A Simple Algorithm for Sorting Chemical Names

GLORIA W. SAGE* and ANTHONY B. LAMACCHIA

Syracuse Research Corporation, Syracuse, New York 13210

Received March 28, 1983

A simple algorithm for sorting chemical names has been developed that is based on the recognition of hyphens, parentheses, and blanks within the name. The algorithm is designed for sorting computerized lists of chemical names at facilities where italic type, greek letters, and other special characters are not available or are too burdensome to use. Nevertheless, the algorithm sorts names containing these special characters. The algorithm and the computer program for its implementation are described.

INTRODUCTION

Many chemists who keep information on large numbers of chemicals (e.g., inventories, hazardous chemicals, water solubilities) stored in computer files are interested in alphabetizing these files by chemical name. The utility sorting routines will not alphabetize chemical names in a manner acceptable to chemists because locants in the name are not used in the primary sort. It is therefore necessary to create a sortkey on which to alphabetize the file. The chemical literature is surprisingly devoid of algorithms for sorting chemical names; a search of *Chemical Abstracts* back to 1962 did not uncover a single reference. Chemical Abstracts Service which probably sorts the greatest number of chemical names in the world in producing its indices has a very powerful, but unpublished, algorithm for sorting chemical names. Its algorithm in comprehensive, complex, and appropriate for producing a large, printed index or catalog. It is burdensome and unnecessarily comprehensive and complex for many uses.

The Chemical Abstract Service algorithm is based on the recognition of Greek, italics, and numeric locants as well as punctuation, subscripts, superscripts, and hosts of special character strings. Purely numeric or alphabetic names are identified and sorted by special routines. Alphanumeric and systematic names are also handled by special routines. Nonnumeric locants in chemical names which would be printed in italic, Greek, or boldface letters must be entered into the computer in a special manner. This is frequently a burdensome and unnecessary effort for a chemist who wants his computer file sorted by chemical name. We have devised an algorithm for sorting computrized lists of chemical names, SORTKEY, which is capable of alphabetizing the preponderance of chemical names yet is simple and does not require locants to be entered into the computer in a special way.

The need for an algorithm to sort computerized lists of chemical names arose during the development of the Environmental Fate Data Bases, DATALOG and CHEMFATE.[1] These data bases contain environmental fate information on over 3000 primarily organic compounds. Although the data bases are searched by CAS registry number, chemical names are included in the records for recognition purposes. The name used may be an IUPAC name,[2] CAS Index name,[3] trivial name, or trade name in the case of some pesticides. Inverted forms of nomenclature are generally avoided, and stereoisomers and optical isomers are not individually listed (e.g., 2-butene includes *cis*- and *trans*-2-butene). The names are written in upper-case alphabetic characters, numbers, and punctuation symbols available on any computer terminal. Greek letters are written out in words, and italices are written in normal letters. The SORTKEY algorithm described below was developed to handle the types of chemicals encountered in the Environmental Fate Data Bases. It is quite general. Both upper and lower case as well as special letters (e.g., Greek, italics) can be used. Some chemical names such as those which include locants in square bracket are not treated in the algorithm but can be accommodated without difficulty. The algorithm fails for some sugars and compounds that are stereoisomers or optical isomers. These cases can be readily identified and frequently dealt with satisfactorily. Some possible modification to the algorithm will be discussed later. At this point we do not wish to detract from the simplicity of an algorithm that works in the large majority of cases.

THE ALGORITHM

The SORTKEY algorithm is based on the recognition of hyphens, parentheses, and spaces within a chemical name. Hyphens are used in chemical names to separate Greek, italic, and numeric locants from the rest of the name which is why we write "Dibutyl phthalate" or "di-*n*-butyl phthalate" but not "di-butyl phthalate". Similarly, parentheses clearly identify the group to which a locant applies which is why "1,3-dimethyl-5-(*tert*-butyl)benzene" is preferable to "1,3-dimethyl-5-*tert*-butylbenzene". SORTKEY is based on the fact that hyphens isolate locants which are not used in the primary sort. Recognition of parentheses and blanks are required to handle chemical names where these characters arise (e.g., "4-(*tert*-butyl)phenol", "methyl *tert*-butyl ketone").

