	.34336	1.913
1023	227	774	0	22	.64889	.35118	1.848
1126	235	862	0	29	.64168	.35838	1.791
1228	244	950	0	34	.63543	.36462	1.743
1331	252	1043	0	36	.62965	.37040	1.700
1433	259	1136	0	38	.62437	.37568	1.662
1535	265	1230	0	40	.61946	.38059	1.628
1638	271	1322	0	45	.61489	.38515	1.596
1740	274	1416	0	50	.61049	.38955	1.567
1843	278	1513	0	52	.60643	.39361	1.541
1945	279	1608	0	58	.60249	.39755	1.516
2047	282	1703	0	62	.59892	.40112	1.493
2150	286	1801	0	63	.59558	.40445	1.473
2252	291	1892	0	69	.59253	.40750	1.454
2355	295	1984	0	76	.58956	.41047	1.436
2457	297	2078	0	82	.58668	.41335	1.419
2559	300	2174	0	85	.58399	.41604	1.404

TABLE X

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	55	47	0	0	.51147	.48947	1.045
204	107	97	0	0	.51163	.48879	1.047
307	162	145	0	0	.51365	.48661	1.056
409	206	201	0	2	.51031	.48988	1.042
511	240	265	0	6	.50464	.49551	1.018
614	276	330	0	8	.50059	.49953	1.002
716	311	390	0	15	.49718	.50292	.989
819	344	457	0	18	.49387	.50622	.976
921	367	530	0	24	.48961	.51046	.959
1023	396	600	0	27	.48675	.51332	.948
1126	419	673	0	34	.48344	.51662	.936
1228	440	741	0	47	.48031	.51974	.924
1331	460	810	0	61	.47735	.52270	.913
1433	483	879	0	71	.47498	.52507	.905
1535	497	958	0	80	.47204	.52800	.894
1638	512	1040	0	86	.46938	.53066	.885
1740	531	1105	0	104	.46726	.53277	.877
1843	544	1187	0	112	.46483	.53520	.869
1945	558	1267	0	120	.46267	.53736	.861
2047	573	1340	0	134	.46070	.53933	.854
2150	585	1418	0	147	.45865	.54138	.847
2252	601	1495	0	156	.45697	.54306	.841
2355	610	1581	0	164	.45498	.54504	.835
2457	619	1664	0	174	.45313	.54690	.829
2559	626	1746	0	187	.45127	.54875	.822

TABLE XI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	96	6	0	0	.98189	.01905	51.538
204	181	23	0	0	.96579	.03462	27.893
307	261	46	0	0	.95320	.04706	20.256
409	326	76	0	7	.93977	.06042	15.555
511	379	113	0	19	.92591	.07424	12.472
614	420	159	0	35	.91213	.08799	10.366
716	454	211	0	51	.89973	.10037	8.964
819	486	261	0	72	.88886	.11122	7.992
921	517	319	0	85	.87961	.12046	7.302
1023	541	379	0	103	.87059	.12948	6.724
1126	567	438	0	121	.86277	.13729	6.284
1228	589	495	0	144	.85550	.14456	5.918
1331	608	550	0	173	.84852	.15153	5.600
1433	624	605	0	204	.84201	.15804	5.328
1535	631	675	0	229	.83529	.16475	5.070
1638	646	734	0	258	.82969	.17036	4.870
1740	655	803	0	282	.82411	.17592	4.685
1843	666	871	0	306	.81903	.18101	4.525
1945	674	941	0	330	.81413	.18590	4.379
2047	681	1007	0	359	.80949	.19055	4.248
2150	689	1071	0	390	.80513	.19490	4.131
2252	697	1142	0	413	.80107	.19896	4.026
2355	698	1223	0	434	.79683	.20320	3.921
2457	702	1296	0	459	.79300	.20703	3.830
2559	703	1372	0	484	.78922	.21081	3.744

TABLE XII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	99	3	0	0	.99385	.00710	140.042
204	178	26	0	0	.97053	.02989	32.468
307	252	51	0	4	.95426	.04600	20.745
409	313	84	0	12	.93865	.06154	15.254
511	356	132	0	23	.92191	.07824	11.783
614	397	186	0	31	.90807	.09205	9.865
716	438	231	0	47	.89718	.10292	8.717
819	464	289	0	66	.88529	.11480	7.712
921	506	338	0	77	.87779	.12229	7.178
1023	534	397	0	92	.86936	.13071	6.651
1126	559	453	0	114	.86146	.13860	6.216
1228	589	506	0	133	.85507	.14498	5.898
1331	615	567	0	149	.84882	.15123	5.613
1433	632	636	0	165	.84241	.15764	5.344
1535	654	697	0	184	.83695	.16309	5.132
1638	665	767	0	206	.83101	.16903	4.916
1740	680	837	0	223	.82588	.17416	4.742
1843	697	906	0	240	.82118	.17885	4.591
1945	711	973	0	261	.81666	.18337	4.454
2047	720	1044	0	283	.81212	.18791	4.322
2150	727	1122	0	301	.80770	.19233	4.199
2252	731	1194	0	327	.80340	.19663	4.086
2355	741	1266	0	348	.79962	.20041	3.990
2457	758	1340	0	359	.79641	.20361	3.911
2559	769	1411	0	379	.79309	.20693	3.833

TABLE XIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	0	0	.99690	.00404	246.486
204	182	22	0	0	.97494	.02548	38.270
307	262	40	0	5	.96296	.03730	25.813
409	330	67	0	12	.95003	.05016	18.940
511	377	111	0	23	.93409	.06605	14.141
614	419	158	0	37	.92029	.07983	11.529
716	459	208	0	49	.90904	.09106	9.983
819	496	251	0	72	.89883	.10126	8.877
921	517	305	0	99	.88784	.11224	7.910
1023	540	364	0	119	.87858	.12148	7.232
1126	563	419	0	144	.87034	.12972	6.709
1228	584	482	0	162	.86285	.13720	6.289
1331	600	545	0	186	.85552	.14453	5.919
1433	619	619	0	195	.84922	.15082	5.631
1535	632	682	0	221	.84295	.15710	5.366
1638	645	749	0	244	.83711	.16293	5.138
1740	651	824	0	265	.83127	.16877	4.925
1843	658	892	0	293	.82584	.17419	4.741
1945	667	964	0	314	.82097	.17906	4.585
2047	672	1031	0	344	.81616	.18387	4.439
2150	677	1103	0	370	.81159	.18844	4.307
2252	684	1178	0	390	.80744	.19259	4.193
2355	690	1246	0	419	.80342	.19661	4.086
2457	694	1320	0	443	.79956	.20047	3.988
2559	699	1394	0	466	.79594	.20409	3.900

TABLE XIV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	98	4	0	0	.98448	.01646	59.806
204	180	24	0	0	.96416	.03625	26.595
307	263	44	0	0	.95368	.04658	20.475
409	321	81	0	7	.93707	.06312	14.846
511	374	121	0	16	.92334	.07681	12.022
614	420	172	0	22	.91086	.08926	10.205
716	456	230	0	30	.89890	.10120	8.882
819	492	283	0	44	.88877	.11131	7.985
921	529	338	0	54	.88038	.11969	7.355
1023	556	400	0	67	.87176	.12830	6.795
1126	584	460	0	82	.86422	.13584	6.362
1228	613	518	0	97	.85769	.14236	6.025
1331	634	589	0	108	.85091	.14914	5.705
1433	654	657	0	122	.84475	.15530	5.439
1535	670	730	0	135	.83877	.16127	5.201
1638	681	810	0	147	.83282	.16722	4.980
1740	688	890	0	162	.82708	.17296	4.782
1843	697	972	0	174	.82183	.17821	4.612
1945	704	1054	0	187	.81685	.18319	4.459
2047	713	1133	0	201	.81231	.18772	4.327
2150	730	1203	0	217	.80845	.19158	4.220
2252	734	1289	0	229	.80416	.19587	4.105
2355	744	1368	0	243	.80036	.19966	4.009
2457	750	1457	0	250	.79661	.20341	3.916
2559	756	1531	0	272	.79305	.20697	3.832

TABLE XV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	97	5	0	0	.98971	.01123	88.128
204	181	23	0	0	.97132	.02910	33.380
307	264	43	0	0	.96038	.03988	24.082
409	332	77	4	0	.94775	.05244	18.073
511	390	121	12	0	.93530	.06485	14.423
614	439	175	22	0	.92327	.07684	12.015
716	492	224	38	0	.91456	.08554	10.692
819	536	282	59	1	.90553	.09456	9.576
921	568	347	75	6	.89618	.10389	8.626
1023	605	409	97	9	.88873	.11134	7.982
1126	629	474	119	23	.88046	.11960	7.362
1228	652	549	133	27	.87307	.12698	6.876
1331	666	627	149	38	.86543	.13462	6.429
1433	678	693	174	62	.85839	.14166	6.060
1535	688	770	196	77	.85178	.14826	5.745
1638	696	849	218	93	.84548	.15456	5.470
1740	705	924	240	111	.83979	.16025	5.241
1843	708	1008	259	127	.83403	.16600	5.024
1945	717	1077	288	151	.82910	.17093	4.851
2047	722	1158	307	167	.82424	.17579	4.689
2150	726	1241	325	183	.81957	.18046	4.542
2252	729	1319	348	204	.81516	.18487	4.409
2355	734	1395	372	226	.81105	.18898	4.292
2457	738	1476	391	243	.80714	.19288	4.185
2559	741	1554	415	264	.80339	.19664	4.086

TABLE XVI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	97	5	52	0	.98971	.01123	88.128
204	181	23	88	0	.97132	.02910	33.380
307	264	43	132	0	.96038	.03988	24.082
409	332	77	167	0	.94775	.05244	18.073
511	390	121	191	0	.93530	.06485	14.423
614	439	175	213	0	.92327	.07684	12.015
716	492	224	243	0	.91456	.08554	10.692
819	537	282	269	0	.90570	.09439	9.595
921	574	347	283	0	.89712	.10296	8.714
1023	613	410	299	0	.88993	.11014	8.080
1126	648	478	313	0	.88300	.11706	7.543
1228	673	555	321	0	.87579	.12426	7.048
1331	695	636	329	0	.86892	.13113	6.627
1433	725	708	341	0	.86351	.13654	6.324
1535	747	788	354	0	.85787	.14218	6.034
1638	764	874	365	0	.85223	.14781	5.766
1740	784	956	373	0	.84729	.15275	5.547
1843	796	1047	380	0	.84210	.15794	5.332
1945	817	1128	389	0	.83790	.16214	5.168
2047	830	1217	396	0	.83347	.16657	5.004
2150	843	1307	401	0	.82926	.17077	4.856
2252	854	1398	405	0	.82523	.17480	4.721
2355	866	1489	412	0	.82143	.17860	4.599
2457	873	1584	416	0	.81762	.18241	4.482
2559	884	1675	421	0	.81419	.18583	4.381

TABLE XVII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	91	11	0	0	.96498	.03596	26.836
204	173	31	0	0	.94819	.05223	18.156
307	249	58	0	0	.93447	.06579	14.204
409	319	90	4	0	.92370	.07649	12.077
511	383	128	8	0	.91406	.08608	10.618
614	445	169	17	0	.90595	.09417	9.621
716	494	222	23	0	.89680	.10330	8.682
819	535	284	29	0	.88764	.11245	7.894
921	576	345	35	0	.87989	.12018	7.321
1023	612	411	42	0	.87250	.12757	6.839
1126	643	483	54	0	.86529	.13477	6.420
1228	673	555	65	0	.85883	.14122	6.081
1331	706	625	82	0	.85325	.14680	5.812
1433	727	703	98	3	.84716	.15289	5.541
1535	739	786	111	10	.84082	.15922	5.281
1638	752	872	125	14	.83501	.16503	5.060
1740	767	956	135	17	.82984	.17020	4.876
1843	783	1037	151	23	.82505	.17498	4.715
1945	792	1121	165	32	.82019	.17985	4.560
2047	799	1207	177	41	.81550	.18453	4.419
2150	804	1295	190	51	.81094	.18909	4.289
2252	807	1387	197	58	.80657	.19346	4.169
2355	814	1478	208	63	.80261	.19741	4.066
2457	817	1566	222	74	.79870	.20132	3.967
2559	825	1652	236	82	.79523	.20480	3.883

TABLE XVIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	91	11	50	0	.96498	.03596	26.836
204	173	31	86	0	.94819	.05223	18.156
307	249	58	126	0	.93447	.06579	14.204
409	319	90	164	0	.92370	.07649	12.077
511	383	128	192	0	.91406	.08608	10.618
614	445	169	222	0	.90595	.09417	9.621
716	494	222	254	0	.89680	.10330	8.682
819	535	284	271	0	.88764	.11245	7.894
921	576	345	286	0	.87989	.12018	7.321
1023	612	411	310	0	.87250	.12757	6.839
1126	643	483	327	0	.86529	.13477	6.420
1228	673	555	343	0	.85883	.14122	6.081
1331	706	625	363	0	.85325	.14680	5.812
1433	730	703	378	0	.84743	.15261	5.553
1535	749	786	386	0	.84169	.15835	5.315
1638	765	873	397	0	.83611	.16393	5.100
1740	781	959	406	0	.83100	.16904	4.916
1843	800	1043	414	0	.82641	.17363	4.760
1945	817	1128	423	0	.82205	.17798	4.619
2047	830	1217	426	0	.81772	.18231	4.485
2150	841	1309	432	0	.81350	.18653	4.361
2252	846	1406	434	0	.80922	.19081	4.241
2355	855	1500	439	0	.80536	.19467	4.137
2457	866	1591	445	0	.80183	.19820	4.046
2559	878	1681	453	0	.79853	.20150	3.963

TABLE XIX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	0	0	.99871	.00224	446.809
204	195	9	0	0	.99040	.01002	98.862
307	293	14	0	0	.98795	.01231	80.280
409	376	26	0	7	.98061	.01957	50.099
511	446	44	0	21	.97108	.02906	33.415
614	496	66	0	52	.95834	.04178	22.936
716	535	94	0	87	.94592	.05418	17.459
819	562	133	0	124	.93336	.06673	13.987
921	583	177	0	161	.92182	.07825	11.780
1023	599	214	0	210	.91112	.08895	10.243
1126	617	260	0	249	.90185	.09821	9.182
1228	632	303	0	293	.89334	.10672	8.371
1331	648	352	0	331	.88568	.11437	7.744
1433	663	397	0	373	.87873	.12131	7.243
1535	673	452	0	410	.87195	.12809	6.807
1638	680	506	0	452	.86541	.13463	6.428
1740	688	560	0	492	.85950	.14054	6.116
1843	695	610	0	538	.85388	.14616	5.842
1945	704	662	0	579	.84882	.15122	5.613
2047	710	715	0	622	.84390	.15614	5.405
2150	718	770	0	662	.83934	.16069	5.223
2252	724	819	0	709	.83498	.16505	5.059
2355	729	874	0	752	.83076	.16927	4.908
2457	732	934	0	791	.82671	.17332	4.770
2559	736	989	0	834	.82291	.17712	4.646

TABLE XX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	0	0	.99639	.00455	218.807
204	198	6	0	0	.99136	.00906	109.395
307	291	16	0	0	.98583	.01443	68.338
409	373	27	0	9	.97816	.02203	44.398
511	439	43	0	29	.96739	.03276	29.531
614	499	70	0	45	.95735	.04277	22.384
716	539	102	0	75	.94520	.05490	17.216
819	570	140	0	109	.93332	.06677	13.978
921	608	168	0	145	.92441	.07567	12.216
1023	626	215	0	182	.91392	.08615	10.609
1126	645	266	0	215	.90472	.09534	9.489
1228	657	311	0	260	.89585	.10420	8.597
1331	669	350	0	312	.88777	.11228	7.907
1433	686	390	0	357	.88098	.11906	7.399
1535	698	439	0	398	.87436	.12568	6.957
1638	711	492	0	435	.86827	.13177	6.589
1740	724	537	0	479	.86271	.13733	6.282
1843	735	588	0	520	.85734	.14269	6.008
1945	751	639	0	555	.85271	.14732	5.788
2047	756	693	0	598	.84770	.15233	5.565
2150	763	750	0	637	.84307	.15696	5.371
2252	769	808	0	675	.83868	.16134	5.198
2355	773	873	0	709	.83439	.16563	5.038
2457	775	940	0	742	.83027	.16975	4.891
2559	778	999	0	782	.82641	.17362	4.760

TABLE XXI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	0	0	.99605	.00489	203.514
204	192	12	0	0	.98539	.01503	65.552
307	284	23	0	0	.97949	.02077	47.153
409	365	43	10	1	.97145	.02873	33.808
511	443	66	17	2	.96472	.03543	27.231
614	512	95	38	7	.95706	.04306	22.227
716	572	130	65	14	.94921	.05089	18.652
819	625	170	96	24	.94117	.05892	15.974
921	671	215	130	35	.93336	.06672	13.990
1023	703	265	182	55	.92468	.07539	12.266
1126	731	324	228	71	.91645	.08361	10.961
1228	751	387	274	90	.90831	.09175	9.900
1331	768	447	332	116	.90059	.09946	9.055
1433	780	507	396	146	.89322	.10683	8.361
1535	789	575	453	171	.88622	.11383	7.786
1638	795	635	530	208	.87948	.12056	7.295
1740	803	704	583	233	.87346	.12658	6.900
1843	811	767	651	265	.86781	.13223	6.563
1945	816	827	728	302	.86240	.13764	6.266
2047	820	889	802	338	.85726	.14277	6.005
2150	823	953	875	374	.85234	.14769	5.771
2252	828	1022	934	402	.84785	.15217	5.572
2355	833	1091	996	431	.84357	.15646	5.392
2457	833	1156	1070	468	.83930	.16072	5.222
2559	838	1220	1138	501	.83549	.16454	5.078

