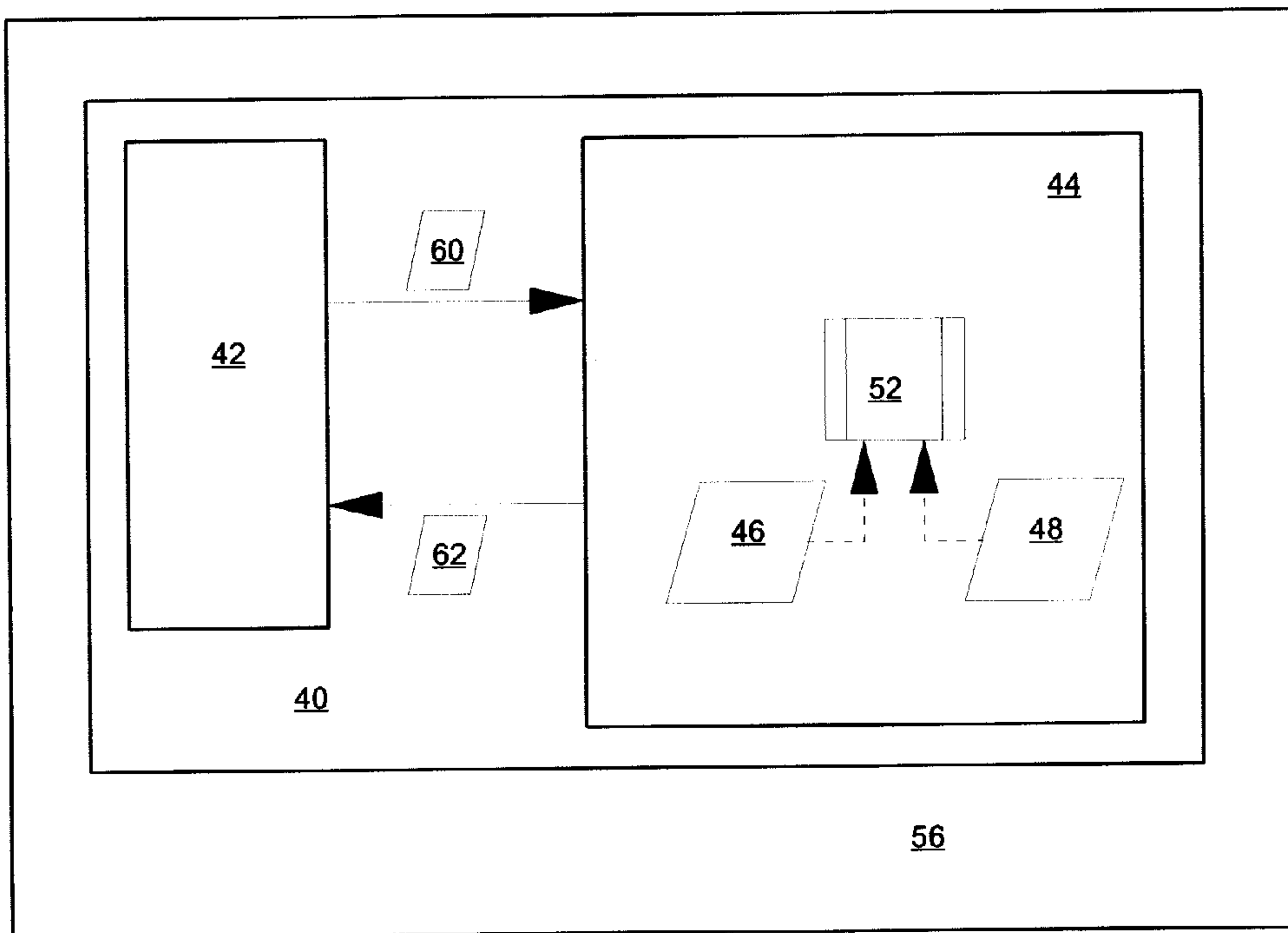




(22) Date de dépôt/Filing Date: 2002/06/18
(41) Mise à la disp. pub./Open to Public Insp.: 2003/12/18

(51) Cl.Int.⁷/Int.Cl.⁷ G06F 17/27, G06F 7/00
(71) Demandeur/Applicant:
IBM CANADA LIMITED-IBM CANADA LIMITEE, CA
(72) Inventeurs/Inventors:
FLASZA, MIROSLAW A., CA;
SHARPE, DAVID C., CA
(74) Agent: ROSEN, ARNOLD

(54) Titre : SYSTEME ET METHODE DE TRI DE DONNEES
(54) Title: SYSTEM AND METHOD FOR SORTING DATA



(57) **Abrégé/Abstract:**

A method for comparing data, and in particular character data, is disclosed. Two pieces of data are compared to determine if they are within an equivalence class based on using a dictionary sort order table with a non-unique collating sequence. If so, the pieces of data are compared using a dictionary sort order table with a unique collating sequence. The comparison method may be implemented within a sorting module that receives an input data set and then uses the comparison to compare two pieces of data in the input data set at a time. The sorting module uses the result of the comparison method to sort the input data set into equivalence classes. The results of a second comparison provides data sorted within equivalence classes. The sorting module may provide sorting services to a database management system or to a calling program.

SYSTEM AND METHOD FOR SORTING DATA

ABSTRACT

5

A method for comparing data, and in particular character data, is disclosed. Two pieces of data are compared to determine if they are within an equivalence class based on using a dictionary sort order table with a non-unique collating sequence. If so, the pieces of data are compared using a dictionary sort order table with a unique collating sequence. The comparison method may be implemented within a sorting module that receives an input data set and then uses the comparison to compare two pieces of data in the input data set at a time. The sorting module uses the result of the comparison method to sort the input data set into equivalence classes. The results of a second comparison provides data sorted within equivalence classes. The sorting module may provide sorting services to a database management system or to a calling program.

10

15

SYSTEM AND METHOD FOR SORTING DATA

Field of the Invention

The present invention relates to a system and method for sorting data. More particularly, the invention relates to sorting character data into equivalence classes and within equivalence classes.

Background of the Invention

Sorting character data is a common operation performed by computer systems.

The English language, like many languages, makes use of multiple forms of letters in an alphabet. Each English letter has an uppercase form and a lowercase form.

Various grammatical rules require the use of the uppercase and lowercase letters in particular circumstances in written English. In addition, writers may elect to use uppercase and lowercase letters to emphasize words or for other reasons. The use of uppercase or lowercase letters does not normally affect the meaning of an English word, and all variations of the English word are generally considered to be equivalent to one another.