TABLE XXII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	36	0	.99896	.00198	504.087
204	194	10	63	0	.98962	.01079	91.677
307	286	21	88	0	.98361	.01665	59.087
409	371	38	111	0	.97708	.02311	42.274
511	453	58	125	0	.97147	.02868	33.875
614	532	82	143	0	.96612	.03400	28.415
716	596	115	164	5	.95895	.04115	23.302
819	650	153	197	16	.95091	.04918	19.336
921	702	194	229	25	.94384	.05623	16.785
1023	736	240	288	47	.93525	.06481	14.430
1126	768	292	343	66	.92739	.07267	12.761
1228	791	346	402	91	.91946	.08059	11.409
1331	815	397	469	119	.91236	.08769	10.404
1433	829	443	561	161	.90506	.09498	9.529
1535	840	504	627	191	.89813	.10191	8.813
1638	846	554	725	238	.89130	.10874	8.197
1740	855	609	806	276	.88526	.11478	7.713
1843	858	663	899	322	.87918	.12085	7.275
1945	863	722	977	360	.87370	.12634	6.916
2047	866	772	1076	409	.86844	.13160	6.599
2150	869	825	1170	456	.86344	.13659	6.322
2252	873	884	1249	495	.85884	.14119	6.083
2355	879	943	1329	533	.85455	.14548	5.874
2457	881	1001	1415	575	.85033	.14970	5.680
2559	883	1058	1501	618	.84632	.15371	5.506

TABLE XXIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	42	0	.97984	.02111	46.422
204	195	9	56	0	.97355	.02687	36.237
307	284	23	85	0	.96611	.03415	28.294
409	367	42	106	0	.95938	.04081	23.510
511	453	58	128	0	.95556	.04459	21.430
614	518	91	149	5	.94705	.05306	17.847
716	580	127	166	9	.93984	.06026	15.596
819	626	166	215	27	.93074	.06934	13.423
921	677	200	265	44	.92388	.07620	12.125
1023	713	247	313	63	.91591	.08416	10.883
1126	741	296	376	89	.90779	.09227	9.838
1228	768	351	428	109	.90052	.09954	9.047
1331	786	402	506	143	.89300	.10705	8.342
1433	804	459	568	170	.88625	.11380	7.788
1535	817	504	663	214	.87966	.12038	7.307
1638	830	558	742	250	.87352	.12652	6.904
1740	839	610	826	291	.86762	.13242	6.552
1843	852	666	901	325	.86236	.13768	6.264
1945	860	719	986	366	.85717	.14286	6.000
2047	864	777	1069	406	.85208	.14796	5.759
2150	868	834	1155	448	.84724	.15279	5.545
2252	873	888	1243	491	.84278	.15725	5.359
2355	876	940	1339	539	.83841	.16161	5.188
2457	879	1002	1414	576	.83432	.16570	5.035
2559	883	1057	1501	619	.83048	.16954	4.898

TABLE XXIV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	0	0	.99605	.00489	203.514
204	192	12	0	0	.98539	.01503	65.552
307	284	23	0	0	.97949	.02077	47.153
409	366	43	9	0	.97183	.02836	34.273
511	445	66	14	0	.96542	.03473	27.800
614	518	96	30	0	.95878	.04134	23.194
716	576	140	45	0	.95044	.04966	19.139
819	633	186	61	0	.94310	.05699	16.549
921	680	241	77	0	.93541	.06467	14.465
1023	724	299	93	0	.92831	.07176	12.936
1126	758	368	112	0	.92077	.07929	11.613
1228	785	443	131	0	.91337	.08669	10.536
1331	806	516	154	9	.90601	.09404	9.635
1433	815	593	172	25	.89832	.10173	8.830
1535	820	677	188	38	.89093	.10911	8.165
1638	827	750	217	61	.88424	.11580	7.636
1740	829	834	234	77	.87774	.12229	7.177
1843	833	913	255	97	.87179	.12825	6.798
1945	833	995	275	117	.86602	.13401	6.462
2047	839	1065	303	143	.86099	.13904	6.192
2150	841	1150	320	159	.85598	.14405	5.942
2252	847	1235	335	170	.85153	.14850	5.734
2355	851	1324	349	180	.84718	.15285	5.542
2457	851	1410	365	196	.84289	.15713	5.364
2559	856	1491	383	212	.83906	.16097	5.213

TABLE XXV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	0	0	.99600	.00494	201.512
204	199	5	0	0	.99340	.00702	141.522
307	295	12	0	0	.98910	.01116	88.623
409	376	33	11	0	.98088	.01931	50.805
511	445	66	22	0	.97098	.02916	33.295
614	511	103	42	0	.96233	.03779	25.469
716	570	140	79	6	.95417	.04593	20.773
819	610	188	122	21	.94380	.05629	16.768
921	625	238	163	58	.93115	.06893	13.509
1023	635	284	211	104	.91951	.08056	11.414
1126	649	335	250	142	.90963	.09043	10.059
1228	657	393	287	178	.90025	.09981	9.020
1331	665	453	324	213	.89172	.10833	8.231
1433	676	502	368	255	.88434	.11571	7.643
1535	683	563	406	289	.87725	.12280	7.144
1638	697	619	447	322	.87122	.12883	6.763
1740	705	673	491	362	.86525	.13478	6.420
1843	709	745	519	389	.85938	.14065	6.110
1945	719	803	559	423	.85435	.14568	5.865
2047	724	873	589	450	.84933	.15070	5.636
2150	726	934	629	490	.84440	.15563	5.426
2252	729	998	664	525	.83984	.16018	5.243
2355	731	1061	702	563	.83545	.16458	5.076
2457	733	1128	735	596	.83132	.16871	4.928
2559	734	1199	765	626	.82735	.17267	4.792

TABLE XXVI

NTERM	NDICT	NGH	TCD	NCD	P1	P2	P1/P2
102	101	1	0	0	.99871	.00224	446.809
204	195	9	0	0	.99040	.01002	98.862
307	293	14	0	0	.98795	.01231	80.280
409	383	26	7	0	.98352	.01667	58.997
511	466	45	23	0	.97801	.02214	44.179
614	543	71	54	0	.97196	.02816	34.518
716	604	103	98	9	.96394	.03616	26.658
819	637	152	137	30	.95213	.04796	19.852
921	658	206	171	57	.94032	.05975	15.737
1023	681	248	222	94	.93031	.06976	13.336
1126	691	307	260	128	.91981	.08025	11.461
1228	700	370	292	158	.91043	.08963	10.158
1331	708	428	333	195	.90180	.09826	9.178
1433	719	481	374	233	.89431	.10573	8.458
1535	724	545	409	266	.88697	.11308	7.844
1638	733	602	451	303	.88046	.11958	7.363
1740	739	672	479	329	.87428	.12575	6.952
1843	742	737	517	364	.86828	.13176	6.590
1945	749	795	557	401	.86300	.13703	6.298
2047	751	856	596	440	.85774	.14229	6.028
2150	752	923	631	475	.85270	.14733	5.788
2252	755	983	670	514	.84810	.15193	5.582
2355	758	1037	716	560	.84371	.15632	5.397
2457	759	1100	754	598	.83949	.16054	5.229
2559	760	1159	796	640	.83548	.16454	5.078

TABLE XXVII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	53	0	.99871	.00224	446.809
204	195	9	96	0	.99040	.01002	98.862
307	293	14	149	0	.98795	.01231	80.280
409	383	26	189	0	.98352	.01667	58.997
511	466	45	237	0	.97801	.02214	44.179
614	543	71	269	0	.97196	.02816	34.518
716	611	105	314	0	.96536	.03474	27.785
819	663	156	339	0	.95686	.04322	22.139
921	707	214	356	0	.94856	.05151	18.414
1023	755	268	378	0	.94188	.05818	16.188
1126	787	339	396	0	.93395	.06611	14.128
1228	812	416	409	0	.92619	.07387	12.539
1331	836	495	418	0	.91900	.08105	11.339
1433	857	576	428	0	.91229	.08775	10.396
1535	874	661	438	0	.90580	.09424	9.612
1638	891	747	446	0	.89977	.10027	8.974
1740	902	838	453	0	.89382	.10622	8.415
1843	915	928	459	0	.88836	.11167	7.955
1945	924	1021	465	0	.88307	.11696	7.550
2047	932	1115	471	0	.87806	.12197	7.199
2150	941	1209	477	0	.87336	.12667	6.895
2252	945	1306	480	1	.86870	.13133	6.615
2355	947	1406	483	2	.86415	.13588	6.360
2457	950	1505	486	2	.85992	.14010	6.138
2559	952	1605	487	2	.85587	.14416	5.937

TABLE XXVIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	79	23	43	0	.69574	.30520	2.280
204	149	55	74	0	.69417	.30625	2.267
307	224	83	113	0	.69657	.30369	2.294
409	290	119	142	0	.69429	.30590	2.270
511	345	166	169	0	.68931	.31083	2.218
614	401	213	195	0	.68555	.31457	2.179
716	450	266	227	0	.68111	.31899	2.135
819	507	312	263	0	.67873	.32135	2.112
921	545	376	277	0	.67388	.32619	2.066
1023	582	441	295	0	.66957	.33050	2.026
1126	614	512	306	0	.66504	.33502	1.985
1228	651	577	331	0	.66162	.33843	1.955
1331	695	636	361	0	.65919	.34086	1.934
1433	735	698	381	0	.65668	.34336	1.912
1535	756	779	386	0	.65275	.34729	1.880
1638	780	858	400	0	.64933	.35071	1.851
1740	809	931	412	0	.64658	.35346	1.829
1843	822	1021	421	0	.64289	.35715	1.800
1945	842	1103	430	0	.63993	.36010	1.777
2047	865	1182	446	0	.63735	.36268	1.757
2150	882	1268	450	0	.63454	.36549	1.736
2252	893	1359	457	0	.63160	.36843	1.714
2355	903	1452	462	0	.62873	.37129	1.693
2457	915	1542	470	0	.62615	.37388	1.675
2559	925	1634	477	0	.62360	.37643	1.657

TABLE XXIX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	57	0	.99605	.00489	203.514
204	192	12	99	0	.98539	.01503	65.552
307	284	23	152	0	.97949	.02077	47.153
409	366	43	193	0	.97183	.02836	34.273
511	445	66	229	0	.96542	.03473	27.800
614	518	96	259	0	.95878	.04134	23.194
716	576	140	293	0	.95044	.04966	19.139
819	633	186	323	0	.94310	.05699	16.549
921	680	241	336	0	.93541	.06467	14.465
1023	724	299	354	0	.92831	.07176	12.936
1126	758	368	373	0	.92077	.07929	11.613
1228	785	443	391	0	.91337	.08669	10.536
1331	815	516	410	0	.90691	.09314	9.737
1433	839	594	417	0	.90058	.09946	9.055
1535	855	680	424	0	.89411	.10593	8.440
1638	869	766	437	3	.88795	.11209	7.922
1740	878	857	444	5	.88195	.11809	7.468
1843	893	943	457	7	.87672	.12332	7.109
1945	899	1035	466	11	.87131	.12872	6.769
2047	907	1125	474	15	.86637	.13366	6.482
2150	911	1222	477	17	.86144	.13859	6.216
2252	918	1315	483	19	.85701	.14302	5.992
2355	920	1413	488	22	.85252	.14751	5.780
2457	927	1506	494	24	.84856	.15147	5.602
2559	929	1600	500	30	.84455	.15547	5.432

TABLE XXX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	54	0	.99600	.00494	201.512
204	199	5	99	0	.99340	.00702	141.522
307	295	12	148	0	.98910	.01116	88.623
409	376	33	180	0	.98088	.01931	50.805
511	445	66	220	0	.97098	.02916	33.295
614	511	103	247	0	.96233	.03779	25.469
716	575	141	280	0	.95516	.04494	21.252
819	629	190	316	0	.94717	.05292	17.899
921	671	250	328	0	.93870	.06137	15.296
1023	719	304	350	0	.93218	.06789	13.731
1126	754	372	372	0	.92472	.07534	12.273
1228	778	450	383	0	.91693	.08312	11.031
1331	808	523	393	0	.91044	.08961	10.160
1433	832	601	404	0	.90408	.09597	9.421
1535	850	685	411	0	.89776	.10228	8.777
1638	867	771	423	0	.89180	.10824	8.239
1740	885	855	435	0	.88643	.11361	7.802
1843	893	950	438	0	.88068	.11936	7.379
1945	903	1042	446	0	.87551	.12452	7.031
2047	912	1135	454	0	.87060	.12943	6.727
2150	921	1229	460	0	.86595	.13408	6.458
2252	927	1325	463	0	.86144	.13859	6.216
2355	932	1423	466	0	.85708	.14294	5.996
2457	936	1521	467	0	.85295	.14708	5.799
2559	940	1619	470	0	.84902	.15101	5.622

TABLE XXXI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	99	3	48	0	.99331	.00763	130.125
204	191	13	83	0	.98339	.01703	57.748
307	283	24	122	0	.97728	.02298	42.524
409	364	45	162	0	.96950	.03068	31.595
511	436	75	198	0	.96091	.03923	24.493
614	503	111	239	0	.95273	.04739	20.103
716	565	151	270	0	.94534	.05476	17.263
819	620	199	302	0	.93778	.06231	15.051
921	662	259	328	0	.92945	.07063	13.159
1023	700	323	338	0	.92167	.07839	11.757
1126	738	388	358	0	.91469	.08537	10.714
1228	766	462	376	0	.90748	.09257	9.803
1331	793	538	387	0	.90079	.09926	9.075
1433	822	611	405	0	.89498	.10507	8.518
1535	851	684	414	0	.88967	.11037	8.060
1638	871	767	425	0	.88400	.11605	7.618
1740	889	851	435	0	.87867	.12136	7.240
1843	905	938	446	0	.87354	.12649	6.906
1945	918	1022	458	5	.86861	.13142	6.609
2047	925	1111	468	11	.86362	.13641	6.331
2150	935	1203	475	12	.85907	.14096	6.094
2252	942	1295	484	15	.85465	.14538	5.879
2355	946	1388	492	21	.85028	.14975	5.678
2457	949	1484	497	24	.84613	.15390	5.498
2559	953	1580	503	26	.84223	.15780	5.337

TABLE XXXII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	96	6	48	0	.98291	.01804	54.492
204	185	19	86	0	.97087	.02955	32.857
307	277	30	130	0	.96589	.03437	28.104
409	357	52	172	0	.95816	.04202	22.801
511	429	82	207	0	.94993	.05022	18.916
614	503	111	240	0	.94396	.05616	16.808
716	564	152	273	0	.93648	.06362	14.721
819	618	201	309	0	.92883	.07126	13.035
921	659	262	321	0	.92046	.07962	11.561
1023	705	318	338	0	.91390	.08617	10.606
1126	740	386	356	0	.90665	.09341	9.706
1228	775	453	376	0	.90027	.09978	9.022
1331	807	524	394	0	.89415	.10590	8.444
1433	832	601	404	0	.88804	.11200	7.929
1535	847	688	411	0	.88160	.11844	7.443
1638	867	771	426	0	.87601	.12403	7.063
1740	888	852	443	0	.87097	.12907	6.748
1843	898	945	448	0	.86547	.13456	6.432
1945	911	1031	459	3	.86060	.13943	6.172
2047	922	1117	469	8	.85591	.14412	5.939
2150	928	1206	481	16	.85116	.14887	5.718
2252	932	1300	487	20	.84663	.15340	5.519
2355	937	1397	489	21	.84235	.15768	5.342
2457	945	1489	498	23	.83849	.16154	5.191
2559	950	1582	503	27	.83468	.16535	5.048

TABLE XXXIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	0	0	.99605	.00489	203.514
204	194	10	0	0	.98730	.01311	75.285
307	289	18	0	0	.98292	.01734	56.692
409	373	35	10	1	.97592	.02426	40.222
511	451	57	19	3	.96906	.03108	31.178
614	521	84	42	9	.96153	.03859	24.919
716	583	118	69	15	.95401	.04609	20.701
819	636	155	106	28	.94592	.05417	17.463
921	683	198	143	40	.93819	.06189	15.160
1023	712	246	204	65	.92902	.07104	13.077
1126	741	300	256	85	.92085	.07921	11.626
1228	765	358	304	105	.91311	.08695	10.502
1331	781	414	371	136	.90524	.09481	9.548
1433	796	466	444	171	.89809	.10195	8.809
1535	805	530	508	200	.89105	.10899	8.175
1638	813	583	595	242	.88443	.11561	7.650
1740	821	645	662	274	.87837	.12166	7.220
1843	824	707	739	312	.87234	.12769	6.832
1945	829	764	822	352	.86691	.13313	6.512
2047	830	823	906	394	.86156	.13847	6.222
2150	833	883	987	434	.85661	.14342	5.973
2252	838	947	1056	467	.85210	.14793	5.760
2355	843	1010	1129	502	.84780	.15223	5.569
2457	843	1072	1209	542	.84351	.15652	5.389
2559	847	1133	1285	579	.83962	.16040	5.235

TABLE XXXIV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	36	0	.99896	.00198	504.087
204	197	7	63	0	.99246	.00796	124.681
307	291	16	88	0	.98733	.01293	76.363
409	378	31	111	0	.98144	.01875	52.340
511	460	51	126	0	.97569	.02446	39.891
614	540	73	146	1	.97049	.02963	32.757
716	606	104	168	6	.96365	.03645	26.439
819	663	138	207	18	.95608	.04401	21.724
921	710	178	249	33	.94815	.05192	18.261
1023	739	224	313	60	.93883	.06124	15.331
1126	767	273	377	86	.93041	.06965	13.359
1228	792	325	436	111	.92267	.07738	11.924
1331	811	374	514	146	.91504	.08501	10.764
1433	827	417	608	189	.90790	.09215	9.853
1535	840	476	677	219	.90112	.09892	9.109
1638	844	525	781	269	.89410	.10594	8.440
1740	855	572	876	313	.88819	.11185	7.941
1843	861	618	981	364	.88230	.11774	7.494
1945	867	672	1069	406	.87686	.12318	7.119
2047	870	722	1168	455	.87158	.12846	6.785
2150	873	772	1268	505	.86657	.13346	6.493
2252	877	829	1350	546	.86194	.13809	6.242
2355	882	887	1433	586	.85758	.14244	6.021
2457	884	942	1524	631	.85334	.14668	5.818
2559	887	996	1615	676	.84937	.15066	5.638

TABLE XXXV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	42	0	.98172	.01922	51.071
204	197	7	57	0	.97677	.02365	41.306
307	287	20	87	0	.96972	.03054	31.756
409	371	38	108	0	.96325	.03694	26.079
511	459	52	130	0	.95994	.04020	23.878
614	527	82	152	5	.95208	.04804	19.820
716	593	114	170	9	.94561	.05449	17.352
819	639	151	221	29	.93641	.06368	14.705
921	690	185	271	46	.92945	.07063	13.159
1023	724	232	323	67	.92113	.07893	11.670
1126	752	279	389	95	.91295	.08711	10.480
1228	778	333	443	117	.90551	.09454	9.578
1331	797	384	518	150	.89804	.10201	8.803
1433	816	439	582	178	.89134	.10871	8.199
1535	828	484	678	223	.88462	.11543	7.664
1638	840	535	764	263	.87836	.12168	7.219
1740	847	587	851	306	.87228	.12776	6.827
1843	858	643	927	342	.86684	.13319	6.508
1945	864	695	1017	386	.86150	.13854	6.219
2047	868	752	1102	427	.85638	.14366	5.961
2150	870	809	1191	471	.85140	.14863	5.728
2252	874	862	1282	516	.84686	.15317	5.529
2355	877	909	1388	569	.84247	.15755	5.347
2457	881	968	1467	608	.83841	.16162	5.188
2559	884	1023	1555	652	.83450	.16552	5.042

TABLE XXXVI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	1	0	.99871	.00224	446.809
204	198	6	6	0	.99339	.00703	141.305
307	300	7	11	0	.99293	.00733	135.534
409	396	13	27	0	.99082	.00937	105.747
511	486	25	52	0	.98729	.01286	76.800
614	568	46	90	0	.98227	.01785	55.044
716	627	73	152	16	.97363	.02647	36.781
819	663	114	213	42	.96219	.03790	25.387
921	688	162	269	71	.95083	.04925	19.307
1023	703	204	337	116	.93959	.06048	15.537
1126	715	258	403	153	.92920	.07086	13.114
1228	724	315	456	189	.91967	.08033	11.449
1331	733	367	523	231	.91109	.08896	10.241
1433	747	413	590	273	.90380	.09625	9.390
1535	751	472	646	312	.89629	.10375	8.639
1638	753	521	713	364	.88915	.11089	8.019
1740	755	579	777	406	.88262	.11742	7.517
1843	757	635	835	451	.87649	.12355	7.094
1945	761	684	901	500	.87096	.12908	6.748
2047	763	733	969	551	.86565	.13438	6.442
2150	764	795	1028	591	.86056	.13947	6.170
2252	767	847	1097	638	.85592	.14411	5.939
2355	770	893	1164	692	.85148	.14855	5.732
2457	771	948	1232	738	.84722	.15280	5.545
2559	772	998	1301	789	.84318	.15684	5.376

TABLE XXXVII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	57	0	.99871	.00224	446.809
204	197	7	103	0	.99254	.00787	126.053
307	296	11	163	0	.99051	.00975	101.605
409	388	21	215	0	.98686	.01333	74.026
511	473	38	277	0	.98188	.01826	53.764
614	552	62	325	0	.97622	.02390	40.852
716	623	93	377	0	.97016	.02994	32.408
819	682	137	422	0	.96284	.03724	25.854
921	730	191	465	0	.95507	.04501	21.220
1023	780	243	507	0	.94857	.05149	18.421
1126	816	310	557	0	.94104	.05902	15.943
1228	844	384	593	0	.93353	.06653	14.033
1331	872	459	631	0	.92667	.07338	12.628
1433	899	534	681	0	.92043	.07962	11.560
1535	920	615	717	0	.91421	.08583	10.651
1638	940	696	752	2	.90835	.09169	9.907
1740	955	782	789	3	.90262	.09742	9.265
1843	965	871	816	7	.89690	.10314	8.696
1945	973	960	855	12	.89149	.10855	8.213
2047	982	1050	894	15	.88648	.11355	7.807
2150	985	1142	934	23	.88138	.11865	7.428
2252	990	1235	975	27	.87673	.12330	7.111
2355	994	1328	1011	33	.87224	.12779	6.826
2457	994	1421	1051	42	.86783	.13220	6.564
2559	995	1516	1085	48	.86368	.13634	6.335

TABLE XXXVIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	50	0	.99605	.00489	203.514
204	195	9	96	0	.98822	.01220	81.019
307	289	18	144	0	.98335	.01691	58.159
409	374	35	190	0	.97671	.02347	41.610
511	455	56	229	0	.97083	.02932	33.117
614	529	85	261	0	.96432	.03580	26.938
716	589	127	296	0	.95628	.04382	21.825
819	649	170	322	0	.94939	.05070	18.727
921	698	223	339	0	.94190	.05817	16.191
1023	745	278	364	0	.93511	.06495	14.396
1126	786	340	388	0	.92834	.07172	12.944
1228	813	415	403	0	.92085	.07920	11.626
1331	845	486	417	0	.91451	.08554	10.691
1433	871	562	432	0	.90830	.09175	9.900
1535	890	645	441	0	.90201	.09804	9.201
1638	905	725	458	8	.89585	.10419	8.599
1740	913	815	466	12	.88971	.11033	8.064
1843	920	898	482	25	.88387	.11616	7.609
1945	926	986	495	33	.87842	.12161	7.223
2047	931	1069	511	47	.87325	.12678	6.888
2150	935	1162	519	53	.86828	.13175	6.591
2252	939	1254	528	59	.86365	.13638	6.333
2355	942	1345	538	68	.85918	.14085	6.100
2457	947	1436	547	74	.85508	.14495	5.899
2559	949	1526	558	84	.85105	.14898	5.713

TABLE XII.

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	0	0	.99871	.00224	446.809
204	201	3	0	0	.99674	.00368	271.048
307	298	9	0	0	.99282	.00745	133.341
409	387	22	11	0	.98778	.01241	79.583
511	463	48	24	0	.97992	.02023	48.439
614	535	79	47	0	.97261	.02751	35.358
716	604	105	85	7	.96631	.03379	28.596
819	639	141	131	39	.95490	.04519	21.130
921	656	187	171	78	.94238	.05770	16.334
1023	663	233	220	127	.93016	.06991	13.305
1126	679	276	266	171	.92038	.07968	11.551
1228	690	323	312	215	.91121	.08884	10.257
1331	702	374	356	255	.90297	.09708	9.301
1433	715	421	403	297	.89567	.10438	8.581
1535	727	473	445	335	.88891	.11113	7.999
1638	731	525	493	382	.88200	.11804	7.472
1740	734	571	547	435	.87559	.12444	7.036
1843	739	633	583	471	.86972	.13032	6.674
1945	743	684	630	518	.86423	.13580	6.364
2047	745	741	673	561	.85897	.14106	6.089
2150	747	799	716	604	.85398	.14605	5.847
2252	752	851	761	649	.84948	.15055	5.642
2355	754	906	807	695	.84503	.15500	5.452
2457	756	967	846	734	.84085	.15917	5.283
2559	757	1031	883	771	.83684	.16318	5.128

TABLE XL

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	56	0	.99871	.00224	446.809
204	200	4	98	0	.99579	.00463	215.059
307	296	11	149	0	.99133	.00893	111.055
409	380	29	184	0	.98422	.01597	61.642
511	454	57	219	0	.97584	.02430	40.151
614	523	91	245	0	.96787	.03225	30.011
716	591	125	283	0	.96139	.03870	24.839
819	656	163	313	0	.95530	.04478	21.331
921	698	223	326	0	.94671	.05337	17.740
1023	751	272	351	0	.94077	.05930	15.864
1126	792	334	372	0	.93392	.06614	14.121
1228	820	408	380	0	.92649	.07357	12.593
1331	852	479	396	0	.92009	.07996	11.507
1433	876	557	404	0	.91364	.08640	10.574
1535	896	639	411	0	.90741	.09263	9.796
1638	916	721	425	1	.90161	.09843	9.160
1740	932	801	439	7	.89602	.10402	8.614
1843	946	889	445	8	.89062	.10942	8.140
1945	959	976	456	10	.88557	.11446	7.737
2047	966	1068	465	13	.88048	.11955	7.365
2150	973	1159	474	18	.87565	.12438	7.040
2252	977	1251	481	24	.87097	.12906	6.749
2355	980	1345	487	30	.86646	.13357	6.487
2457	985	1440	491	32	.86233	.13770	6.262
2559	988	1534	497	37	.85830	.14172	6.056

TABLE XLI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	99	3	53	0	.99331	.00763	130.125
204	193	11	84	0	.98558	.01484	66.416
307	285	22	129	0	.97933	.02093	46.781
409	366	43	167	0	.97146	.02873	33.818
511	443	68	210	0	.96440	.03574	26.982
614	514	100	239	0	.95716	.04296	22.280
716	577	139	273	0	.94987	.05023	18.912
819	638	181	306	0	.94334	.05675	16.623
921	682	239	326	0	.93522	.06485	14.420
1023	722	301	336	0	.92764	.07243	12.808
1126	762	364	354	0	.92081	.07925	11.619
1228	791	437	370	0	.91365	.08640	10.574
1331	820	511	381	0	.90710	.09295	9.759
1433	850	583	398	0	.90132	.09873	9.130
1535	879	656	413	0	.89596	.10409	8.608
1638	901	737	424	0	.89039	.10965	8.120
1740	921	819	434	0	.88517	.11487	7.706
1843	935	905	446	3	.87986	.12017	7.322
1945	946	988	459	11	.87476	.12527	6.983
2047	955	1074	473	18	.86986	.13018	6.682
2150	961	1164	483	25	.86503	.13500	6.407
2252	971	1251	494	30	.86074	.13928	6.180
2355	977	1338	504	40	.85644	.14359	5.965
2457	983	1430	512	44	.85241	.14762	5.774
2559	987	1523	519	49	.84848	.15154	5.599

TABLE XLII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	97	5	47	0	.98727	.01367	72.228
204	187	17	82	0	.97586	.02455	39.742
307	279	28	131	0	.97056	.02970	32.681
409	360	49	169	0	.96306	.03712	25.942
511	434	77	206	0	.95532	.04483	21.310
614	509	105	235	0	.94945	.05067	18.738
716	571	145	271	0	.94207	.05803	16.236
819	627	192	306	0	.93469	.06539	14.294
921	668	253	316	0	.92623	.07385	12.543
1023	718	305	335	0	.92014	.07993	11.512
1126	758	368	354	0	.91341	.08665	10.542
1228	794	434	365	0	.90707	.09298	9.755
1331	827	504	383	0	.90098	.09907	9.094
1433	854	579	395	0	.89499	.10506	8.519
1535	871	664	404	0	.88865	.11139	7.978
1638	894	744	419	0	.88324	.11680	7.562
1740	916	822	432	2	.87822	.12182	7.209
1843	924	914	438	5	.87254	.12750	6.843
1945	934	998	450	13	.86741	.13262	6.541
2047	949	1078	461	20	.86292	.13711	6.294
2150	960	1162	473	28	.85842	.14161	6.062
2252	966	1254	480	32	.85396	.14607	5.846
2355	972	1349	482	34	.84969	.15034	5.652
2457	979	1436	494	42	.84574	.15429	5.482
2559	983	1527	503	49	.84185	.15818	5.322

TABLE XLIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	1	0	.99871	.00224	446.809
204	198	6	6	0	.99339	.00703	141.305
307	300	7	11	0	.99293	.00733	135.534
409	396	13	27	0	.99082	.00937	105.747
511	486	25	51	0	.98729	.01286	76.800
614	568	46	88	0	.98227	.01785	55.044
716	632	68	147	16	.97472	.02538	38.402
819	670	106	207	43	.96363	.03646	26.432
921	696	152	264	73	.95240	.04767	19.978
1023	708	192	333	123	.94072	.05935	15.851
1126	727	238	399	161	.93116	.06890	13.515
1228	739	292	456	197	.92199	.07806	11.811
1331	744	342	525	245	.91294	.08711	10.480
1433	750	384	598	299	.90490	.09515	9.510
1535	752	440	655	343	.89721	.10283	8.725
1638	754	487	724	397	.89006	.10998	8.093
1740	756	541	789	443	.88352	.11652	7.583
1843	758	593	849	492	.87738	.12265	7.153
1945	762	640	917	543	.87185	.12819	6.801
2047	764	687	986	596	.86653	.13350	6.491
2150	765	747	1047	638	.86144	.13859	6.216
2252	768	799	1116	685	.85679	.14324	5.981
2355	771	841	1184	743	.85235	.14768	5.772
2457	772	890	1255	795	.84809	.15194	5.582
2559	774	936	1327	849	.84409	.15594	5.413

TABLE XLIV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	57	0	.99871	.00224	446.809
204	198	6	104	0	.99339	.00703	141.305
307	300	7	164	0	.99293	.00733	135.534
409	396	13	208	0	.99082	.00937	105.747
511	486	25	258	0	.98729	.01286	76.800
614	568	46	303	0	.98227	.01785	55.044
716	646	70	349	0	.97759	.02251	43.431
819	720	99	394	0	.97282	.02726	35.680
921	786	135	442	0	.96768	.03240	29.867
1023	849	174	481	0	.96279	.03727	25.830
1126	901	224	524	1	.95703	.04303	22.239
1228	934	283	565	11	.94989	.05017	18.934
1331	966	344	616	21	.94328	.05677	16.614
1433	982	415	671	36	.93588	.06417	14.585
1535	992	494	717	49	.92859	.07145	12.996
1638	1001	569	763	68	.92174	.07830	11.772
1740	1003	647	809	90	.91496	.08508	10.755
1843	1003	726	850	114	.90846	.09157	9.921
1945	1003	806	897	136	.90246	.09757	9.249
2047	1003	876	953	168	.89683	.10320	8.690
2150	1003	950	1006	197	.89150	.10853	8.214
2252	1004	1019	1068	229	.88657	.11346	7.814
2355	1004	1093	1123	258	.88182	.11821	7.460
2457	1004	1171	1174	282	.87736	.12267	7.152
2559	1004	1241	1229	314	.87312	.12690	6.880

TABLE XLV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	1	0	.99871	.00224	446.809
204	201	3	4	0	.99633	.00409	243.588
307	299	8	12	0	.99327	.00699	142.129
409	392	17	32	0	.98961	.01058	93.575
511	468	43	50	0	.98148	.01866	52.591
614	545	69	76	0	.97531	.02481	39.315
716	620	96	103	0	.97018	.02992	32.428
819	686	129	135	4	.96412	.03597	26.806
921	745	164	189	12	.95805	.04203	22.796
1023	783	215	237	25	.94986	.05021	18.918
1126	806	275	283	45	.94070	.05936	15.848
1228	827	335	327	66	.93241	.06765	13.783
1331	852	392	387	87	.92524	.07481	12.369
1433	864	464	435	105	.91765	.08240	11.137
1535	878	531	485	126	.91086	.08918	10.214
1638	890	601	539	147	.90440	.09564	9.456
1740	899	668	584	173	.89827	.10177	8.827
1843	905	743	634	195	.89231	.10773	8.283
1945	906	814	686	225	.88647	.11356	7.806
2047	907	892	720	248	.88101	.11902	7.402
2150	910	967	769	273	.87594	.12409	7.059
2252	912	1053	805	287	.87116	.12887	6.760
2355	912	1137	847	306	.86649	.13354	6.489
2457	914	1217	895	326	.86221	.13782	6.256
2559	916	1293	937	350	.85814	.14189	6.048

TABLE XLVI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	27	0	.99871	.00224	446.809
204	201	3	91	0	.99633	.00409	243.588
307	299	8	143	0	.99327	.00699	142.129
409	389	20	189	0	.98833	.01186	83.360
511	460	51	248	0	.97865	.02150	45.519
614	532	82	287	0	.97127	.02885	33.666
716	601	115	324	0	.96494	.03516	27.443
819	666	153	372	0	.95877	.04132	23.205
921	731	190	428	0	.95368	.04640	20.554
1023	782	241	469	0	.94732	.05275	17.960
1126	817	309	498	0	.93964	.06042	15.553
1228	851	377	534	0	.93278	.06727	13.866
1331	883	448	579	0	.92634	.07371	12.567
1433	902	531	612	0	.91938	.08066	11.398
1535	927	607	656	1	.91352	.08652	10.558
1638	941	695	698	2	.90719	.09285	9.770
1740	955	780	726	5	.90140	.09864	9.138
1843	964	869	765	10	.89562	.10442	8.577
1945	977	955	803	13	.89054	.10949	8.133
2047	980	1048	828	19	.88517	.11486	7.706
2150	983	1145	859	22	.88008	.11995	7.337
2252	991	1238	892	23	.87560	.12443	7.037
2355	994	1336	923	25	.87106	.12897	6.754
2457	997	1433	956	27	.86681	.13322	6.507
2559	999	1528	981	32	.86271	.13731	6.283