Words are often sorted alphabetically based on a standard dictionary sort order, without regard to whether they are written using uppercase letter, lowercase letter or a mixture of uppercase and lowercase letters. For example, the words "Chad", "CHAD" and "chad" are generally considered equivalent by most readers. Any version of the word "alpha" would be alphabetized before any version of the word "chad", and any version of the word "delta" would be alphabetized after any version of the word "chad". The three versions of the word "chad", as well as other versions such as "cHAd", can be said to be in a single equivalence class, when words are organized alphabetically. Within such an equivalence class, one typical method of alphabetizing different forms of a word is to give precedence to an uppercase letter over a lowercase letter. Accordingly, the three versions of "chad" above may be ordered as follows: "CHAD", then "Chad", and then "chad".

Computer systems use character sets that are used to form coded character strings to represent words. Typically, a character set will include different characters for each form of a letter. A common character set used by digital computers is the ASCII character set which provides distinct coded characters for representing all uppercase forms of letters and distinct coded characters for representing all lowercase forms of letters. To the digital computer system, the different coded characters ('coded character' is hereinafter referred to as 'character') are unrelated to one another, and character strings formed using the different characters are seen by the computer system as distinct from one another.

A computer system would see the three character strings "Chad", "CHAD" and "chad" as distinct from one another. As a result, the computer system may not alphabetize the character string "alpha" before the character string "CHAD". The computer system may also not alphabetize the character string "DELTA" before the character string "chad". In general, the computer system cannot use its basic character set to sort words in the same way that a person would. To allow computers to group different forms of the same word, dictionary sort order tables are defined to map the dictionary sort order to the order of characters in the computer system's character set.

Dictionary sort order tables may have a unique collating sequence that allows all character strings to be distinguished from one another and organized in a desirable sequence, such as the alphabetic sequence described above. Such sort order tables have the problem that they cannot be used to identify character strings that are in the same equivalence class – i.e. they are different forms of the same word using different combinations of uppercase and lowercase letters.

Other dictionary sort order tables have a non-unique collating sequence that allows character strings in the same equivalence class to be identified, but they cannot be used to order the strings in a desirable order within an equivalence class.

Accordingly, a solution that addresses, at least in part, this and other shortcomings is desired.

Summary of the Invention

The invention provides a system and method for sorting data such as character data by equivalence classes as well as within equivalence classes. In one embodiment, the present invention provides a method of comparing two pieces of data in which the two data are first compared using a dictionary sort order table with a non-unique collating sequence, and if the two pieces of data are found to members of the same equivalence class, they are compared again using a dictionary sort order table with a unique collating sequence, and the result of the second comparison is used to identify which of the two pieces of data should be presented first.

In an aspect of the invention, there is provided a method of ordering a first character string and a second character string including comparing the first character string and the second character string according to a first dictionary sort order table with a non-unique collating sequence, and comparing the first character string and the second character string according to a second dictionary sort order table with a unique collating sequence.

In another aspect of the present invention, there is provided a data processing system having computer readable code for directing said data processing system to impliment the method of as described in the previous paragraph.

In yet another aspect of the invention, there is provided a method of sorting an input data list containing a list of character strings including (a) selecting one of the character strings as a first character string and another of the character strings as a second character string, (b) comparing the first character string and the second character string according to a first dictionary sort order table with a non-unique collating sequence, (c) comparing the first character string and the second character string according to a second dictionary sort order table with a unique collating sequence, (d) repeating steps (a) to (c) iteratively by selecting different pairs of first and second character strings in accordance with a sorting algorithm, (e) using the results of step (b) to sort the character strings into equivalence classes, and (f) using the results of step (c) to sort the character strings within their equivalence classes.

In yet another aspect of the present invention, there is provided a data processing system having computer readable code for directing said data processing system to impliment the method of as described in the previous paragraph.

5 In yet another aspect of the invention, there is provided a computer readable medium embodying instructions for performing a method of ordering a first character string and a second character string, the method including comparing the first character string and the second character string according to a first dictionary sort order table with a non-unique collating sequence, and comparing the first character string and the
10 second character string according to a second dictionary sort order table with a unique collating sequence.

In yet another aspect of the invention, there is provided a computer readable medium embodying computer readable instructions for performing a method of sorting an input data list containing a list of character strings, the method including (a) selecting one of the character strings as a first character string and another of the character
15 strings as a second character string, (b) comparing the first character string and the second character string according to a first dictionary sort order table with a non-unique collating sequence, (c) comparing the first character string and the second character string according to a second dictionary sort order table with a unique collating
20 sequence, (d) repeating steps (a) to (c) iteratively by selecting different pairs of first and second character strings in accordance with a sorting algorithm, (e) using the results of step (b) to sort the character strings into equivalence classes, and (f) using the results of step (c) to sort the character strings within their equivalence classes.

25 A better understanding of these and other embodiments of the present invention can be obtained with reference to the following drawings and description of the preferred embodiments.

Brief Description of the Drawings

30 An exemplary embodiment of the present invention will now be described with reference to the accompanying drawings, in which:

Figure 1 illustrates a portion of the ASCII character set widely used in computer systems;

Figure 2 illustrates a dictionary sort order table with a unique collating sequence;

Figure 3 illustrates a dictionary sort order table with a non-unique collating
5 sequence;

Figure 4 illustrates a system including a comparison module according to the present invention; and

Figures 5 and 6 illustrate a method according to the present invention.

10 Detailed Description of an Exemplary Embodiment

Reference is first made to Figure 1. Alphabetic characters are represented in computer memory by numbers defined by a character set. A common example of a character set is the ASCII character set. The ASCII character set uses 8 bit numbers
15 between 0 and 255 to represent alpha-numeric characters, control characters and other characters. Other character sets may have more than 256 characters, requiring the use of numbers with more than 8 bits. Each character in the character set has a unique number, which may be referred to as the character's code point. Figure 1 illustrates a portion of the ASCII character set 20.

20 ASCII character set 20 includes characters for the Roman letters that are generally used for the English language and other languages. The alphabet of most languages is typically presented in a standardized dictionary sort order. This dictionary sort order defines the weight of each letter in the alphabet to be used when sorting letters in the alphabet. In the dictionary sort order, a letter with a lower weight precedes
25 a letter with a higher weight. The dictionary sort order for a particular alphabet can depend on the particular language and, in some cases, the geographic territory in question. In some languages a single letter may have more than one representation. For example, in English, each letter has an uppercase and a lowercase form. In the dictionary sort order of the English alphabet, the uppercase and a lowercase form of
30 each letter are given the same weight.

The order of characters in a computer character set, such as ASCII character set 20, will typically be different from the dictionary sort order for the letters that are included in the character set. To sort the characters in the computer character set consistently with the dictionary sort order for the alphabet in use, computer programs use dictionary sort order tables that provide a mapping between the character code points in the character set and the letter weights in the dictionary set order. Known 5 dictionary sort order tables may have a unique collating sequence or a non-unique collating sequence.