TABLE XLVII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	35	0	1.00094	0.00000	R
204	203	1	98	0	.99915	.00127	789.303
307	300	7	157	0	.99530	.00497	200.455
409	388	21	198	0	.98965	.01053	93.956
511	471	40	263	0	.98396	.01619	60.792
614	542	72	308	0	.97630	.02382	40.982
716	607	109	346	0	.96904	.03106	31.195
819	671	148	390	0	.96259	.03749	25.673
921	729	192	435	0	.95633	.04375	21.860
1023	769	254	475	0	.94841	.05166	18.359
1126	805	321	568	0	.94087	.05919	15.897
1228	835	393	538	0	.93356	.06650	14.039
1331	861	468	571	2	.92650	.07355	12.598
1433	880	545	610	8	.91954	.08051	11.422
1535	897	628	641	10	.91300	.08704	10.489
1638	916	712	677	10	.90706	.09298	9.756
1740	930	800	711	10	.90127	.09876	9.125
1843	944	887	734	12	.89583	.10420	8.597
1945	957	971	768	17	.89075	.10928	8.151
2047	966	1061	796	20	.88576	.11427	7.751
2150	975	1154	820	21	.88101	.11902	7.402
2252	979	1247	833	26	.87631	.12372	7.083
2355	987	1336	872	32	.87203	.12800	6.813
2457	990	1434	906	33	.86776	.13226	6.561
2559	992	1530	938	37	.86367	.13636	6.334

TABLE XLVIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	3	0	1.00094	0.00000	R
204	203	1	5	0	.99915	.00127	789.303
307	300	7	15	0	.99530	.00497	200.455
409	391	18	29	0	.99083	.00936	105.851
511	477	34	52	0	.98603	.01412	69.852
614	552	62	72	0	.97932	.02080	47.080
716	620	96	85	0	.97262	.02748	35.395
819	693	126	123	0	.96775	.03234	29.925
921	760	161	155	0	.96282	.03725	25.846
1023	803	220	187	0	.95522	.04484	21.302
1126	843	283	214	0	.94809	.05197	18.244
1228	875	351	248	2	.94092	.05913	15.912
1331	903	422	285	6	.93399	.06606	14.139
1433	929	491	331	13	.92760	.07245	12.804
1535	949	570	365	16	.92125	.07879	11.692
1638	967	654	405	17	.91517	.08487	10.784
1740	977	741	440	22	.90903	.09101	9.989
1843	988	827	470	28	.90334	.09670	9.342
1945	993	911	509	41	.89769	.10235	8.771
2047	997	998	539	52	.89234	.10770	8.286
2150	1006	1089	568	55	.88755	.11248	7.890
2252	1008	1179	588	65	.88270	.11733	7.523
2355	1010	1268	631	77	.87807	.12196	7.200
2457	1011	1359	672	87	.87368	.12635	6.915
2559	1011	1455	707	93	.86946	.13057	6.659

TABLE XLIX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	54	0	1.00094	0.00000	R
204	203	1	108	0	.99955	.00087	1154.966
307	298	9	163	0	.99444	.00583	170.718
409	386	23	211	0	.98877	.01142	86.617
511	468	43	265	0	.98268	.01747	56.247
614	547	67	311	0	.97704	.02308	42.336
716	608	108	358	0	.96895	.03115	31.110
819	660	159	397	0	.96042	.03966	24.214
921	717	204	448	0	.95405	.04602	20.729
1023	770	253	505	0	.94798	.05209	18.200
1126	813	313	543	0	.94127	.05879	16.012
1228	846	382	582	0	.93430	.06575	14.210
1331	874	457	636	0	.92743	.07262	12.771
1433	899	531	677	3	.92100	.07905	11.651
1535	913	615	719	7	.91419	.08586	10.648
1638	[illegible]	[illegible]	761	16	.90746	.09258	9.801
1740	[illegible]	[illegible]	805	25	.90175	.09829	9.175
1843	[illegible]	[illegible]	850	31	.89603	.10401	8.615
[illegible]	[illegible]	[illegible]	893	42	.89043	.10960	8.124
[illegible]	[illegible]	[illegible]	934	55	.88531	.11472	7.717
[illegible]	[illegible]	[illegible]	972	67	.88039	.11964	7.358
[illegible]	[illegible]	[illegible]	1009	78	.87574	.12429	7.046
2355	[illegible]	1298	1043	84	.87120	.12883	6.763
2457	977	1385	1083	95	.86699	.13303	6.517
2559	981	1477	1115	101	.86300	.13703	6.298

TABLE L

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	19	0	1.00094	0.00000	R
204	203	1	26	0	.99955	.00087	1154.966
307	298	9	34	0	.99444	.00583	170.718
409	388	21	49	0	.98964	.01054	93.868
511	476	35	64	0	.98542	.01473	66.914
614	560	54	85	0	.98100	.01911	51.324
716	628	88	103	0	.97428	.02582	37.737
819	688	131	127	0	.96707	.03301	29.295
921	749	172	161	0	.96121	.03886	24.734
1023	808	215	200	0	.95585	.04422	21.618
1126	859	267	241	0	.95003	.05003	18.988
1228	895	332	280	1	.94330	.05676	16.620
1331	929	400	332	2	.93693	.06312	14.844
1433	956	470	380	7	.93060	.06945	13.400
1535	969	554	420	12	.92362	.07642	12.086
1638	979	639	463	20	.91689	.08315	11.027
1740	990	718	507	32	.91082	.08922	10.208
1843	997	801	555	45	.90482	.09521	9.503
1945	1004	884	602	57	.89930	.10074	8.927
2047	1007	968	645	72	.89387	.10616	8.420
2150	1010	1051	687	89	.88873	.11130	7.985
2252	1013	1132	730	107	.88393	.11610	7.614
2355	1013	1222	768	120	.87919	.12084	7.276
2457	1014	1304	813	139	.87479	.12523	6.985
2559	1014	1389	849	156	.87057	.12945	6.725

TABLE LI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	94	8	0	0	.97352	.02742	35.503
204	173	31	0	0	.95140	.04902	19.410
307	256	51	0	0	.94202	.05824	16.176
409	325	84	0	0	.93045	.06974	13.342
511	395	116	0	0	.92253	.07762	11.885
614	452	162	0	0	.91272	.08740	10.443
716	501	215	0	0	.90343	.09667	9.346
819	542	277	0	0	.89412	.10596	8.438
921	587	334	0	0	.88695	.11313	7.840
1023	624	399	0	0	.87961	.12046	7.302
1126	653	472	0	1	.87205	.12801	6.812
1228	670	552	0	6	.86410	.13596	6.356
1331	683	637	0	11	.85646	.14359	5.964
1433	708	711	0	14	.85071	.14934	5.697
1535	725	792	0	18	.84476	.15528	5.440
1638	737	874	0	27	.83883	.16121	5.204
1740	752	954	0	34	.83364	.16640	5.010
1843	768	1038	0	37	.82882	.17121	4.841
1945	780	1121	0	44	.82412	.17591	4.685
2047	790	1210	0	47	.81960	.18043	4.542
2150	803	1294	0	53	.81547	.18456	4.418
2252	815	1377	0	60	.81157	.18846	4.306
2355	827	1463	0	65	.80784	.19218	4.203
2457	835	1547	0	75	.80415	.19587	4.106
2559	839	1639	0	81	.80046	.19956	4.011

TABLE LII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	89	13	0	0	.95116	.04978	19.105
204	181	23	0	0	.94529	.05512	17.148
307	255	52	0	0	.93027	.06999	13.292
409	312	97	0	0	.91420	.08599	10.631
511	375	136	0	0	.90432	.09583	9.437
614	426	188	0	0	.89363	.10649	8.391
716	478	238	0	0	.88543	.11467	7.721
819	526	293	0	0	.87770	.12239	7.172
921	564	357	0	0	.86963	.13044	6.667
1023	592	425	0	6	.86131	.13875	6.207
1126	614	502	0	10	.85312	.14694	5.806
1228	637	567	0	24	.84606	.15399	5.494
1331	652	645	0	34	.83881	.16124	5.202
1433	674	711	0	48	.83293	.16711	4.984
1535	697	783	0	55	.82766	.17239	4.801
1638	715	859	0	64	.82234	.17770	4.628
1740	727	940	0	73	.81705	.18299	4.465
1843	739	1022	0	82	.81207	.18796	4.320
1945	748	1106	0	91	.80729	.19274	4.189
2047	755	1187	0	105	.80269	.19735	4.067
2150	765	1272	0	113	.79848	.20155	3.962
2252	774	1358	0	120	.79452	.20551	3.866
2355	784	1444	0	127	.79078	.20925	3.779
2457	787	1531	0	139	.78693	.21310	3.693
2559	789	1618	0	152	.78322	.21680	3.613

TABLE LIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	98	4	0	0	.98771	.01324	74.616
204	190	14	0	0	.97746	.02296	42.578
307	265	42	0	0	.96114	.03912	24.570
409	334	75	0	0	.94868	.05151	18.419
511	398	113	0	0	.93821	.06194	15.148
614	454	160	0	0	.92784	.07228	12.837
716	501	215	0	0	.91777	.08233	11.147
819	547	272	0	0	.90909	.09099	9.991
921	590	331	0	0	.90132	.09875	9.127
1023	623	400	0	0	.89325	.10682	8.362
1126	654	472	0	0	.88578	.11428	7.751
1228	682	546	0	0	.87889	.12116	7.254
1331	700	630	0	1	.87160	.12845	6.785
1433	720	708	0	5	.86522	.13482	6.417
1535	735	786	0	14	.85898	.14106	6.089
1638	753	865	0	20	.85341	.14663	5.820
1740	767	949	0	24	.84801	.15203	5.578
1843	776	1036	0	31	.84261	.15742	5.352
1945	789	1113	0	43	.83789	.16215	5.167
2047	799	1202	0	46	.83327	.16676	4.997
2150	809	1290	0	51	.82889	.17114	4.843
2252	815	1382	0	55	.82459	.17544	4.700
2355	826	1465	0	64	.82074	.17929	4.578
2457	834	1550	0	73	.81699	.18304	4.463
2559	841	1639	0	79	.81337	.18665	4.358

TABLE LIV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	92	10	0	0	.96799	.03295	29.378
204	184	20	0	0	.96029	.04013	23.931
307	260	47	0	0	.94592	.05434	17.406
409	331	78	0	0	.93509	.06510	14.363
511	391	120	0	0	.92359	.07655	12.065
614	455	159	0	0	.91573	.08439	10.852
716	497	219	0	0	.90493	.09517	9.508
819	542	277	0	0	.89632	.10376	8.638
921	581	340	0	0	.88818	.11190	7.937
1023	606	417	0	0	.87918	.12088	7.273
1126	636	490	0	0	.87179	.12827	6.797
1228	670	558	0	0	.86574	.13432	6.445
1331	701	627	0	3	.85988	.14017	6.134
1433	717	709	0	7	.85326	.14678	5.813
1535	731	793	0	11	.84704	.15300	5.536
1638	741	882	0	15	.84095	.15909	5.286
1740	756	964	0	20	.83573	.16430	5.087
1843	768	1045	0	30	.83063	.16941	4.903
1945	775	1130	0	40	.82559	.17444	4.733
2047	783	1214	0	50	.82093	.17910	4.584
2150	792	1299	0	59	.81657	.18346	4.451
2252	799	1384	0	69	.81239	.18764	4.330
2355	812	1460	0	83	.80871	.19131	4.227
2457	819	1547	0	91	.80497	.19505	4.127
2559	823	1626	0	110	.80127	.19875	4.032

TABLE LV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	95	7	0	0	.97801	.02293	42.647
204	180	24	0	0	.96026	.04016	23.913
307	262	45	0	0	.94960	.05066	18.743
409	325	84	0	0	.93519	.06500	14.387
511	384	127	0	0	.92350	.07664	12.049
614	439	175	0	0	.91323	.08689	10.510
716	493	223	0	0	.90499	.09511	9.515
819	538	281	0	0	.89637	.10371	8.643
921	580	341	0	0	.88869	.11138	7.979
1023	620	403	0	0	.88176	.11831	7.453
1126	655	469	0	2	.87495	.12511	6.993
1228	680	535	0	13	.86785	.13221	6.564
1331	702	613	0	16	.86105	.13900	6.195
1433	721	692	0	20	.85471	.14533	5.881
1535	741	765	0	29	.84899	.15106	5.620
1638	755	847	0	36	.84319	.15685	5.376
1740	762	934	0	44	.83736	.16267	5.148
1843	772	1020	0	51	.83210	.16794	4.955
1945	775	1113	0	57	.82679	.17324	4.772
2047	789	1198	0	60	.82249	.17754	4.633
2150	803	1281	0	66	.81841	.18162	4.506
2252	807	1373	0	72	.81405	.18597	4.377
2355	816	1460	0	79	.81016	.18987	4.267
2457	824	1544	0	89	.80646	.19357	4.166
2559	831	1636	0	92	.80289	.19713	4.073

TABLE LVI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	95	7	0	0	.96766	.03328	29.075
204	172	32	0	0	.94303	.05739	16.431
307	249	58	0	0	.93039	.06987	13.315
409	320	89	0	0	.92030	.07988	11.520
511	378	133	0	0	.90902	.09113	9.975
614	446	168	0	0	.90251	.09761	9.246
716	494	222	0	0	.89325	.10685	8.360
819	539	280	0	0	.88486	.11522	7.679
921	572	349	0	0	.87599	.12409	7.059
1023	598	425	0	0	.86730	.13276	6.533
1126	633	493	0	0	.86065	.13941	6.173
1228	655	566	0	7	.85336	.14670	5.817
1331	677	637	0	17	.84673	.15332	5.522
1433	692	718	0	23	.84014	.15990	5.254
1535	710	799	0	26	.83438	.16567	5.036
1638	724	883	0	31	.82869	.17135	4.836
1740	737	962	0	41	.82343	.17662	4.662
1843	750	1044	0	49	.81847	.18157	4.508
1945	763	1130	0	52	.81389	.18614	4.373
2047	775	1214	0	58	.80956	.19048	4.250
2150	785	1297	0	68	.80532	.19471	4.136
2252	795	1383	0	74	.80137	.19866	4.034
2355	800	1467	0	88	.79734	.20269	3.934
2457	804	1552	0	101	.79350	.20652	3.842
2559	806	1646	0	107	.78977	.21026	3.756

TABLE LVII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	96	6	0	0	.98479	.01615	60.975
204	185	19	0	0	.97277	.02765	35.182
307	266	41	0	0	.96056	.03970	24.197
409	340	69	0	0	.95063	.04956	19.183
511	394	117	0	0	.93699	.06316	14.836
614	444	170	0	0	.92513	.07499	12.337
716	489	227	0	0	.91476	.08534	10.719
819	528	291	0	0	.90490	.09519	9.507
921	577	344	0	0	.89814	.10193	8.811
1023	620	403	0	0	.89146	.10861	8.208
1126	652	474	0	0	.88413	.11593	7.626
1228	682	545	0	1	.87749	.12257	7.159
1331	709	617	0	5	.87111	.12894	6.756
1433	727	697	0	9	.86458	.13547	6.382
1535	743	773	0	19	.85843	.14161	6.062
1638	763	848	0	27	.85303	.14701	5.802
1740	773	932	0	35	.84736	.15268	5.550
1843	785	1016	0	42	.84217	.15787	5.335
1945	798	1095	0	52	.83744	.16259	5.150
2047	809	1174	0	64	.83289	.16714	4.983
2150	817	1257	0	76	.82840	.17163	4.827
2252	822	1344	0	86	.82405	.17598	4.683
2355	832	1427	0	96	.82015	.17988	4.560
2457	837	1512	0	108	.81625	.18378	4.442
2559	840	1603	0	116	.81245	.18757	4.331

TABLE LVIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	96	6	0	0	.97966	.02128	46.039
204	182	22	0	0	.96559	.03483	27.726
307	260	47	0	0	.95205	.04821	19.748
409	330	79	0	0	.94028	.05991	15.695
511	384	127	0	0	.92681	.07333	12.636
614	444	170	0	0	.91782	.08230	11.152
746	497	219	0	0	.90930	.09080	10.014
819	542	277	0	0	.90059	.09949	9.052
921	581	340	0	0	.89237	.10771	8.285
1023	621	394	0	8	.88534	.11473	7.717
1126	645	465	0	16	.87709	.12297	7.132
1228	670	534	0	24	.86997	.13009	6.687
1331	694	597	0	40	.86335	.13670	6.316
1433	721	665	0	47	.85771	.14233	6.026
1535	739	737	0	59	.85179	.14825	5.746
1638	758	808	0	72	.84637	.15367	5.508
1740	771	891	0	78	.84097	.15907	5.287
1843	778	971	0	94	.83548	.16456	5.077
1945	789	1047	0	109	.83067	.16937	4.905
2047	794	1132	0	121	.82580	.17423	4.740
2150	803	1214	0	133	.82140	.17863	4.598
2252	814	1290	0	148	.81741	.18262	4.476
2355	819	1369	0	167	.81329	.18674	4.355
2457	822	1455	0	180	.80933	.19070	4.244
2559	825	1544	0	190	.80556	.19446	4.142