Figure 2 illustrates a dictionary sort order table 22 with a unique collating 10 sequence. In a dictionary sort order table with a unique collating sequence each character in the computer character set is assigned a unique collating weight based on the weights assigned to corresponding letters in the dictionary sort order of the relevant language. Since all characters are assigned unique weights, different forms of the same letter are often assigned consecutive or effectively consecutive weights. Typically, 15 the uppercase form of an English letter is considered to have a lower weight than its corresponding lowercase form. Dictionary sort order table 22 follows this rule, but could follow the opposite rule. In dictionary sort order table 22, the uppercase "D" is assigned a weight of 69 and the lowercase "d" is assigned a higher weight of 70.

A single word, such as "chad" may be written in various combinations of 20 uppercase and lowercase letters. In a computer, such combinations are usually referred to a character strings. Two different character strings corresponding to the word "chad" are "CHAD" and "Chad". When character strings are sorted using dictionary sort order table 22 with a unique collating sequence, uppercase and lowercase forms of the same letter have different weights. By comparing successive 25 pairs of letter in a pair of strings, one of the strings may be determined to have a lower collating weight, unless the strings are identical. For example, the character string "CHAD" can be determined to have a lower collating weight than the character string "Chad". Initially, the first letter of each string is compared. Each string begins with an uppercase C so these letters have equal weight (144). Then the next letter of each 30 string is compared. Since the uppercase H in "CHAD" has a lower weight (154) than

the lowercase h in Chad (which has a weight of 155), the character string "CHAD" has a lower collating weight than the character string Chad according to dictionary sort order table 22.

As noted above, the character strings "CHAD" and Chad (as well as "chad", etc.) are typically considered to be the same word in the English language. These character strings can be said to be in an "equivalence class". By sorting them with dictionary sort order table 22, the two different character strings have been distinguished and sorted, but the fact that they are in the same equivalence class (i.e. they are the same English word) has been lost. This type of sort may be referred to as a "case-sensitive" sort.

Figure 3 illustrates a dictionary sort order table 24 with a non-unique collating sequence. In a dictionary sort order table with a non-unique collating sequence each character corresponding to the same letter is assigned the same collating weight, based on the weight of the letter in the dictionary sort order for the language in use. Accordingly, both the uppercase A and lowercase a are assigned the same collating weight in dictionary sort order table 24.

When the character strings "CHAD" and Chad are sorted using dictionary sort order table 24, they are determined to be in the same equivalence class, since each corresponding pair of letters in both strings has the same weight. These and other character strings such as "chad", cHad, chAD, etc) are all in the same equivalence class and/ dictionary sort order table 24 does not distinguish between them. As a result, they could be sorted in any arbitrary order. As noted above, in many cases it preferable to list these strings in the order "CHAD", Chad. This may be desirable to provide an aesthetically pleasing list for a report, etc. In other cases, the opposite order may be preferable.

By sorting these character strings using dictionary sort order table 23 with a non-unique collating sequence, the fact that both character strings "CHAD" and Chad are the same English word and in the same equivalence class is recognized but the desired sort order of the character strings (within the equivalence class) themselves is ignored. This type of sort may be referred to as a "case-insensitive" sort.

Reference is next made to Figure 4 which illustrates a system 40 that allows different character strings to be sorted in a desirable sequence, including character strings that represent the same word.

System 40 includes a sorting module 44, a dictionary sort order table 46 with a non-unique collating sequence and a dictionary sort order table 48 with a unique collating sequence. Sorting module 44 also includes a comparison module 52. Alternatively, comparison module 52 may be separate from sorting module 44 and may include a function call to allow sorting module 44 to access comparison module 52.

In this exemplary embodiment of the present invention, dictionary sort order table 46 is identical to dictionary sort order table 24 (Figure 3) and dictionary sort order table 48 is identical to dictionary sort order table 22 (Figure 2). Dictionary sort order table 46 is chosen to allow equivalence classes of English language character strings to be distinguished from one another, without providing any distinction between character strings that are in the same equivalence class. Dictionary sort order table 48 is chosen to allow character strings within an equivalence class to be distinguished from one another. In other embodiments of the invention, other dictionary sort order tables may be used depending on the dictionary sort order for the language in use or on the specific distinctions to be made between equivalence classes and elements within equivalence classes.

System 40 may be used to provide data sorting services to a calling program 42. Alternatively, system 40 may be part of a database management system (not shown) and may provide data sorting services to the database management system. Typically, system 40 will be installed in a computer system 56. Computer system 56 may include more than one computer, storage devices and other elements. The components of system 40 may be distributed in different parts of computer system 56.

Sorting module 44 is configured to receive an unsorted input data set 60 from calling program 42. Input data set 60 may be any type of character string data in which any particular datum may include different forms of letters or other symbols that could be given an equal weight in a dictionary sort order, but for which a preferred order of sorting may be defined. An exemplary input data set 60 comprises the five data:

character strings "chad", "Alpha", "CHAD", "delta", and "Chad". This exemplary input data set 60 will be used to explain the operation of system 40.

5 Sorting module 44 sorts the data in input data set 60 into their equivalence classes according to dictionary sort order table 46 and within their equivalence classes according to dictionary sort order table 48 to produce an output data set 62. Output data set 62 is returned to calling program 42.

10 To sort input data set 60 to produce output data set 62, sorting module 44 may implement any sorting algorithm such as bubble sort, quick sort, insertion sort, etc. During each iteration of the sorting algorithm, sorting module 44 passes two data from input data set 60 to comparison module 52. In response, comparison module 52 returns a first return value R1 to sorting module 44. The first return value R1 is based on a comparison of the two datum based on dictionary sort order table 46. If the two datum are equal (i.e. they are in the same equivalence class) when compared according to dictionary sort order table 46, comparison module 52 also returns a
15 second return value R2 to sorting module 44. The second return value R2 is based on a comparison of the two datum based on dictionary sort order table 48. During successive iterations of the sorting algorithm, sorting module 44 will receive a series of return values R1 and R2 from comparison module 52.

20 Sorting module 44 sorts the data in input data set 60 into a single list in which (i) equivalence classes are sorted and grouped together based on the series of return values R1 and (ii) data within equivalence classes are ordered into a desirable order based on the series of return values R2. The sorted data forms output data set 62, which is returned to the calling program 42 when input data set 60 has been fully sorted.

25 Reference is next made to Figures 5 and 6. Method 100 illustrates the operation of comparison module 52. Method 100 will be explained using an example in which two of the data in input data set 60, character strings "CHAD" and "Chad", are compared to each other.

Method 100 begins in step 102 in which sorting module 44 receives a pair of data D1 and D2 from calling program 42. For example, D1 may be character string "CHAD" and D2 may be character string "Chad".