TABLE LIX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	93	9	0	0	.97413	.02682	36.323
204	179	25	0	0	.95786	.04256	22.508
307	256	51	0	0	.94446	.05580	16.927
409	337	72	0	0	.93797	.06222	15.075
511	406	105	0	0	.92941	.07074	13.139
614	458	156	0	0	.91831	.08181	11.224
716	507	209	0	0	.90895	.09115	9.972
819	556	263	0	0	.90097	.09911	9.090
921	592	329	0	0	.89231	.10777	8.280
1023	636	387	0	0	.88587	.11420	7.757
1126	664	458	0	4	.87813	.12193	7.202
1228	684	536	0	8	.87044	.12961	6.716
1331	699	614	0	18	.86291	.13714	6.292
1433	712	698	0	23	.85599	.14406	5.942
1535	734	776	0	25	.85043	.14961	5.684
1638	748	858	0	32	.84462	.15542	5.434
1740	758	942	0	40	.83901	.16102	5.211
1843	770	1026	0	47	.83388	.16616	5.019
1945	782	1111	0	52	.82915	.17089	4.852
2047	796	1192	0	59	.82484	.17520	4.708
2150	806	1279	0	65	.82051	.17953	4.570
2252	821	1361	0	70	.81675	.18328	4.456
2355	829	1449	0	77	.81279	.18724	4.341
2457	832	1542	0	83	.80883	.19120	4.230
2559	840	1624	0	95	.80530	.19472	4.136

TABLE LX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	97	5	0	0	.98957	.01138	86.977
204	184	20	0	0	.97466	.02576	37.837
307	260	47	0	0	.95980	.04046	23.723
409	322	87	0	0	.94444	.05574	16.943
511	377	134	0	0	.93128	.06887	13.522
614	428	186	0	0	.91994	.08018	11.474
716	478	238	0	0	.91073	.08936	10.191
819	529	290	0	0	.90315	.09693	9.317
921	575	346	0	0	.89596	.10412	8.605
1023	609	414	0	0	.88810	.11197	7.931
1126	638	488	0	0	.88043	.11963	7.360
1228	666	559	0	3	.87361	.12645	6.909
1331	689	636	0	6	.86687	.13318	6.509
1433	720	704	0	9	.86156	.13848	6.222
1535	738	783	0	14	.85563	.14442	5.925
1638	757	862	0	19	.85016	.14988	5.672
1740	776	936	0	28	.84517	.15487	5.457
1843	791	1019	0	33	.84021	.15983	5.257
1945	799	1106	0	40	.83518	.16486	5.066
2047	810	1190	0	47	.83065	.16938	4.904
2150	818	1274	0	58	.82617	.17386	4.752
2252	829	1359	0	64	.82216	.17787	4.622
2355	836	1453	0	66	.81811	.18191	4.497
2457	838	1548	0	71	.81408	.18595	4.378
2559	843	1639	0	77	.81038	.18964	4.273

TABLE LXI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	95	7	0	0	.96931	.03163	30.644
204	185	19	0	0	.95907	.04134	23.198
307	256	51	0	0	.94169	.05857	16.079
409	325	84	0	0	.93001	.07018	13.251
511	382	129	0	0	.91793	.08222	11.165
614	439	175	0	0	.90832	.09179	9.895
716	485	231	0	0	.89848	.10162	8.842
819	528	291	0	0	.88963	.11046	8.054
921	564	357	0	0	.88112	.11895	7.407
1023	602	421	0	0	.87398	.12608	6.932
1126	631	495	0	0	.86646	.13360	6.485
1228	658	566	0	4	.85968	.14038	6.124
1331	677	647	0	7	.85269	.14737	5.786
1433	689	732	0	12	.84576	.15428	5.482
1535	710	807	0	18	.84020	.15984	5.257
1638	727	888	0	23	.83471	.16533	5.049
1740	740	971	0	29	.82939	.17065	4.860
1843	752	1055	0	36	.82433	.17571	4.692
1945	765	1135	0	45	.81972	.18031	4.546
2047	778	1218	0	51	.81541	.18463	4.417
2150	785	1306	0	59	.81096	.18907	4.289
2252	794	1392	0	66	.80692	.19311	4.178
2355	801	1481	0	73	.80296	.19707	4.074
2457	805	1576	0	76	.79909	.20093	3.977
2559	809	1666	0	84	.79543	.20460	3.888

TABLE LXII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	97	5	0	0	.98966	.01128	87.729
204	181	23	0	0	.97031	.03011	32.226
307	254	53	0	0	.95356	.04670	20.420
409	314	95	0	0	.93757	.06262	14.972
511	371	140	0	0	.92530	.07485	12.363
614	429	185	0	0	.91570	.08442	10.847
716	476	240	0	0	.90594	.09416	9.621
819	516	303	0	0	.89640	.10369	8.645
921	555	366	0	0	.88825	.11183	7.943
1023	584	439	0	0	.87981	.12026	7.316
1126	611	515	0	0	.87203	.12803	6.811
1228	628	594	0	6	.86408	.13597	6.355
1331	652	668	0	11	.85754	.14251	6.017
1433	671	746	0	16	.85122	.14882	5.720
1535	699	820	0	16	.84621	.15383	5.501
1638	721	899	0	18	.84106	.15898	5.290
1740	731	985	0	24	.83548	.16456	5.077
1843	743	1072	0	28	.83037	.16966	4.894
1945	754	1155	0	36	.82560	.17443	4.733
2047	774	1233	0	40	.82168	.17835	4.607
2150	789	1315	0	46	.81766	.18237	4.483
2252	797	1400	0	55	.81353	.18650	4.362
2355	808	1488	0	59	.80973	.19029	4.255
2457	818	1573	0	66	.80613	.19389	4.158
2559	823	1664	0	72	.80248	.19755	4.062

TABLE LXIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	0	0	.99443	.00651	152.652
204	200	4	0	0	.99221	.00821	120.815
307	294	13	0	0	.98692	.01334	74.004
409	384	25	0	0	.98215	.01804	54.452
511	468	43	0	0	.97696	.02318	42.142
614	544	70	0	0	.97069	.02943	32.982
716	607	109	2	0	.96317	.03693	26.078
819	656	163	11	0	.95422	.04587	20.803
921	706	215	26	0	.94688	.05320	17.800
1023	746	277	37	0	.93912	.06094	15.409
1126	776	346	52	4	.93095	.06911	13.470
1228	800	417	69	11	.92309	.07697	11.993
1331	814	500	79	17	.91493	.08512	10.749
1433	828	582	92	23	.90762	.09243	9.820
1535	848	658	108	29	.90144	.09860	9.142
1638	862	742	122	34	.89521	.10483	8.540
1740	865	836	128	39	.88870	.11133	7.982
1843	879	914	145	50	.88336	.11668	7.571
1945	889	994	162	62	.87817	.12187	7.206
2047	893	1084	173	70	.87294	.12709	6.869
2150	898	1176	182	76	.86804	.13199	6.576
2252	909	1258	198	85	.86379	.13624	6.340
2355	916	1347	208	92	.85952	.14051	6.117
2457	921	1433	224	103	.85542	.14460	5.916
2559	922	1527	231	110	.85134	.14869	5.726

TABLE LXIV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	54	0	.99443	.00651	152.652
204	200	4	119	0	.99221	.00821	120.815
307	294	13	178	0	.98692	.01334	74.004
409	384	25	233	0	.98215	.01804	54.452
511	468	43	284	0	.97696	.02318	42.142
614	544	70	332	0	.97069	.02943	32.982
716	607	109	369	0	.96317	.03693	26.078
819	656	163	395	0	.95422	.04587	20.803
921	706	215	427	0	.94688	.05320	17.800
1023	746	277	455	0	.93912	.06094	15.409
1126	778	348	473	0	.93119	.06887	13.522
1228	808	420	492	0	.92400	.07606	12.148
1331	827	504	503	0	.91633	.08372	10.945
1433	845	588	517	0	.90937	.09068	10.029
1535	868	667	532	0	.90344	.09661	9.352
1638	883	755	537	0	.89727	.10277	8.731
1740	889	851	542	0	.89097	.10907	8.169
1843	908	935	555	0	.88596	.11408	7.766
1945	923	1022	565	0	.88108	.11896	7.407
2047	929	1118	568	0	.87595	.12408	7.060
2150	936	1214	572	0	.87115	.12888	6.759
2252	942	1310	573	0	.86661	.13342	6.495
2355	950	1405	577	0	.86238	.13765	6.265
2457	958	1499	585	0	.85842	.14161	6.062
2559	962	1597	587	0	.85446	.14557	5.870

TABLE LXV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	5	0	.99827	.00267	373.910
204	202	2	13	0	.99718	.00323	308.266
307	297	10	13	0	.99221	.00805	123.232
409	392	17	23	0	.98936	.01083	91.356
511	483	28	31	0	.98621	.01394	70.768
614	571	43	46	0	.98278	.01734	56.681
716	644	72	62	0	.97708	.02302	42.453
819	702	117	88	0	.96947	.03062	31.661
921	760	161	118	0	.96313	.03694	26.070
1023	808	210	152	5	.95624	.04383	21.819
1126	838	273	193	15	.94785	.05221	18.156
1228	863	336	238	29	.93992	.06013	15.632
1331	877	412	287	42	.93161	.06845	13.611
1433	899	481	331	53	.92486	.07518	12.302
1535	916	552	373	67	.91828	.08176	11.231
1638	924	633	434	81	.91144	.08860	10.287
1740	928	721	467	91	.90489	.09515	9.510
1843	935	789	520	119	.89895	.10109	8.893
1945	942	858	571	145	.89346	.10657	8.384
2047	945	938	620	164	.88807	.11196	7.932
2150	946	1023	661	181	.88285	.11718	7.534
2252	953	1094	711	205	.87830	.12173	7.215
2355	955	1176	760	224	.87369	.12633	6.916
2457	956	1250	814	251	.86933	.13070	6.651
2559	957	1335	851	267	.86518	.13485	6.416

TABLE LXVI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	100	2	58	0	.99443	.00651	152.652
204	201	3	131	0	.99379	.00663	149.953
307	295	12	192	0	.98841	.01185	83.384
409	386	23	257	0	.98401	.01618	60.819
511	471	40	319	0	.97911	.02103	46.555
614	547	67	385	0	.97278	.02734	35.586
716	611	105	441	0	.96542	.03468	27.840
819	660	159	485	0	.95643	.04365	21.910
921	710	211	534	0	.94906	.05102	18.603
1023	750	273	580	0	.94127	.05880	16.009
1126	783	343	624	0	.93344	.06662	14.011
1228	813	415	667	0	.92622	.07384	12.544
1331	832	499	713	0	.91853	.08152	11.267
1433	850	583	754	0	.91155	.08850	10.300
1535	873	662	788	0	.90559	.09445	9.588
1638	889	749	842	0	.89949	.10055	8.946
1740	895	845	869	0	.89317	.10687	8.358
1843	914	929	908	0	.88814	.11190	7.937
1945	931	1014	942	0	.88338	.11666	7.572
2047	938	1109	979	0	.87830	.12173	7.215
2150	946	1204	1012	0	.87354	.12649	6.906
2252	953	1299	1048	0	.86904	.13099	6.634
2355	961	1394	1085	0	.86480	.13523	6.395
2457	969	1488	1121	0	.86082	.13921	6.184
2559	973	1586	1144	0	.85685	.14317	5.985

TABLE LXVII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	5	0	.99827	.00267	373.910
204	202	2	13	0	.99718	.00323	308.266
307	297	10	13	0	.99221	.00805	123.232
409	393	16	23	0	.98979	.01039	95.219
511	486	25	31	0	.98729	.01286	76.789
614	575	39	46	0	.98407	.01605	61.305
716	650	66	62	0	.97877	.02133	45.895
819	713	106	85	0	.97200	.02809	34.605
921	777	143	113	1	.96656	.03351	28.841
1023	834	183	142	6	.96086	.03921	24.506
1126	875	236	184	15	.95378	.04628	20.608
1228	901	290	229	37	.94589	.05417	17.463
1331	924	353	286	54	.93842	.06163	15.227
1433	945	422	333	66	.93152	.06852	13.594
1535	957	490	379	88	.92446	.07558	12.231
1638	961	569	442	108	.91725	.08279	11.079
1740	964	650	482	126	.91058	.08946	10.179
1843	971	714	538	158	.90459	.09544	9.478
1945	974	780	592	191	.89881	.10123	8.879
2047	975	857	642	215	.89327	.10676	8.367
2150	976	938	688	236	.88801	.11202	7.927
2252	980	1001	746	271	.88327	.11676	7.565
2355	983	1073	800	299	.87869	.12134	7.242
2457	983	1142	857	332	.87425	.12578	6.951
2559	984	1214	905	361	.87007	.12995	6.695

TABLE LXVIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	55	0	.99827	.00267	373.910
204	202	2	124	0	.99718	.00323	308.266
307	297	10	185	0	.99221	.00805	123.232
409	393	16	244	0	.98979	.01039	95.219
511	486	25	304	0	.98729	.01286	76.789
614	575	39	366	0	.98407	.01605	61.305
716	650	66	414	0	.97877	.02133	45.895
819	713	106	460	0	.97200	.02809	34.605
921	778	143	507	0	.96671	.03337	28.972
1023	841	182	548	0	.96183	.03824	25.156
1126	896	230	588	0	.95645	.04361	21.931
1228	945	283	624	0	.95105	.04901	19.406
1331	979	350	672	2	.94463	.05542	17.046
1433	997	426	718	10	.93740	.06265	14.963
1535	1003	501	763	31	.92977	.07027	13.231
1638	1008	584	823	46	.92260	.07744	11.914
1740	1010	669	861	61	.91582	.08422	10.874
1843	1014	734	921	95	.90959	.09045	10.056
1945	1015	804	972	126	.90364	.09640	9.374
2047	1015	884	1020	148	.89801	.10203	8.802
2150	1015	968	1067	167	.89266	.10737	8.314
2252	1017	1036	1121	199	.88779	.11224	7.909
2355	1017	1109	1175	229	.88303	.11700	7.547
2457	1017	1184	1230	256	.87856	.12147	7.233
2559	1017	1263	1274	279	.87432	.12571	6.955

TABLE LXIX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	1	0	.99841	.00253	394.738
204	202	2	4	0	.99719	.00322	309.314
307	296	11	10	0	.99139	.00887	111.763
409	382	27	17	0	.98471	.01548	63.603
511	469	42	25	0	.98049	.01965	49.894
614	551	63	33	0	.97564	.02447	39.863
716	626	90	43	0	.97052	.02958	32.806
819	699	120	65	0	.96573	.03436	28.106
921	769	152	100	0	.96129	.03878	24.788
1023	825	195	130	3	.95552	.04455	21.450
1126	873	240	169	13	.94938	.05068	18.732
1228	902	297	216	29	.94188	.05818	16.189
1331	924	367	249	40	.93433	.06572	14.218
1433	941	441	286	51	.92711	.07293	12.712
1535	961	506	325	68	.92076	.07929	11.613
1638	973	585	374	80	.91421	.08583	10.652
1740	987	660	413	93	.90837	.09166	9.910
1843	993	744	454	106	.90233	.09770	9.236
1945	995	823	501	127	.89650	.10353	8.659
2047	996	902	546	149	.89097	.10906	8.170
2150	998	985	587	167	.88579	.11424	7.753
2252	999	1066	622	187	.88089	.11913	7.394
2355	1001	1149	665	205	.87627	.12375	7.081
2457	1004	1229	707	224	.87199	.12804	6.810
2559	1004	1314	747	241	.86778	.13225	6.562

TABLE LXX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	61	0	.99841	.00253	394.738
204	202	2	129	0	.99719	.00322	309.314
307	296	11	190	0	.99139	.00887	111.763
409	382	27	253	0	.98471	.01548	63.603
511	469	42	306	0	.98049	.01965	49.894
614	551	63	360	0	.97564	.02447	39.863
716	626	90	406	0	.97052	.02958	32.806
819	699	120	456	0	.96573	.03436	28.106
921	769	152	507	0	.96129	.03878	24.788
1023	828	195	542	0	.95593	.04414	21.657
1126	889	237	580	0	.95133	.04873	19.524
1228	940	288	618	0	.94626	.05380	17.589
1331	978	353	650	0	.94028	.05977	15.733
1433	999	429	686	5	.93338	.06667	14.001
1535	1010	499	734	26	.92622	.07382	12.547
1638	1014	582	783	42	.91901	.08103	11.341
1740	1017	661	827	62	.91232	.08772	10.401
1843	1017	746	874	80	.90584	.09420	9.616
1945	1017	823	924	105	.89985	.10018	8.982
2047	1017	904	968	126	.89424	.10579	8.453
2150	1017	983	1013	150	.88892	.11111	8.001
2252	1017	1064	1048	171	.88396	.11607	7.616
2355	1018	1152	1088	185	.87927	.12076	7.281
2457	1018	1230	1132	209	.87482	.12520	6.987
2559	1018	1308	1177	233	.87060	.12943	6.727

TABLE LXXI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	101	1	1	0	.99841	.00253	394.738
204	202	2	4	0	.99719	.00322	309.314
307	296	11	10	0	.99139	.00887	111.763
409	382	27	19	0	.98471	.01548	63.603
511	469	42	30	0	.98049	.01965	49.894
614	551	63	43	0	.97564	.02447	39.863
716	626	90	57	0	.97052	.02958	32.806
819	698	120	77	1	.96555	.03454	27.958
921	767	152	104	2	.96097	.03910	24.576
1023	827	194	125	2	.95574	.04432	21.563
1126	887	237	153	2	.95103	.04903	19.399
1228	937	289	184	2	.94585	.05420	17.451
1331	972	357	212	2	.93958	.06047	15.539
1433	1001	428	250	4	.93342	.06663	14.009
1535	1019	497	295	19	.92686	.07318	12.665
1638	1021	584	341	33	.91947	.08057	11.412
1740	1022	662	386	56	.91263	.08740	10.442
1843	1023	749	431	71	.90622	.09381	9.660
1945	1023	826	480	96	.90023	.09980	9.020
2047	1023	905	525	119	.89462	.10541	8.487
2150	1023	987	567	140	.88930	.11073	8.031
2252	1023	1071	600	158	.88433	.11570	7.643
2355	1023	1160	640	172	.87959	.12044	7.303
2457	1024	1241	683	192	.87519	.12484	7.011
2559	1024	1321	727	214	.87096	.12906	6.748

TABLE LXXII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	5	0	1.00094	0.00000	R
204	203	1	8	0	.99951	.00091	1101.886
307	297	10	12	0	.99395	.00631	157.509
409	392	17	20	0	.99125	.00894	110.889
511	481	30	33	0	.98719	.01296	76.174
614	566	48	41	0	.98300	.01712	57.409
716	640	76	51	0	.97749	.02261	43.233
819	706	113	64	0	.97133	.02876	33.777
921	772	149	96	0	.96614	.03393	28.472
1023	816	207	120	0	.95866	.04141	23.151
1126	862	261	162	3	.95225	.04781	19.919
1228	883	320	210	25	.94384	.05621	16.791
1331	901	383	256	47	.93588	.06417	14.583
1433	918	446	311	69	.92865	.07140	13.006
1535	926	525	358	84	.92126	.07878	11.694
1638	928	605	403	105	.91392	.08612	10.612
1740	931	686	453	123	.90727	.09276	9.780
1843	936	759	504	148	.90117	.09886	9.116
1945	939	838	553	168	.89541	.10462	8.559
2047	941	912	603	194	.88995	.11008	8.085
2150	944	981	657	225	.88483	.11520	7.681
2252	947	1058	704	247	.88005	.11997	7.335
2355	950	1132	748	273	.87549	.12453	7.030
2457	953	1204	790	300	.87121	.12881	6.764
2559	956	1274	846	329	.86715	.13287	6.526

TABLE LXXIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	55	0	1.00094	0.00000	R
204	203	1	122	0	.99951	.00091	1101.886
307	297	10	180	0	.99395	.00631	157.509
409	392	17	241	0	.99125	.00894	110.889
511	481	30	309	0	.98719	.01296	76.174
614	566	48	364	0	.98300	.01712	57.409
716	640	76	411	0	.97749	.02261	43.233
819	706	113	450	0	.97133	.02876	33.777
921	772	149	489	0	.96614	.03393	28.472
1023	816	207	527	0	.95866	.04141	23.151
1126	871	255	572	0	.95333	.04673	20.401
1228	925	303	612	0	.94856	.05150	18.419
1331	969	362	644	0	.94315	.05690	16.574
1433	999	428	691	6	.93706	.06299	14.876
1535	1005	516	730	14	.92943	.07061	13.162
1638	1012	604	769	22	.92242	.07762	11.884
1740	1013	687	820	40	.91557	.08447	10.839
1843	1014	763	871	66	.90913	.09091	10.001
1945	1015	846	917	84	.90319	.09685	9.326
2047	1017	927	959	103	.89768	.10235	8.771
2150	1018	1007	1007	125	.89239	.10764	8.291
2252	1020	1086	1055	146	.88752	.11251	7.888
2355	1021	1163	1098	171	.88281	.11721	7.532
2457	1022	1245	1132	190	.87840	.12163	7.222
2559	1022	1325	1180	212	.87416	.12587	6.945

TABLE LXXIV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	7	0	1.00094	0.00000	R
204	203	1	12	0	.99951	.00091	1101.886
307	297	10	16	0	.99395	.00631	157.509
409	392	17	27	0	.99125	.00894	110.889
511	481	30	42	0	.98719	.01296	76.174
614	566	48	52	0	.98300	.01712	57.409
716	640	76	64	0	.97749	.02261	43.233
819	706	113	75	0	.97133	.02876	33.777
921	772	149	97	0	.96614	.03393	28.472
1023	816	207	113	0	.95866	.04141	23.151
1126	871	255	144	0	.95333	.04673	20.401
1228	924	303	178	1	.94845	.05161	18.379
1331	964	366	210	1	.94264	.05741	16.420
1433	997	431	259	5	.93683	.06322	14.819
1535	1007	517	299	11	.92955	.07049	13.187
1638	1016	605	337	17	.92270	.07734	11.931
1740	1020	689	387	31	.91607	.08397	10.909
1843	1020	767	436	56	.90956	.09048	10.053
1945	1021	852	480	72	.90361	.09642	9.372
2047	1022	937	520	88	.89804	.10199	8.805
2150	1022	1017	569	111	.89270	.10733	8.317
2252	1022	1098	616	132	.88771	.11232	7.904
2355	1022	1176	659	157	.88295	.11707	7.542
2457	1023	1260	691	174	.87854	.12149	7.231
2559	1023	1338	740	198	.87430	.12573	6.954

TABLE LXXV

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	2	0	1.00094	0.00000	R
204	201	3	3	0	.99652	.00389	255.916
307	299	8	6	0	.99344	.00682	145.732
409	387	22	12	0	.98760	.01258	78.484
511	468	43	19	0	.98137	.01878	52.256
614	544	70	26	0	.97491	.02521	38.671
716	618	98	36	0	.96953	.03057	31.720
819	686	133	58	0	.96384	.03624	26.595
921	747	174	77	0	.95810	.04198	22.824
1023	806	217	108	0	.95281	.04726	20.161
1126	854	264	143	8	.94664	.05342	17.722
1228	902	312	184	14	.94128	.05877	16.016
1331	928	381	215	22	.93412	.06593	14.168
1433	947	456	247	30	.92708	.07297	12.705
1535	960	527	290	48	.92013	.07991	11.514
1638	968	607	324	63	.91328	.08676	10.526
1740	973	693	361	74	.90678	.09326	9.723
1843	978	772	401	93	.90068	.09935	9.066
1945	978	851	443	116	.89473	.10530	8.497
2047	978	928	487	141	.88916	.11088	8.019
2150	979	1007	527	164	.88392	.11611	7.613
2252	982	1082	574	188	.87915	.12088	7.273
2355	985	1161	616	209	.87459	.12544	6.972
2457	988	1233	663	236	.87031	.12971	6.709
2559	989	1318	700	252	.86616	.13387	6.470

TABLE LXXVI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	57	0	1.00094	0.00000	R
204	201	3	119	0	.99652	.00389	255.916
307	299	8	181	0	.99344	.00682	145.732
409	387	22	236	0	.98760	.01258	78.484
511	468	43	287	0	.98137	.01878	52.256
614	544	70	336	0	.97491	.02521	38.671
716	618	98	388	0	.96953	.03057	31.720
819	686	133	439	0	.96384	.03624	26.595
921	747	174	479	0	.95810	.04198	22.824
1023	806	217	520	0	.95281	.04726	20.161
1126	862	264	556	0	.94762	.05244	18.072
1228	919	309	599	0	.94324	.05682	16.602
1331	956	375	626	0	.93715	.06291	14.898
1433	987	446	654	0	.93120	.06885	13.525
1535	1009	523	691	3	.92498	.07507	12.322
1638	1014	608	724	16	.91785	.08219	11.167
1740	1018	698	758	24	.91124	.08879	10.263
1843	1020	782	797	41	.90491	.09512	9.513
1945	1020	867	835	58	.89893	.10110	8.891
2047	1020	946	877	81	.89333	.10670	8.372
2150	1020	1030	917	100	.88801	.11202	7.927
2252	1020	1112	961	120	.88305	.11698	7.549
2355	1020	1197	1000	138	.87832	.12171	7.216
2457	1020	1279	1044	158	.87388	.12615	6.927
2559	1020	1368	1078	171	.86966	.13037	6.671

TABLE LXXVII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	2	0	1.00094	0.00000	R
204	201	3	3	0	.99652	.00389	255.916
307	299	8	7	0	.99344	.00682	145.732
409	387	22	14	0	.98760	.01258	78.484
511	468	43	24	0	.98137	.01878	52.256
614	544	70	33	0	.97491	.02521	38.671
716	618	98	46	0	.96953	.03057	31.720
819	686	133	68	0	.96384	.03624	26.595
921	747	174	82	0	.95810	.04198	22.824
1023	806	217	106	0	.95281	.04726	20.161
1126	862	264	132	0	.94762	.05244	18.072
1228	919	309	169	0	.94324	.05682	16.602
1331	955	376	193	0	.93705	.06300	14.873
1433	981	452	219	0	.93064	.06940	13.409
1535	1009	525	257	1	.92494	.07511	12.315
1638	1016	610	291	12	.91797	.08208	11.184
1740	1019	702	326	19	.91128	.08875	10.268
1843	1020	789	362	34	.90488	.09515	9.510
1945	1020	874	400	51	.89890	.10113	8.888
2047	1020	957	439	70	.89330	.10673	8.369
2150	1020	1044	476	86	.88798	.11205	7.925
2252	1020	1128	518	104	.88302	.11701	7.547
2355	1020	1213	558	122	.87829	.12174	7.214
2457	1020	1295	602	142	.87385	.12618	6.925
2559	1020	1384	636	155	.86963	.13040	6.669

TABLE LXXVIII

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	55	0	1.00094	0.00000	R
204	204	0	120	0	1.00042	0.00000	R
307	307	0	188	0	1.00026	0.00000	R
409	409	0	256	0	1.00019	0.00000	R
511	511	0	327	0	1.00015	0.00000	R
614	614	0	402	0	1.00012	0.00000	R
716	706	10	465	0	.99805	.00205	486.964
819	801	18	535	0	.99666	.00343	290.559
921	896	25	607	0	.99563	.00445	223.850
1023	981	41	679	1	.99337	.00670	148.302
1126	1011	94	759	21	.98455	.01551	63.462
1228	1016	165	840	47	.97399	.02607	37.367
1331	1017	227	921	87	.96399	.03606	26.734
1433	1018	294	1004	121	.95501	.04503	21.207
1535	1018	361	1088	156	.94672	.05333	17.753
1638	1021	418	1172	199	.93924	.06080	15.449
1740	1022	486	1253	232	.93226	.06778	13.755
1843	1022	554	1341	267	.92564	.07440	12.442
1945	1023	615	1429	307	.91959	.08045	11.431
2047	1024	672	1516	351	.91392	.08611	10.613
2150	1024	730	1606	396	.90848	.09155	9.923
2252	1024	801	1692	427	.90340	.09663	9.350
2355	1024	861	1776	470	.89856	.10147	8.856
2457	1024	923	1864	510	.89402	.10601	8.433
2559	1024	984	1947	551	.88970	.11033	8.064

TABLE LXXIX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	5	0	1.00094	0.00000	R
204	204	0	10	0	1.00042	0.00000	R
307	307	0	16	0	1.00026	0.00000	R
409	409	0	31	0	1.00019	0.00000	R
511	511	0	48	0	1.00015	0.00000	R
614	614	0	71	0	1.00012	0.00000	R
716	706	10	102	0	.99805	.00205	486.964
819	801	18	143	0	.99666	.00343	290.559
921	895	26	194	0	.99548	.00460	216.611
1023	979	44	255	0	.99308	.00699	142.113
1126	1015	94	329	17	.98501	.01505	65.468
1228	1020	166	409	42	.97447	.02559	38.083
1331	1021	228	490	82	.96446	.03559	27.103
1433	1022	295	573	116	.95548	.04456	21.440
1535	1023	363	656	149	.94727	.05278	17.949
1638	1024	422	739	192	.93963	.06041	15.555
1740	1024	491	819	225	.93257	.06746	13.823
1843	1024	559	907	260	.92595	.07409	12.498
1945	1024	622	995	299	.91983	.08020	11.469
2047	1024	682	1080	341	.91410	.08593	10.637
2150	1024	740	1170	386	.90866	.09137	9.945
2252	1024	813	1256	415	.90358	.09645	9.369
2355	1024	874	1339	457	.89874	.10129	8.873
2457	1024	939	1426	494	.89419	.10583	8.449
2559	1024	1005	1507	530	.88987	.11015	8.079

TABLE LXXX

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	55	0	1.00094	0.00000	R
204	204	0	120	0	1.00042	0.00000	R
307	306	1	186	0	.99971	.00055	1815.039
409	406	3	250	0	.99886	.00133	749.790
511	504	7	315	0	.99757	.00258	386.731
614	602	12	385	0	.99635	.00377	264.538
716	699	17	453	0	.99532	.00478	208.183
819	796	23	524	0	.99431	.00578	172.141
921	890	31	589	0	.99316	.00691	143.658
1023	969	54	650	0	.99012	.00995	99.502
1126	1005	102	722	19	.98207	.01799	54.576
1228	1012	162	803	54	.97178	.02828	34.366
1331	1015	230	883	86	.96201	.03804	25.288
1433	1018	294	974	121	.95323	.04682	20.360
1535	1020	363	1053	152	.94512	.05492	17.208
1638	1021	431	1135	186	.93750	.06254	14.991
1740	1021	497	1221	222	.93046	.06958	13.373
1843	1021	577	1290	245	.92385	.07619	12.126
1945	1021	643	1373	281	.91774	.08229	11.152
2047	1022	707	1450	318	.91208	.08795	10.371
2150	1022	775	1538	353	.90666	.09337	9.710
2252	1022	854	1621	376	.90159	.09844	9.159
2355	1022	925	1701	408	.89676	.10327	8.684
2457	1022	995	1785	440	.89222	.10780	8.276
2559	1022	1064	1869	473	.88791	.11211	7.920

TABLE LXXXI

NTERM	NDICT	NCH	TCD	NCD	P1	P2	P1/P2
102	102	0	2	0	1.00094	0.00000	R
204	204	0	3	0	1.00042	0.00000	R
307	306	1	9	0	.99971	.00055	1815.039
409	406	3	20	0	.99886	.00133	749.790
511	504	7	39	0	.99757	.00258	386.731
614	602	12	61	0	.99635	.00377	264.538
716	698	18	87	0	.99512	.00498	199.917
819	795	24	127	0	.99412	.00597	166.541
921	889	32	170	0	.99297	.00710	139.789
1023	967	55	227	1	.98978	.01028	96.246
1126	1014	108	298	4	.98310	.01696	57.967
1228	1022	172	378	34	.97291	.02714	35.843
1331	1024	244	458	63	.96303	.03702	26.016
1433	1024	312	548	97	.95397	.04608	20.704
1535	1024	381	630	130	.94568	.05436	17.396
1638	1024	452	711	162	.93798	.06206	15.115
1740	1024	518	798	198	.93093	.06910	13.471
1843	1024	595	869	224	.92432	.07571	12.208
1945	1024	666	951	255	.91821	.08182	11.222
2047	1024	730	1028	293	.91249	.08754	10.423
2150	1024	800	1116	326	.90706	.09297	9.756
2252	1024	877	1200	351	.90199	.09804	9.201
2355	1024	948	1281	383	.89716	.10287	8.721
2457	1024	1019	1364	414	.89262	.10741	8.311
2559	1024	1092	1447	443	.88831	.11172	7.952

APPENDIX II

LIST OF THE 2 SETS OF DATA

(Row-wise in order of decreasing frequency)