5 Method 100 proceeds to step 104, in which a current position counter POS is set to 0.

Method 100 proceeds to step 106. In step 106, a variable N1 is set equal to the weight of the character in the current position of datum D1, according to dictionary sort order table 46, which has a non-unique collating sequence. A skilled person will understand that the characters in a character string having a length of M characters are typically referred to as being in positions 0, 1, 2, ..., M-1. Accordingly, when the current position counter equals 0, the first character of the character string is at the current position. Alternatively, the current position counter POS could be initialized to 1 in step 104 and the positions of each character string may be numbered 1, 2, 3, ..., M.

10 The character in the current position of datum D1 is "C" and N1 is thus equal to 93 (See Figure 3). In addition, a variable N2 is set equal to the weight of the character in the current position of datum D1. The character in the current position of datum D2 is "C" and N2 is thus also set to 93.

Method 100 next proceeds to decision step 108, in which the values of N1 and N2 are compared. If N1 is equal to N2, then method 100 proceeds to decision step 110. If N1 is not equal to N2, then method 100 proceeds to step 126.

20 From decision step 110, if the character at the current position of datum D1 is the last character of datum D1 or if the character at the current position of datum D2 is the last character of datum D2, then method 100 proceeds to decision step 114. Otherwise, there is at least one more character in each of datum D1 and datum D2 and method 100 proceeds to step 112.

25 In step 112, the current position pointer POS is incremented and method 100 returns to step 106.

In the present example, method 100 will loop through steps 106, 108 and 110 four times and step 112 three times while the successive characters in datum D1 ("CHAD") and datum D2 ("Chad") are compared. Since variables N1 and N2 are set in

step 106 using dictionary sort order table 46, with has a non-unique collating sequence with uppercase and lowercase forms of each letter having the same weight, method 100 will reach the ends of datum D1 and D2 on the fourth iteration through step 110. At that point, method 100 will proceed to step 114.

5 In decision step 114, the lengths of datum D1 and D2 are compared. If their lengths are equal, then method 100 proceeds to step 116. Otherwise, method 100 proceeds to decision step 120.

10 In step 116, return value R1 is set to EQ, indicating that data D1 and D2 are members of the same equivalence class according to dictionary sort order table 46. Data D1 and D2 will be in the same equivalence class if they have the same number of characters and if each corresponding letter of each datum D1 and D2 have the same weight according to dictionary sort order table 46. Method 100 proceeds to step 140 (Figure 6).

15 In the present example, method 100 will proceeds through step 116 to step 140, since datum D1 and datum D2 are of equal length.

 From decision step 120, method 100 proceeds to step 122 if the length of datum D1 is less than the length of datum D2. If the length of datum D1 is longer than the length of datum D2, then method 100 proceeds to step 124.

20 In step 122, return value R1 is set to "D1", indicating that datum D1 has a lower weight than datum D2. Method 100 then proceeds to step 132.

 In step 124, return value R2 is set to "D2", indicating that datum D2 has a lower weight than datum D1. Method 100 then proceeds to step 132.

25 Step 114, 116, 120 and 122 implement a rule that if one of the datum is longer than the other, but no difference in the weight of corresponding character is found in any iteration of step 108, then the shorter datum is deemed to have a lower collating weight. In another embodiment, the longer datum may be deemed to have a lower collating weight. In another embodiment, differences in the length of data D1 and D2 may be ignored and method 100 may proceed directly from step 110 to step 116 if the end of datum D1 or D2 has been reached. In such an embodiment, steps 114, 120 and
30 122 would not exist.

In step 126, the weights N1 and N2 of the characters in the current position of data D1 and D2 are compared. If N1 is less than N2, then method 100 proceeds to step 128. If N2 is greater than N1, then method 100 proceeds to step 130.

5 In step 128, return value R1 is set to "D1", indicating that datum D1 has a lower weight than datum D2, when they are compared according to dictionary sort order table 46. Method 100 then proceeds to step 132.

In step 130, return value R1 is set to "D2", indicating that datum D2 has a lower weight than datum D1, when they are compared according to dictionary sort order table 48. Method 100 then proceeds to step 132.

10 In step 132, method 100 returns return value R1 to calling program 42 and then ends.

Reference is next made to Figure 6. If method 100 reaches step 140, then data D1 and D2 are equal when compared according to dictionary sort order table 46 and they have the same length. In the following steps, data D1 and D2 are compared
15 according to dictionary sort order table 48, which has a unique collating sequence. This allows uppercase and lowercase forms of the same letter to be distinguished and allows character strings within the same equivalence class to be ordered based on the unique collating weights defined in dictionary sort order table 48.

In step 140, current position counter POS is set to 0.

20 Method 100 proceeds to step 142. In step 142, variable N1 is set equal to the weight of the character in the current position of datum D1, according to dictionary sort order table 48. The character in the current position of datum D1 is an uppercase "C" and N1 is thus set equal to 144. Variable N2 is set equal to the weight of the character in the current position of datum D2. The character in the current position of datum D2
25 is also an uppercase "C" and N2 is also set to 144.

Method 100 next proceeds to decision step 144, in which the values of N1 and N2 are compared. If N1 is equal to N2, then method 100 proceeds to decision step 146. If N1 is not equal to N2, then method 100 proceeds to step 152.

30 From decision step 144, if the character at the current position of datum D1 is the last character of datum D1 or if the character at the current position of datum D2 is

the last character of datum D2, then method 100 proceeds to step 150. Otherwise, there is at least one more character in each of datum D1 and datum D2 and method 100 proceeds to step 148.

In step 148, the current position pointer POS is incremented and method 100
5 returns to step 142.

In step 150, return value R1 is set to EQ, indicating that data D1 and D2 are equal according to dictionary sort order table 46. Data D1 and D2 will be equal if each corresponding pair of letters in each of them is the same form (uppercase or lowercase) of the same letter. Method 100 proceeds to step 158.

10 In the present example, method 100 will loop through steps 142 and 144 twice and steps 146 and 148 once while the successive characters in datum D1 ("CHAD") and datum D2 ("Chad") are compared. Variables N1 and N2 are set in step 142 using dictionary sort order table 48, which has a unique collating sequence with uppercase and lower case forms of each letter having distinct weights. When the position counter
15 is incremented to 1, variables N1 and N2 will be set based on the second character in datum D1 and datum D2, respectively. The second character in datum D1 is an uppercase "H" and the value of N1 is set to 154. The second character of datum D2 is a lowercase "h" so the value of N2 is set to 155. When method 100 reaches step 144 for the second time, method 100 will proceed to step 152, since N1 will not be equal to
20 N2.

In step 152, the weights N1 and N2, according to dictionary sort order table 48, of the characters in the current position of data D1 and D2 are compared. If N1 is less than N2, then method 100 proceeds to step 154. If N2 is greater than N1, then method 100 proceeds to step 156.

25 In step 128, return value R2 is set to "D1", indicating that datum D1 has a lower weight than datum D2, according to dictionary sort order table 58. Method 100 then proceeds to step 158.

In step 156, return value R2 is set to "D2", indicating that datum D2 has a lower weight than datum D1, according to dictionary sort order table 58. Method 100 then
30 proceeds to step 158.

In step 158, method 100 returns return values R1 and R2 to calling program 42. Method 100 then ends.

Return value R1 returned by method 100 to calling program 42 indicates whether, when data D1 and D2 passed to method 100 in step 102 are compared according to dictionary sort order table 46, (i) datum D1 has a lower weight than datum D2; (ii) datum D2 has a lower weight than datum D1; or (iii) data D1 and D2 have the same weight and are in the same equivalence.

If return value R1 indicates that data D1 and D2 are in the same equivalence class, then return value R2 indicates whether, when data D1 and D2 are compared according to dictionary sort order table 48, (i) datum D1 has a lower weight than datum D2; (ii) datum D2 has a lower weight than datum D1; or (iii) data D1 and D2 have the same weight. In this exemplary embodiment, when the value of return value is D1 or D2, then the value of return value R2 is not calculated by method 100 and therefore has no meaning.

In an alternative embodiment of the present invention, return value R2 may be calculated regardless of the value of return value R1. To implement this option, method 100 would proceed from step 122, 124, 128 or 130 to step 140, rather than to step 132. Return values R1 and R2 are returned to calling program 42 together in step 158.

Table 1 illustrates the results of method 100 when each combination of the data "chad", Alpha, "CHAD", delta, and "Chad" is passed to method 100 as data D1 and D2 in step 102.

D1	D2	R1	R2
"chad"	"Alpha"	D2	-
"chad"	"CHAD"	EQ	D2
"chad"	"delta"	D1	-
"chad"	"Chad"	EQ	D2
"Alpha"	"CHAD"	D1	-
"Alpha"	"delta"	D1	-

"Alpha"	"Chad"	D1	-
"CHAD"	"delta"	D1	-
"CHAD"	"Chad"	EQ	D1
"delta"	"Chad"	D2	-

Table 1. Return values R1 and R2 from method 100 for combinations of data D1 and D2.

5

Depending on the sorting algorithm implemented in sorting module 44, sorting module may call comparison module 52 and pass it some or all of the combinations of data D1 and D2 set out in Table 1. Sorting module 44 uses return values R1 and R2 from comparison module 52 to organize the character strings in output data set in the order set out in Table 2. Character strings "chad", "Chad" and "CHAD" are listed consecutively, since they are in the same equivalence class. The order of these strings in output data list 62 is controlled by the unique collating sequence defined in dictionary sort order table 48.

10

"Alpha"
"CHAD"
"Chad"
"chad"
"delta"

15

Table 2. Output data set 62

In another embodiment of the present invention, a sorting module may be configured to provide an output data set in which duplicate data in the same equivalence class have been eliminated so that only one datum from each equivalence class, according to dictionary sort order table 46, is included. Such a sorting module would use return values R1 to identify duplicate members of a single equivalence class.

20

The sorting module may be configured to select one member of the equivalence class for inclusion in the output data on any basis. The one member may be selected at random, based on the order in which the members of the equivalence class appear in the input data set, or return values R2 may be used to select the member of the
5 equivalence class with the lowest (or highest) collating weight according to dictionary sort order table 46.

An embodiment of the present invention based on sorting English language words or character strings has been described. The invention may be modified by a skilled person to be used to sort word or character strings in any other language by
10 configuring dictionary sort order tables 46 and 48.

In addition, the present invention may be modified to provide multi-level sorting between character strings formed of symbols or other indicia by similarly configuring dictionary sort order tables 46 and 48.

It will be appreciated that variations of some elements are possible to adapt the
15 invention for specific conditions or functions. The concepts of the present invention can be further extended to a variety of other applications that are clearly within the scope of this invention. Having thus described the present invention with respect to a preferred embodiment as implemented, it will be apparent to those skilled in the art that many modifications and enhancements are possible to the present invention without
20 departing from the basic concepts as described in the preferred embodiment of the present invention. Therefore, what is intended to be protected by way of letters patent should be limited only by the scope of the following claims.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A method of ordering a first character string and a second character string
5 comprising:

(a) comparing the first character string and the second character string according to a first dictionary sort order table with a non-unique collating sequence; and

(b) comparing the first character string and the second character string according to a second dictionary sort order table with a unique collating sequence.
10

2. The method of claim 1 wherein step (a) is performed to determine whether:

(i) the first character string has a lower collating weight than the second character string according to the non-unique collating sequence;

(ii) the second character string has a lower collating weight than the first collating sequence according to the non-unique collating sequence; or
15

(iii) the first and second character strings are in a single equivalence class according to the non-unique collating sequence,

and step (b) is performed to determine whether:

(i) the first character string has a lower collating weight than the second character string according to the unique collating sequence;
20

(ii) the second character string has a lower collating weight than the first character string according to the unique collating sequence; or

(iii) the first and second characters strings are equal according to the unique collating sequence.
25

3. The method of claim 1 or 2 wherein step (b) if performed only is the result of step (a) is that the first and second character strings are in the single equivalence class.

4. The method of claim 1, 2 or 3 wherein step (a) is performed by:

(A) selecting a first comparison character from the first character string wherein the first comparison character is the first character in the first character string;

(B) selecting a second comparison character from the second character string, wherein the second comparison character is the first character in the second character string;

(C) comparing the first comparison character and the second comparison character to determine if:

(i) the first comparison character and the second comparison character do not have an equal weighting according to the first dictionary sort order table;

(ii) the first comparison character is the last character of the first character string;

(iii) the second comparison character is the last character of the second character string; or

(iv) the first comparison character is the last character of the first character string and the second comparison character is the last character of the second character string; and

(D) repeating step (C) until at least one of conditions set out in parts (i), (ii), (iii) and (iv) of step (C) occurs, wherein during each such repetition, the first comparison character is the next successive character of the first character string and the second comparison character is the next successive character of the second character string.

5. The method of claim 4 wherein step (a) further includes:

(E) if during step (C), the first comparison character has a lower weighting than the than the second comparison character according to the first dictionary sort order table, then determining that the first character string is ordered before the second character string;

(F) if during step (C), the second comparison character has a lower weighting than the first comparison character according to the first dictionary sort order table,

then determining that the second character string is ordered before the first character string; and

(G) if during step (C), the characters being compared have an identical weighting and the first character string ends before the second character string, then
5 determining that the first character string is ordered before the second character string;

(H) if during step (C), the characters being compared have an identical weighting and the second character string ends before the first character string, then
determining that the second character string is ordered before the first character string;
and

10 (I) if during step (C), the characters being compared have an identical weighting and the first character string and the second character string have an identical length, then determining that the first character string and the second character string are in the single equivalence class.