LIST 1

the	of	and	to	a	in
that	is	was	he	for	it
with	as	his	on	be	at
by	i	this	had	not	are
but	from	or	have	an	they
which	one	you	were	her	all
she	there	would	their	we	him
been	has	when	who	will	more
no	if	out	so	said	what
up	its	about	into	than	them
can	only	other	new	some	could
time	these	two	may	then	do
first	any	my	now	such	like
our	over	man	me	even	most
made	after	also	did	many	before
must	through	back	years	where	much
your	way	well	down	should	because
each	just	those	people	mr	how
too	little	state	good	very	make
world	still	own	see	men	work
long	get	here	between	both	life
being	under	never	day	same	another

know	while	last	might	us	great
old	year	off	come	since	against
go	came	right	used	take	three
states	himself	few	house	use	during
without	again	place	american	around	however
home	small	found	mrs	thought	went
say	part	once	general	high	1
upon	school	every	does	got	united
left	number	course	war	untile	always
away	something	2	fact	though	water
less	public	put	think	almost	hand
enough	far	took	head	yet	system
better	set	told	nothing	night	end
why	called	eyes	find	going	look
asked	later	knew	point	next	program
city	business	give	group	president	toward
young	days	let	room	side	social
given	present	several	order	national	dinner
rather	second	face	per	among	form
often	things	looked	early	white	case
john	become	large	big	need	four
within	felt	along	children	saw	best
chruch	ever	least	power	light	thing
seemed	family	interest	want	members	mind
country	area	others	done	turned	although
open	god	service	certain	kind	problem
began	different	door	thus	help	sense

means	whole	matter	perhaps	itself	york
times	hunan	law	line	above	name
example	action	company	hands	local	show
five	history	whether	gave	either	today
act	feet	across	3	past	quite
taken	anything	having	seen	death	body
half	really	week	car	field	word
words	already	tell	college	shall	together
money	period	held	keep	sure	probably
free	real	seems	behind	cannot	miss
political	air	question	making	office	brought
whose	special	heard	major	problems	ago
became	federal	moment	study	available	known
result	street	economic	boy	position	reason
change	south	board	job	society	areas
weast	close	turn	love	community	true
court	force	full	cost	seem	am
wife	age	future	voice	wanted	center
woman	common	control	necessary	following	policy
front	sometimes	girl	six	clear	further
land	abel	mother	music	party	provide
education	child	effect	level	students	military
run	short	stood	town	morning	total
outside	figure	rate	art	century	class
north	usually	leave	plan	therefore	evidence
million	sound	top	black	hard	strong
various	believe	play	says	surface	type

value	mean	soon	lines	modern	near
peace	table	red	road	tax	minutes
personal	process	situation	4	alone	english
gone	idea	increas	nor	schools	women
america	living	started	book	longer	cut
dr	finally	nature	private	secretary	third
months	section	call	entirely	greater	expected
fire	needed	ground	kept	values	view
dark	pressure	basis	space	east	father
required	spirit	union	complete	except	moved
wrote	return	support	attention	late	recent
hope	live	brown	costs	else	beyond
forces	hours	nations	person	takimg	coming
dead	inside	low	material	report	srage
data	heart	instead	looking	lqst	miles
read	added	amount	feeling	followed	makes
pay	single	basic	cold	hundred	including
industry	move	research	developed	simply	tried
1960	hold	reached	committee	defense	equipment
island	actually	shown	religious	son	river
ten	beginning	central	getting	sort	doing
received	rest	st	terms	trying	care
friends	indeed	medical	picture	dififcult	fine
simple	sunject	buliding	higher	range	wall
meeting	walked	bring	cent	floor	foreign
paper	passed	similar	final	natural	property
training	county	growth	market	police	england

stare	talk	written	hear	story	suddenly
answer	congress	hall	issus	needs	countries
likely	working	earth	sat	entire	happened
labor	purpose	results	casses	hair	meet
stand	william	fall	food	involved	stock
earlier	increased	whom	bellow	club	effort
letter	knowledge	provided	paid	sent	thinging
using	christian	hour	yes	bill	blue
boyes	certainly	ideas	points	ready	square
trade	10	addition	bad	deal	due
girls	methode	methods	moral	color	decied
directly	nearly	neither	showed	statement	weeks
anyone	kennedy	questions	reading	try	according
french	lay	nation	programs	services	physical
remember	size	comes	member	record	southern
western	normal	strength	appeared	concerned	distract
merely	s	volume	direction	maybe	ran
summer	trial	trouble	1961	5	continued
evening	friend	list	salses	army	generally
influence	led	met	chance	changes	former
husband	opened	science	step	student	aid
average	c	cause	hot	month	series
works	direct	effective	george	lead	myself
piece	planning	soviet	stopped	systems	theory
wrong	ask	clearly	forms	freedom	structure
movement	ways	worked	beautiful	bed	consider
efforts	fear	lot	meaning	note	press

somewhat	spring	hotel	treatment	placed	truth
carried	degree	easy	farm	groups	herself
numbers	plant	respect	wide	j	manner
reaction	approach	feed	game	larger	lower
recently	running	charge	couple	daily	de
eye	arms	blood	persons	march	described
progress	radio	severed	stop	technical	based
chief	decision	image	main	oh	religion
reported	steps	test	window	appear	british
character	europe	gun	midle	account	b
horse	learned	writing	activity	fiscal	green
length	ones	serious	types	activites	audience
corner	forward	hit	letters	lived	nuclear
obtained	returned	slowly	specifec	design	doubt
justice	latter	moving	obviously	plane	quality
straight	born	choice	figures	function	include
operation	parts	pattern	plans	poor	saying
seven	staff	stay	6	cars	gives
shot	sun	whatever	faith	pool	ball
extent	heavy	hospital	lack	mass	speak
standard	waiting	wish	ahead	crops	deep
effects	firm	income	language	visit	principle
15	analysis	designed	distance	expect	growing
none	indicated	price	products	attitude	cities
continue	determine	division	elements	existence	leaders
pretty	serve	stress	afternoon	applied	agreement
closed	easily	factors	hardly	limited	reach

scene	write	30	attack	drive	health
married	remained	rhode	season	station	suggested
covered	current	despite	eight	negro	palyed
role	spent	8	built	council	date
exactly	machine	mouth	original	race	reasons
studies	teeth	unit	becomes	demand	news
prepared	rates	related	relations	rise	supply
bit	director	dropped	e	events	james
officer	playing	raised	sides	standing	sunday
trees	unless	actual	clay	doctor	energy
meant	places	talking	thomas	walk	filled
glass	hjazz	knows	poet	bridge	caught
chicago	claim	conclern	entered	fight	gas
happy	popular	share	style	12	1959
cattle	christ	communist	dollars	follow	heat
included	materials	radiation	status	suppose	thousand
security	accepted	behavior	books	charles	churches
film	giving	opinion	primary	sitting	usual
attemt	changed	funds	hell	marriage	proper
sea	sir	arm	everyone	highly	park
practice	shows	sign	someone	source	tradition
wait	worth	annual	americans	authority	june
lord	oil	older	project	remain	success
fell	jack	obvious	pieces	principal	thin
base	civil	complex	condition	measure	mike
objective	parents	records	u	weight	7
balance	caused	d	dance	equal	kitchen

noted	produced	purposed	clothes	develop	failure
famous	goes	london	names	pass	published
quickly	regard	1958	active	add	announced
bottom	break	carry	check	cover	enemy
greatest	key	king	laws	leaving	manager
mary	morover	pain	poetry	sources	20
battle	bright	facts	carefully	companies	finished
fixed	operating	product	spoke	touch	units
allowed	build	citizens	died	financial	inches
loss	otherwise	patient	previous	require	rose
seeing	sight	takes	workers	capital	captain
classes	concept	german	marked	musical	rules
shape	stated	stations	variety	affairs	appears
aware	begin	broad	catholic	learn	m
named	post	proposed	remains	reports	sex
strange	w	bank	capacity	governor	henry
houses	interests	mark	offered	officers	opening
p	prevent	regular	robert	ship	slightly
speed	spread	team	winter	yesterday	bar
crisis	drink	fresh	instance	poems	presented
produce	train	youth	25	agreed	apartment
campaingn	cells	created	essential	event	file
forced	germany	immediate	index	Yives	neck
nine	opposite	provides	round	subjects	trip
watch	watched	explained	features	fully	gray
indicate	lady	offer	russian	session	teacher
twenty	desire	economy	maximum	mentioned	procedure

reality	reduced	sam	separate	studied	term
beside	coffee	edge	enter	fast	favor
literary	lookes	mission	picked	secret	smaller
tone	address	anode	belived	editor	election
fair	follows	judge	laid	model	permit
response	rights	solid	title	bottle	buildings
formed	hearind	knife	memory	presence	quiet
receive	region	telephone	watching	camp	dog
expressed	fit	junior	killed	murder	nice
official	planned	removed	rock	stayed	treated
turning	virgina	vote	ability	berlin	chapter
claims	contrast	du	faculty	failed	fourth
frame	france	gain	h	interior	jewish
jr	leader	nobody	november	personnel	pointed
positive	rich	selected	send	standards	store
twice	advantage	brife	brother	die	louis
observed	powers	pulled	rule	valley	writer
writers	accept	allow	assumed	boat	broke
command	daughter	detail	everybody	evil	faces
familiars	fields	figs	fighting	hill	increases
items	jones	legal	master	morgan	ordinary
phase	platform	plus	resources	russia	sharp
somehow	upper	village	wine	april	beauty
carrying	chosen	column	copared	constant	factor
forth	mercer	proved	richard	smiled	unity
universe	wants	aircraft	box	buy	calls
cleans	communism	danger	dogs	drawn	dust

exchange	t	foot	naturally	rain	rome
san	sections	shelter	song	sweet	waited
walls	100	ancient	china	completed	fashion
league	levels	liberal	lips	ordered	politics
realize	realized	seek	settled	teachers	texas
weather	willing	actions	animal	article	beat
directed	drew	dry	electric	emotional	excellent
families	fifty	frank	horses	intials	largely
leading	monday	ought	pictures	policies	practical
projects	quick	signs	stands	starting	trafic
won	answered	aside	asking	career	chairman
dress	estimated	flat	flow	impact	jury
legs	occurred	paris	referance	saturday	sit
teaching	thick	warm	wonder	yourself	adequate
arts	besides	brith	block	capable	chair
closely	cutting	declared	ends	fingers	forest
gross	hanover	helped	honor	issuses	measured
durselves	page	plays	r	relife	score
search	attorney	cell	desk	discussed	dominant
employees	escape	gets	holding	hung	jobes
join	july	newspaper	object	objects	passing
phil	rapidly	sleep	supposed	typical	wore
aspects	belief	bodies	credit	dream	empty
explain	grew	jesus	lads	located	matters
message	minimum	primarily	site	towards	yards
50	argument	assume	benefit	broken	contact
domestic	dramatic	fellow	grow	happen	highest

kill	location	narrow	parker	powerful	relation
rifle	shop	signal	tom	tomorrow	unusual
wind	9	achieved	agencies	appeal	arrived
baby	billion	careful	cool	december	drove
equally	extreme	fund	greatly	guests	homes
internal	library	officials	please	pleasure	portion
recognize	reduce	rising	save	senate	sets
speaking	struggle	willson	important	11	acting
beach	boston	closer	coast	courses	duty
eat	european	feelings	friendly	greek	ideal
imagine	kid	la	metal	minister	possibly
prices	shore	shoulder	showing	soft	speech
tests	travel	vast	victory	weapons	advance
armed	chemical	circle	contained	contract	ended
exiting	fat	friday	garden	goal	heavily
judgment	keeping	learing	machinery	maintain	moon
nose	onto	refused	scale	setting	slow
somewhere	stared	streets	task	technique	text
bible	budget	drop	exist	finds	formula
headed	housing	lie	mine	notice	novel
paitnting	parties	plants	providing	repeated	roof
sensitive	sexual	snow	solution	songs	stories
struck	taste	tension	thirty	tree	tuesday
ultimate	uses	animals	avoid	busy	causes
commerce	critical	culture	dallas	emphasis	establish
etc	exercise	fairly	grounds	hence	hole
india	leg	liked	lose	minor	negroes

dccasion	drders	pale	perfect	railroad	remove
roads	artist	baseball	bay	bear	becoming
beneath	birds	charged	congo	details	enjoyed
entrance	flowers	goods	informed	lewis	majority
notes	permitted	processes	professor	replied	requires
sample	shook	somebody	spot	truck	truly
uncle	academic	agency	apply	bought	chinese
double	draw	entitled	evident	fifteen	granted
guess	hat	intensity	joined	loved	mineds
motor	organized	palmer	pure	regarded	review
september	soldiers	plenty	spite	vision	vital
wage	wheel	wild	artists	begins	brftian
conduct	conducted	cross	cultural	demands	device
divided	executive	extended	firms	fort	games
identity	improved	inner	joe	joseph	1
marine	martin	motion	o	pick	properly
rooms	runs	sought	sounds	tall	theme
wagon	win	wished	wood	yellow	14
attend	create	dear	decisions	faced	huge
item	jews	lake	machines	phone	positions
putting	risk	rural	seat	smith	spil
suggest	supported	symbol	thoughts	trained	unable
walking	absence	assigned	august	band	bitter
breakfast	breath	chest	content	crowd	depth
disease	driving	examples	finding	grass	handle
hudson	january	japanese	largest	loose	minute
negative	payment	percent	practices	prove	pushed

remarks	sin	slight	spend	troops	vehicles
version	welfare	wet	windows	wonderful	advanced
alfred	bedroom	centers	conflict	contrary	correct
detailed	detective	dozen	element	flesh	gold
hero	indian	intoduced	los	raise	silence
telling	theater	trust	widely	abroad	achieve
advice	ages	agree	angle	approval	arthur
begun	boats	colors	cousin	david	devoted
easier	elections	estate	foods	gradually	guy
laughed	listen	nodded	october	papers	player
pull	rear	shoulders	sick	stages	stream
supreme	surprise	throat	uniform	views	warren
waves	16	assembly	brilliant	burning	chain
childhood	choose	convinced	courts	desired	engine
expense	extra	extremly	female	fill	g
hills	institute	issued	knowing	latin	meption
moments	motors	noticed	pair	proud	strike
taught	till	tiny	towns	welcome	wooden
worse	18	count	creative	depends	driver
employed	firmly	moly	incident	leaves	measures
milk	millions	operator	partly	passage	payments
request	ride	silent	speaker	sports	tendency
worry	tragedy	24	anger	attitudes	charlie
co	concrete	cry	destroy	drinking	formal
functions	grand	guard	hearst	hoped	hopes
limit	liquid	mantle	mile	missile	operate
pink	plain	poem	precisely	quitely	royal

screen	shooting	sorry	suit	swung	tired
twelve	via	angeles	aspect	bills	blind
boards	bonds	cook	cuba	dented	deny
engaged	expansion	expenses	fears	grant	honest
humor	italian	lights	lincoln	luck	mail
mere	models	moscow	northern	numerous	opera
patterns	periods	prior	purchase	ring	rolled
safety	singing	skin	sold	soul	stairs
supplies	surely	thousnds	unxndwn	vacation	wearing
13	anyway	artery	atomic	author	avenue
award	bond	centuries	chamber	conscious	creation
curious	dangerous	decade	doctrine	dollar	encourage
fiction	flight	fought	georgia	insurance	liberty
loan	losses	native	panels	pocket	precision
relative	salt	seriously	shares	shut	superior
threw	trend	vidlence	wave	weakness	wright
1954	1957	23	adopted	africa	alexander
angry	ballet	brain	calling	cast	charges
contain	cup	curve	depend	diameter	discovery
edward	elsewhere	february	feels	ice	includes
intended	load	lucy	meat	medium	mold
mounted	offers	offices	prime	primise	promised
qualities	referred	riding	sheet	steel	sum
target	taxes	terrible	universal	valuable	watson
21	22	acres	adam	admitted	agent
amounts	answers	arranged	asia	brush	burden
changing	climbed	collected	confused	confusion	driven

em	enjoy	errors	expensive	extensive	fired
fun	hans	helping	hundreds	lies	listed
lovely	mama	mobile	mostly	nearby	odd
opinions	origin	path	pilot	sale	seeking
shoes	slaves	snake	spirits	suffering	tables
thickness	volumes	warning	washing	wisdom	younger
60	believes	bureau	catch	cloth	coat
comfort	concerns	consists	dancing	darkness	dealing
dirt	drama	emotions	external	flying	grown
heads	heaven	insisted	iron	lawyer	lifted
liquor	marketing	mental	mountains	occur	oxygen
pace	porch	pounds	rapid	raw	reaching
reader	readily	recorded	republic	resulting	route
salary	saved	separated	ships	suffered	switch
tend	tour	warfare	whenever	adams	anne
anxiety	anybody	appointed	bag	bound	comment
crossed	demanded	distinct	ease	editorial	engineer
excess	exists	express	fed	golden	guns
hardy	hate	holds	journal	linda	lots
muscle	obtain	particles	pride	prison	rachel
reactions	reduction	reflected	regional	peplaced	reply
seeds	skill	smooth	thkeat	throw	touched
unlike	urban	varied	varying	wages	waters
weapon	wire	arc	atoms	bread	brothers
carl	continues	codling	describe	display	downtown
favorite	francised	funny	henrietta	involves	kate
limits	match	musicians	n	opposed	pike

pleased	proposal	queen	rare	rarely	remining
removal	rough	sake	seed	sell	shift
smoke	societies	spending	steady	stepped	storage
teach	tissue	vice	virtually	visited	whereas
writes	afford	approved	atlantic	automatic	bars
bob	brings	burned	code	combined	composed
criticish	customers	gean	decide	democrats	dependent
discover	drawing	eleven	exception	existed	finger
focus	glance	goals	grace	guidance	handsome
happens	highway	illindis	inch	indicates	intense
joy	languages	missed	necessity	neighbors	notion
orleans	painted	papa	parallel	permanent	pope
prominent	proof	regarding	regions	rode	self
senator	shared	shear	shouted	stranger	studying
talent	thrown	tool	treasury	visual	walter
winston	acts	agents	allotment	anywhere	assured
colleges	comedy	concert	deeply	defined	derived
destroyed	emergency	estimate	finish	forever	furniture
gained	guest	holes	hydrogen	ill	improve
joint	lawrence	listening	mad	magazine	mankind
maturity	mystery	neutral	rayburn	recall	revealed
selection	severe	sleeping	soldier	stick	striking
trials	accounts	attempts	axis	briefly	bringing
clouds	contains	copy	cotton	districts	ears
experts	fifth	forgotten	glad	handed	handling
instant	japan	knees	leaned	mayor	ohio
onset	organic	palace			