15 6. The method of any of claims 1 to 5 wherein step (b) includes:

(A) selecting a first comparison character from the first character string
wherein the first comparison character is the first character in the first character string;

(B) selecting a second comparison character from the second character
string, wherein the second comparison character is the first character in the second
20 character string;

(C) comparing the first comparison character and the second comparison
character to determine if:

(i) the first comparison character and the second comparison
character do not have an equal weighting according to the second dictionary sort order
25 table; or

(ii) the first comparison character is the last character of the first
character string and the second comparison character is the last character of the
second character string; and

(D) repeating step (C) until at least one of conditions set out in parts (i) or (ii)
30 of step (C) occurs, wherein during each such repetition, the first comparison character

is the next successive character of the first character string and the second comparison character is the next successive character of the second character string.

7. The method of claim 6 wherein step (b) further includes:

5 (E) if during step (C), the first comparison character has a lower weighting than the than the second comparison character according to the second dictionary sort order table, then determining that the first character string is ordered before the second character string within the single equivalence class;

10 (F) if during step (C), the second comparison character has a lower weighting than the first comparison character according to the first dictionary sort order table, then determining that the second character string is ordered before the first character string within the single equivalence class; and

15 (G) if during step (C), the characters being compared have an identical weighting and the first character string ends before the second character string, then determining that the first character string is ordered before the second character string within the single equivalence class;

20 (H) if during step (C), the characters being compared have an identical weighting and the second character string ends before the first character string, then determining that the second character string is ordered before the first character string within the single equivalence class; and

(I) if during step (C), the characters being compared have an identical weighting and the first character string and the second character string have an identical length, then determining that the first character string and the second character string are not distinct from one another within the single equivalence class.

25

8. The method of any one of claims 1 to 5 further comprising:

(c) outputting the results of steps (a) and (b).

9. The method of any one of claims 1 to 5 further comprising:

(a.1) before step (a), receiving the first and second character strings from an invoking program; and

(c) passing the results of steps (a) and (b) to the invoking program.

5 10. The method of any one of claims 1 to 6, wherein the unique collating sequence is case sensitive.

11. The method of any one of claim 1 to 7, wherein the non-unique collating sequence is case insensitive.

10

12. A method of sorting an input data list containing a list of character strings comprising:

(a) selecting one of the character strings as a first character string and another of the character strings as a second character string;

15

(b) comparing the first character string and the second character string according to a first dictionary sort order table with a non-unique collating sequence;

20

(c) comparing the first character string and the second character string according to a second dictionary sort order table with a unique collating sequence;

(d) repeating steps (a) to (c) iteratively by selecting different pairs of first and second character strings in accordance with a sorting algorithm;

(e) using the results of step (b) to sort the character strings into equivalence classes; and

25

(f) using the results of step (c) to sort the character strings within their equivalence classes.

13. The method of claim 12 wherein step (b) is performed to determine whether:

30

(i) the first character string has a lower collating weight than the second character string according to the non-unique collating sequence;

(ii) the second character string has a lower collating weight than the first character string according to the unique collating sequence; or

(iii) the first and second character strings are in a same equivalence class according to the non-unique collating sequence,

5 and step (c) is performed to determine whether:

(i) the first character string has a lower collating weight than the second character string according to the unique collating sequence;

(ii) the second character string has a lower collating weight than the first character string according to the unique collating sequence; or

10 (iii) the first and second characters strings are equal according to the unique collating sequence.

14. The method of either of claims 12 or 13 wherein, during each iteration in step (d), step (c) is performed only if the result of step (b) is that the first and second character strings are in the same equivalence class.

15. The method of any one of claims 12 to 14 further comprising:

(a.1) before step (a), receiving the input data list from a calling program; and

(g) passing the sorted character strings to the calling program as an output data list.

16. The method of any one of claims 12 to 15, wherein the unique collating sequence is case sensitive.

25 17. The method of any one of claim 12 to 16, wherein the non-unique collating sequence is case insensitive.

18. A computer readable medium embodying computer-readable instructions for directing a data processing system to perform a method of ordering a first character string and a second character string, the method comprising:

(a) comparing the first character string and the second character string according to a first dictionary sort order table with a non-unique collating sequence; and

(b) comparing the first character string and the second character string according to a second dictionary sort order table with a unique collating sequence.

5

19. The computer readable medium of claim 18 wherein step (a) is performed to determine whether:

(i) the first character string has a lower collating weight than the second character string according to the non-unique collating sequence;

10 (ii) the second character string has a lower collating weight than the first collating sequence according to the non-unique collating sequence; or

(iii) the first and second character strings are in a single equivalence class according to the non-unique collating sequence,

and step (b) is performed to determine whether:

15 (i) the first character string has a lower collating weight than the second character string according to the unique collating sequence;

(ii) the second character string has a lower collating weight than the first character string according to the unique collating sequence; or

20 (iii) the first and second characters strings are equal according to the unique collating sequence.

20. The computer readable medium of claim 18 or 19 wherein step (b) if performed only is the result of step (a) is that the first and second character strings are in the single equivalence class.

25

21. The computer readable medium of claim 18, 19 or 20 wherein step (a) is performed by:

(A) selecting a first comparison character from the first character string wherein the first comparison character is the first character in the first character string;

(B) selecting a second comparison character from the second character string, wherein the second comparison character is the first character in the second character string;

(C) comparing the first comparison character and the second comparison character to determine if:

(i) the first comparison character and the second comparison character do not have an equal weighting according to the first dictionary sort order table;

(ii) the first comparison character is the last character of the first character string;

(iii) the second comparison character is the last character of the second character string; or

(iv) the first comparison character is the last character of the first character string and the second comparison character is the last character of the second character string; and

(D) repeating step (C) until at least one of conditions set out in parts (i), (ii), (iii) and (iv) of step (C) occurs, wherein during each such repetition, the first comparison character is the next successive character of the first character string and the second comparison character is the next successive character of the second character string.

22. The computer readable medium of claim 21 wherein step (a) further includes:

(E) if during step (C), the first comparison character has a lower weighting than the than the second comparison character according to the first dictionary sort order table, then determining that the first character string is ordered before the second character string;

(F) if during step (C), the second comparison character has a lower weighting than the first comparison character according to the first dictionary sort order table, then determining that the second character string is ordered before the first character string; and

(G) if during step (C), the characters being compared have an identical weighting and the first character string ends before the second character string, then determining that the first character string is ordered before the second character string;

(H) if during step (C), the characters being compared have an identical weighting and the second character string ends before the first character string, then determining that the second character string is ordered before the first character string; and

(I) if during step (C), the characters being compared have an identical weighting and the first character string and the second character string have an identical length, then determining that the first character string and the second character string are in the single equivalence class.