LIST 2

the	of	and	in	a	to
for	on	history	new	an	from
american	education	report	world	with	guide
life	law	developmen	states	study	staf
united	by	at	economic	book	america
modern	english	1968	man	studies	research
act	theory	public	introducti	social	art
s	analysis	south	1967	management	water
century	his	county	survey	how	internatio
science	school	government	great	as	war
handbook	national	program	africa	north	other
control	its	problems	1969	family	literature
planning	i	system	business	systems	story
federal	is	first	health	1966	british
some	u	administra	industry	england	manual
west	years	policy	john	east	human
india	selected	library	city	thier	bibliograp
poems	land	general	industrial	under	children
politics	society	time	your	proceeding	plan
church	old	area	political	use	geology
methode	tax	indian	resources	1965	essays
black	work	practice	principles	people	early
my	agricultur	foreign	service	services	community
trade	western	power	high	dictionary	europe
music	design	stories	teaching	california	programs

c	relations	through	aspects	our	year
revolution	york	two	economics	river	training
urban	data	negro	structure	1964	civil
materials	sea	one	what	you	french
laws	university	all	college	philosophy	poetry
william	african	central	basic	home	growth
review	works	schools	house	language	men
computer	engineerin	letters	physical	southern	about
air	informatio	soviet	three	death	elementary
it	chemistry	virginia	christian	short	young
1	e	r	australia	books	love
slavery	local	conference	space	illinois	educationa
god	medical	progress	britain	german	ancient
economy	physics	age	de	middle	production
technology	future	or	tales	texas	canada
atlas	change	historical	washington	chemical	no
thomas	contempora	be	psychology	regional	labor
papers	who	marketing	military	special	action
code	employment	small	mathematic	rules	union
way	d	effects	white	into	district
evaluation	field	behavior	second	soil	summary
technical	china	day	directory	making	religion
up	complete	related	techniques	women	approach
child	food	james	project	collection	european
carolina	london	market	nuclear	region	constructi
eastern	henry	libraries	mexico	reading	republic
readings	relating	latin	rural	natural	pakistan

safety	congress	statistics	today	financial	france
income	list	that	committee	h	mental
freedom	ii	living	this	applicatio	asia
department	housing	little	st	between	country
notes	australian	during	greek	times	catalogue
island	northern	twentieth	criminal	energy	reform
scientific	canadian	distributi	critical	free	king
last	long	manpower	peace	2	care
court	index	organizati	plays	role	russian
areas	building	geography	highway	youth	five
learning	practical	properties	commercial	light	problem
symposium	alaska	case	japan	out	developing
population	sciences	commission	elements	plants	present
products	1960	medicine	money	more	nature
transporta	culture	model	pasific	activities	chinese
cookbook	facilities	lectures	legal	nomination	pennsylvan
plant	students	thought	forest	gold	nations
reference	1963	biology	robert	theatre	welfare
zealand	accounting	architectu	higher	israel	understand
arts	george	office	records	cases	germany
real	rights	before	bible	crisis	deffense
good	mechanics	personnel	title	basin	counties
countries	finance	golden	p	party	regulation
aid	farth	four	ground	investment	mineral
red	are	company	course	days	evolution
experiment	group	insurance	literary	patterns	procedures
process	processing	self	standards	transport	valley

writing	adventures	clinical	june	co	diseases
nursing	crime	current	farm	ireland	1
policies	roman	royal	spanish	after	b
drawings	investigat	lake	recreation	statistica	characteri
laboratory	movement	operations	organic	secondary	view
we	1962	commoditie	constituti	islands	non
sources	1970	3	assignment	conflict	experience
flow	gas	labour	marine	n	standard
superscrip	encycloped	families	foundation	march	opportunit
road	sex	town	treatment	trends	1945
anatomy	florida	functions	painting	processes	quality
series	1961	coast	cost	student	teachers
third	toward	animals	changing	ohio	paintings
sam	sociology	drama	estate	heart	irish
mass	me	recent	against	comparativ	heading
mystery	needs	over	security	since	vocational
communicat	do	folk	low	measuremen	method
night	private	reports	capital	common	fiscal
game	michican	park	personal	radiation	teacher
administrb	iography	equipment	look	mind	pictures
police	practices	procedure	projects	street	communist
computers	groups	medieval	names	official	oregon
renaissanc	words	big	colorado	descendant	fundamenta
ideas	impact	j	maryland	museum	property
rise	search	slave	without	writings	among
concrete	effect	empire	environmen	including	journey
july	kingdom	mission	performanc	algebra	columbia

comprehens	conservati	georgia	indians	inventory	japanese
metropolit	recommenda	tradition	wild	aircraft	biological
charles	effectiven	electrical	fire	italian	legislatio
linear	major	maps	master	memoirs	place
religious	taxation	testament	upper	when	atomic
board	challenge	concepts	disease	g	hospital
line	motor	oil	past	pollution	programmin
rehabilita	stage	victorian	working	1914	certain
film	journal	la	logic	not	october
origin	paper	radio	railway	shakespear	t
they	wales	was	agencies	animal	applied
banking	birds	civilizati	commonweal	electronic	function
jesus	let's	mr	ocean	profession	russia
sir	x	1900	cultural	dynamics	hundred
institutio	kentucky	know	nineteenth	record	sales
six	status	theology	children's	cities	conditions
corporatio	force	jersey	massachuse	tree	w
word	archaeolog	go	lost	may	moon
mountain	now	observatio	order	price	secret
senate	solid	supply	why	amendments	atlantic
colonial	costs	democracy	effective	factors	green
job	make	need	nevada	o	operation
outline	poets	seven	village	where	adult
ages	army	color	corps	decisions	eighteenth
f	growing	heritage	potential	preliminar	tennessee
wood	brife	credit	determinat	drugs	exchange
fiction	fish	fourth	geometry	her	lord

novel	part	recipes	stars	temperatur	thermal
v	voyage	boy	center	commentary	cook
decision	drug	faith	interest	iron	legislativ
machine	magic	off	reactions	revision	structures
tables	test	trial	truth	around	bay
beyond	convention	criticism	final	hawaii	house
influence	lands	large	like	lower	made
manufactur	marriage	mexican	perspectiv	physiology	play
post	publicatio	quantum	range	rates	ray
richard	soils	songs	steel	transfer	1950
better	blue	digest	executive	genus	heat
issues	january	louisiana	many	metal	resource
rock	rocks	vietnam	1958	best	body
christmas	colour	cooking	cotton	cross	documents
equations	garden	instructio	late	live	mentally
mississipp	personalit	port	questions	right	scotland
session	spirit	summer	sun	surface	universiti
ussr	utilizatio	affairs	can	coal	differenti
engineers	form	frontier	grammar	italy	jewish
k	key	missouri	poverty	press	structural
towards	traffic	1800	anti	april	arkansas
associatio	class	creative	curriculum	december	famous
favorite	friends	glass	iii	illustrate	neutron
ontario	portrait	race	seminar	species	speech
subscript	technique	testing	transition	upon	waters
woman	4	annotated	anthology	changes	christ
compositio	consumer	cooperatio	girl	minnesota	paul

pioneer	president	relation	sound	spain	style
value	ways	wind	advanced	arizona	battle
bridge	classifica	collected	colleges	deposits	diagnosis
discovery	fifty	games	have	implicatio	institute
instrument	interpreta	joseph	mining	monetary	open
peoples	peter	reader	retarded	sculpture	americans
bank	blood	calculus	cancer	compensati	council
czechoslov	down	dream	electron	ethics	maintenanc
means	montana	moral	nutrition	poft	prose
revenue	surgery	television	ten	things	august
catholic	concept	electric	forms	historic	hope
kansas	magnetic	nation	navy	number	queen
silver	sketches	twenty	type	using	will
wisconsin	agreement	assembly	aviation	catalog	classical
diary	end	examinatio	guidelines	hill	improvemen
inquiry	level	mary	origins	plans	poor
protection	round	selection	seventeent	star	stock
strategy	uses	10	dental	dynamic	evidence
exhibition	eye	grand	hearings	metals	models
mountains	narrative	proposed	repair	requiremen	roads
scattering	spring	workers	1850	1959	alabama
alexander	buildings	career	cat	connecticu	contributi
correspond	courts	egypt	essentials	fall	forces
genealogy	greece	image	lady	lincoln	louis
machines	numerical	outdoor	pattern	period	photograph
popular	psychologi	september	travel	treasury	15
1957	centuries	creek	facts	financing	guidance

hand	map	plain	procuremen	reflection	side
single	steam	unions	wall	1860	1917
attitudes	behaviour	brazil	communism	concerning	dakota
essay	extension	held	knowledge	november	philippine
pictorial	primary	song	synthesis	theoretica	values
12	artist	beginning	commerce	compounds	domestic
earthquake	ecology	edward	emergency	far	february
idaho	loan	martin	mother	northwest	point
printing	province	quest	question	response	secretary
supreme	tests	travels	various	verse	acid
america's	annual	answers	compilatio	doctor	flight
holy	indiana	internal	kennedy	material	media
municipal	occupation	own	phase	philosophi	psychiatry
retirement	samuel	strength	table	textbook	them
tomorrow	trees	universe	utah	van	1789
67	approaches	arms	assessment	boundary	budget
canal	cape	collective	concise	customs	dance
deep	democratic	descriptio	fields	fifth	johnson
juvenile	man's	manuscript	molecular	musical	operating
results	section	ships	speaking	storage	sweden
terms	violence	weather	1939	5	acts
advertisin	amended	centers	chicago	choice	comparison
films	fishes	full	idea	individual	introducto
meaning	netherland	novels	ordinances	pressure	quantitati
rome	speeches	stratigrap	support	taxes	testimony
topics	us	1940	1947	30	68
7	activity	autobiogra	christiani	counseling	dark

david	delinquenc	demand	dift	digital	efficiency
exploring	fishing	flood	half	heroes	homes
junior	letter	mark	masters	matter	mechanical
near	norway	profile	settlement	so	son
southeast	stability	staff	station	struggle	theories
therapy	true	unit	victoria	abstracts	accounts
adventure	agency	bill	character	conquest	continenta
corporate	districts	division	father	governor	hospitals
interior	liquid	markets	minerals	most	motion
mrs	native	nigeria	operative	periodical	persons
peru	petroleum	poland	prince	prospects	reality
rule	saint	savings	simple	simplified	than
thinking	tragedy	wildlife	1830	1918	1955
9	account	authors	bargaining	bird	careers
census	complex	disorders	dog	flowers	flying
francisco	furniture	generation	ghost	handling	hills
myth	officers	optical	pradesh	presented	revised
romance	sense	smith	soul	sport	subject
survival	types	18	1956	academic	achievemen
advances	astronomy	authority	beauty	birth	brain
clay	coins	contract	dante	dimensions	feasibilit
funds	gospel	graphic	handicappe	immigratio	iv
languages	lecture	naval	participat	picture	portraits
pottery	pre	preparatio	proposals	railways	relationsh
resonance	scientists	socialist	solids	stress	voice
waves	wonderful	wyoming	1700	1815	1865
8	anniversar	biographic	bombay	build	childhood

coastal	cold	communitie	desert	directions	friend
hero	houses	if	inner	interactio	joint
liberty	lives	phenomena	plasma	plastics	pocket
reactor	select	statements	used	vehicle	visit
voices	wilderness	27	alloys	appraisal	articles
background	behind	being	benefits	checklist	come
companies	contracts	dead	disaster	double	dr
el	encyclopae	establishm	face	flora	flower
fluid	glossary	integratio	landscape	left	legends
meeting	memorial	merrill	nationalis	nine	pilot
projection	railroad	regions	renewal	rhodesia	romantic
socialism	southeaste	thoughts	visual	walter	yugoslavia
1949	back	blance	boston	came	cell
climate	decay	elizabeth	era	expedition	gods
grades	grass	gulf	independen	indonesia	iowa
jr	lakes	maine	mammals	mechanisms	metabolism
migration	nothing	oklahoma	only	organisati	pay
poultry	primer	quadrangle	salt	scottish	solar
spectra	thermodyna	thirty	trail	wheat	within
66	abroad	absorption	allied	analytical	arab
artists	bear	behavioral	branch	car	castle
cooperativ	die	diplomatic	discoverin	election	enforcemen
functional	genetics	geologic	girls	god's	grain
irrigation	los	managing	members	overseas	pairs
parish	parts	philip	probabilit	productivi	rate
recollecti	retardatio	see	sixth	technologi	tiger
towns	twelve	unemployme	uniform	volume	wages

weapons	well	were	wine	world's	y
100	16	1920	1948	adjustment	anthropolo
appropriat	available	based	brown	check	chief
colony	copper	correlatio	decade	defence	door
elizabetha	employees	fair	football	forestry	forests
francis	fund	heaven	heavy	honor	insects
ion	korea	location	memories	nam	nurses
parks	particle	places	poetical	portugal	postal
prices	published	revolt	scale	simulation	spectrosco
successful	text	tropical	variations	vehicles	wage
wave	winter	1870	1890	20	50
affecting	arthur	asian	atmosphere	authorizat	basis
captain	classroom	competitio	daniel	devonian	exploratio
export	fast	forty	frederick	gamma	gardens
get	grade	greatest	guinea	intelligen	inter
later	legacy	liberal	linguistic	luther	ministry
mount	multiple	news	nova	opinions	oral
ore	partial	pathology	pleasure	polish	pope
powers	presidents	principal	reorganiza	room	roots
solutions	source	texts	trials	what's	written
1919	6	acquisitio	along	am	anglo
bengal	breeding	caribbean	ceylon	coming	cretaceous
descriptiv	desserts	disposal	eight	electricit	enterprise
fuel	gift	here	intellectu	leadership	livestock
makers	milk	morning	morphology	murder	numbers
oceanograp	path	payments	prevention	printed	profiles
programme	protest	psychiatri	reconstruc	residentia	responsibi

return	rice	run	ship	size	sky
solution	stephen	streams	subjects	there	those
treasure	treasures	treatise	turkey	unity	versus
viet	views	vision	walk	wealth	writers
yorkshire	13	20th	31	65	69
address	agents	approved	associated	automatic	bar
base	basketball	biochemist	boys	cars	cells
churches	circuits	clur	colored	craft	creation
delaware	doctrine	edgar	equal	expansion	finland
fisheries	fox	fun	further	geological	hard
head	horses	hydrology	interim	jews	journals
maharashtr	meat	network	ninety	optimal	outlook
patient	points	puerto	rebellion	phetoric	sand
selling	snow	soldier	statutes	stone	submitted
success	teach	term	tudor	tune	undergroun
units	worlds	1783	1867	24	active
addresses	automation	ballads	beautiful	beginners	brother
call	campaign	circulatio	citizens	containing	cookery
cycle	destiny	different	disadvanta	employee	estimates
every	fairy	finite	forecastin	fortran	frank
frequency	geographic	golf	graduate	grants	hampshire
highways	ice	identifica	inside	interstate	it's
jerusalem	leaders	levels	lewis	measures	medium
mill	mine	miss	morality	navigation	never
nonlinear	norman	operators	prayer	presidenti	prints
profit	promise	psychother	racing	reaction	reformatio
remarks	reserve	rose	sacred	shaw	shells

site	sixteenth	skin	southwest	southweste	sports
territory	textile	tour	track	treaty	which
1941	1951	25	adults	aids	album
ancestors	arctic	baby	boat	bodies	bridges
budgeting	but	causes	circle	compendium	crystals
daily	dear	delta	devices	diplomacy	direct
does	dogs	dragon	drawing	elastic	faces
frontiers	gallery	gandhi	geochemica	glory	going
goods	governing	gun	hamilton	hidden	holiday
hungary	imperial	improving	inspection	ladies	lancashire
lawrence	learn	lines	margaret	mechanism	meet
milton	northeaste	others	oxygen	pioneers	primitive
print	product	programmed	propagatio	proverbs	purpose
rajasthan	rare	recognitio	relative	reminiscen	retail
rivers	saskatchew	seed	selective	shell	should
significan	silence	square	statement	strategic	these
total	uranium	very	waste	wisdom	writer
z	1776	1922	aging	alered	alliance
alpha	analytic	atmospheri	benjamin	biblical	claims
companion	congo	content	conversion	criteria	cry
crystal	daughter	designs	detection	draft	dreams
dutch	easy	establishi	estimation	expenditur	fact
folklore	forgotten	gases	giant	greater	hall
hebrew	hydrologic	imperialis	indies	indo	innovation
liquids	lyrics	malaysia	measure	memoir	meteorolog
mouse	mysore	myths	nuclei	orders	parliament
pension	philadelph	position	random	register	rembrandt

russell	santa	seasons	secrets	senior	shadows
something	speak	springs	sugar	supervisio	surveys
taylor	textiles	theodore	thousand	timber	transmissi
vegetables	vi	vicinity	von	ward	week
women's	yellow	zone	zoo	1750	1812
1837	1848	1861	1930	1953	1954
advisory	again	argentina	banks	beef	ben
beverages	blind	bond	brazilian	campus	cattle
centennial	clark	conduct	controllin	crops	damage
darkness	demonstrat	dialogue	dilemma	dimension	dimensiona
eyes	fighting	fine	flowering	fly	focus
fresh	fruits	ghosts	grant	gravity	heating
hindu	hotel	hours	hunting	improved	inorganic
insect	journalism	kinetics	kings	let	loss
managerial	maritime	merchant	methodist	neighborho	next
noise	normal	opinion	ordinary	painter	paradise
pastoral	peninsula	pet	physiologi	portuguese	prophets
proposal	provisions	racial	read	reason	rhode
scotia	seas	securities	separation	setting	seventh
shadow	shipping	sic	silent	singapore	situation
societies	socialogic	surgical	tale	terror	thailand
themes	thin	tomorrow's	trip	vegetation	worker
1775	1868	1869	1910	1975	21
64	abc	abuse	adams	adjacent	adopted
affair	agreements	amateur	angeles	angels	antarctic
antique	armed	away	belt	bright	bront
camera	centre	ceramic	charlotte	clean	collecting