23. The computer readable medium of any of claims 18 to 22 wherein step (b) includes:

(A) selecting a first comparison character from the first character string wherein the first comparison character is the first character in the first character string;

(B) selecting a second comparison character from the second character string, wherein the second comparison character is the first character in the second character string;

(C) comparing the first comparison character and the second comparison character to determine if:

(i) the first comparison character and the second comparison character do not have an equal weighting according to the second dictionary sort order table; or

(ii) the first comparison character is the last character of the first character string and the second comparison character is the last character of the second character string; and

(D) repeating step (C) until at least one of conditions set out in parts (i) or (ii) of step (C) occurs, wherein during each such repetition, the first comparison character

is the next successive character of the first character string and the second comparison character is the next successive character of the second character string.

24. The computer readable medium of claim 23 wherein step (b) further includes:

5 (E) if during step (C), the first comparison character has a lower weighting than the than the second comparison character according to the second dictionary sort order table, then determining that the first character string is ordered before the second character string within the single equivalence class;

10 (F) if during step (C), the second comparison character has a lower weighting than the first comparison character according to the first dictionary sort order table, then determining that the second character string is ordered before the first character string within the single equivalence class; and

15 (G) if during step (C), the characters being compared have an identical weighting and the first character string ends before the second character string, then determining that the first character string is ordered before the second character string within the single equivalence class;

20 (H) if during step (C), the characters being compared have an identical weighting and the second character string ends before the first character string, then determining that the second character string is ordered before the first character string within the single equivalence class; and

(I) if during step (C), the characters being compared have an identical weighting and the first character string and the second character string have an identical length, then determining that the first character string and the second character string are not distinct from one another within the single equivalence class.

25

25. The computer readable medium of any one of claims 18 to 22 further comprising:

(c) outputting the results of steps (a) and (b).

26. The computer readable medium of any one of claims 18 to 22 further comprising:

(a.1) before step (a), receiving the first and second character strings from an invoking program; and

5 (c) passing the results of steps (a) and (b) to the invoking program.

27. The computer readable medium of any one of claims 18 to 23, wherein the unique collating sequence is case sensitive.

10 28. The computer readable medium of any one of claim 18 to 23, wherein the non-unique collating sequence is case insensitive.

29. A computer readable medium embodying computer readable instructions for directing a data processing system to perform a method of sorting an input data list containing a list of character strings, the method comprising:

15 (a) selecting one of the character strings as a first character string and another of the character strings as a second character string;

(b) comparing the first character string and the second character string according to a first dictionary sort order table with a non-unique collating sequence;

20 (c) comparing the first character string and the second character string according to a second dictionary sort order table with a unique collating sequence;

(d) repeating steps (a) to (c) iteratively by selecting different pairs of first and second character strings in accordance with a sorting algorithm;

25 (e) using the results of step (b) to sort the character strings into equivalence classes; and

(f) using the results of step (c) to sort the character strings within their equivalence classes.

30

30. The computer readable medium of claim 29 wherein step (b) is performed to determine whether:

(i) the first character string has a lower collating weight than the second character string according to the non-unique collating sequence;

(ii) the second character string has a lower collating weight than the first character string according to the unique collating sequence; or

(iii) the first and second character strings are in a same equivalence class according to the non-unique collating sequence,

and step (c) is performed to determine whether:

(i) the first character string has a lower collating weight than the second character string according to the unique collating sequence;

(ii) the second character string has a lower collating weight than the first character string according to the unique collating sequence; or

(iii) the first and second characters strings are equal according to the unique collating sequence.

31. The computer readable medium of either of claims 29 or 30 wherein, during each iteration in step (d), step (c) is performed only if the result of step (b) is that the first and second character strings are in the same equivalence class.

32. The computer readable medium of any one of claims 29 to 31 further comprising:

(a.1) before step (a), receiving the input data list from a calling program; and

(g) passing the sorted character strings to the calling program as an output data list.

33. The computer readable medium of any one of claims 29 to 32, wherein the unique collating sequence is case sensitive.

34. The computer readable medium of any one of claim 29 to 33, wherein the non-unique collating sequence is case insensitive.

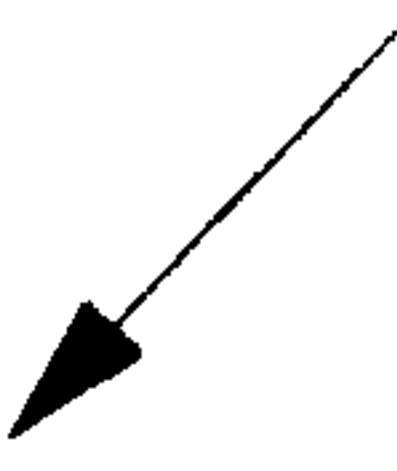
5 35. A data processing system having computer readable code for directing said data processing system to impliment the method of any of claims 1 to 11.

36. A data processing system having computer readable code for directing said data processing system to impliment the method of any of claims 12 to 17.

10

Figure 1

20



Character	Code Point
...	
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
...	
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
....	

Figure 2

Character	Collating Weight	ASCII Code Points
...		
A	140	65
B	142	66
C	144	67
D	146	68
E	148	69
F	150	70
G	152	71
H	154	72
I	156	73
J	158	74
...		
a	141	97
b	143	98
c	145	99
d	147	100
e	149	101
f	151	102
g	153	103
h	155	104
i	157	105
j	159	106
....		

22



Figure 3

Character	Collating Weight	ASCII Code Points
...		
A	91	65
B	92	66
C	93	67
D	94	68
E	95	69
F	96	70
G	97	71
H	98	72
I	99	73
J	100	74
...		
a	91	97
b	92	98
c	93	99
d	94	100
e	95	101
f	96	102
g	97	103
h	98	104
i	99	105
j	100	106
....		

24



Figure 4

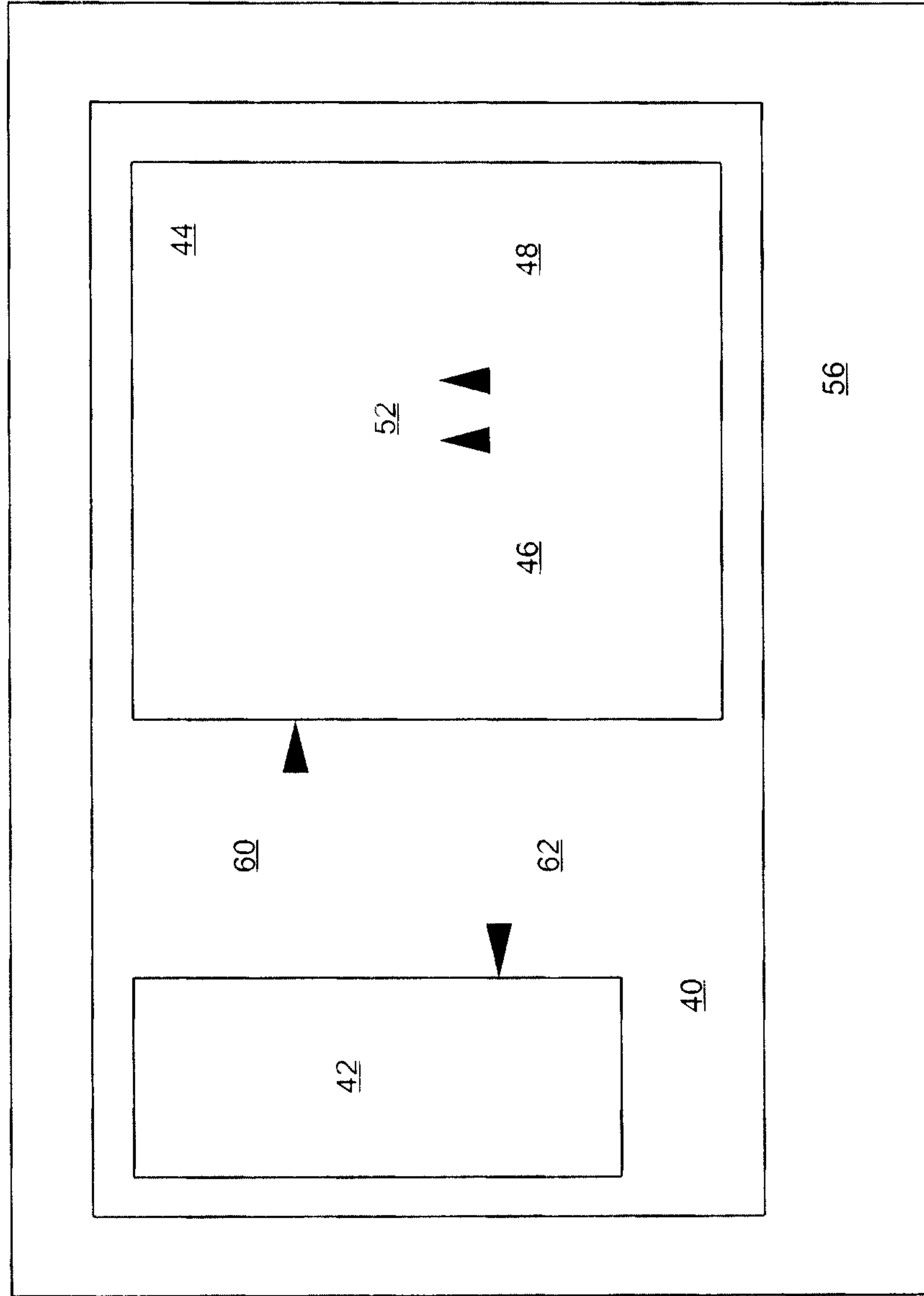


Figure 5

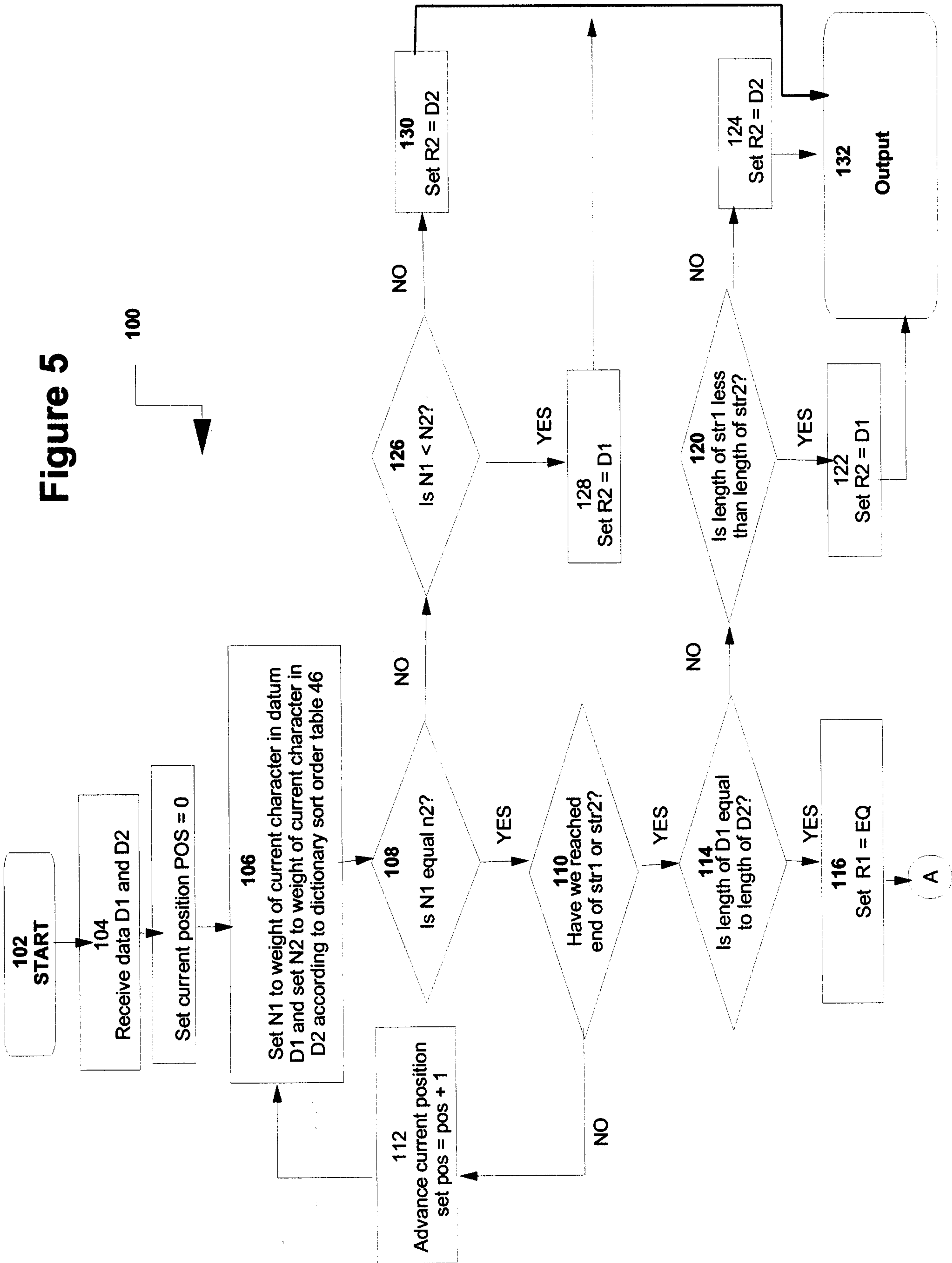


Figure 6