The SORTKEY algorithm is used to create a sortkey from the chemical name as follows: reading the chemical name from right to left, locate pairs of hyphens and eliminate the characters in between along with the hyphens. If the number of hyphens is odd, eliminate the hyphen and the characters to the left of the leftmost hyphen. Place the last group of characters eliminated in a separate, secondary sortkey field. If one has a "compound" name that is made up of several words or sets of parentheses, process the hyphens within each word or in between parentheses separately and eliminate the spaces between words or parentheses from the sortkey.

Figure 1a illustrates how the algorithm operates by tracking the series of transformations a chemical name would undertake in forming the sortkey. In the first step, the name fragment between parentheses is processed. Since there is an odd number of hyphens, the characters to the left of the hyphen, namely "2,4", are eliminated along with the hyphen, and the parentheses are removed. In the second step, the "-2-" is eliminated. Finally, the "5-" is eliminated, leaving "chlorodichlorophenoxyphenol" as the sortkey and "5" as the secondary sortkey. SORTKEY has removed locants from the chemical name to create a primary and secondary sortkey. Obviously a third sortkey can be retained containing the second locant, but for most purposes two sortkeys are sufficient.

THE SORTKEY PROGRAM

The SORTKEY program was written to implement the simple chemical sort algorithm. The program is written in Xerox

a) 5-chloro-2-(2,4-dichlorophenoxy)phenol

↓

5-chloro-2-dichlorophenoxyphenol

↓

5-chlorodichlorophenoxyphenol

↓

chlorodichlorophenoxyphenol

b) 5-chloro-2-(2,4-dichlorophenoxy)phenol

↓

5-chloro-2--2,4-dichlorophenoxyphenol

↓

5-chloro-2-dichlorophenoxyphenol

↓

5-chlorodichlorophenoxyphenol

↓

chlorodichlorophenoxyphenol

Figure 1. Application of the SORTKEY algorithm: (a) example of transformations in forming the sortkey; (b) computer implementation of transformations in forming sortkey.

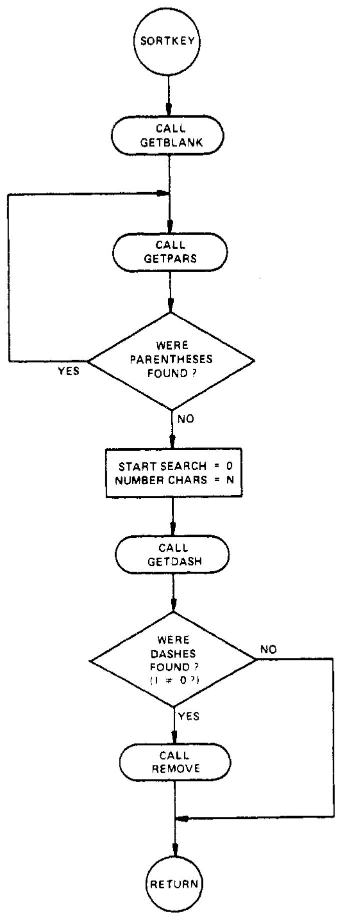

Figure 2. Flowchart of SORTKEY.

FORTRAN IV for the Honeywell Sigma V computer. The program uses several assembly language subroutines written

Table I. Subroutines in SORTKEY and Their Functions

subroutine	function
GETBLANK	replaces embedded blanks in the chemical name with parentheses. The number of characters, *N*, in a chemical name is determined by recognizing two contiguous blanks. For example "ethyl ether" is replaced by "ethyl(ether)", and the number of characters to be processed is 12.
GETPARS	processes nested parentheses. This subroutine determines whether the number of hyphens within parentheses is odd or even. If the number is odd, the left parenthesis is replaced by a hyphen and the right one removed. If the number is even, both parentheses are removed. For example "bis(2-chloroethoxy)-methane" and "2-(dimethylamino)ethanol" would be replaced by "bis-2-chloroethoxy-methane" and "2-dimethylaminoethanol", respectively.
GETDASH	determines the location of hyphens in a name and stores this information *a* in table D. *I* is the running index and the largest "*I*" is the total number of hyphens in the name. The subroutine requires the location for starting the search, startsearch, and the number of characters to be searched, number chars, as input.
REMOVE	processing from right to left, it removes characters between pairs of hyphens as well as the hyphens themselves to create a sortkey. The eliminated characters are *not* replaced with blanks so the sortkey field is contracted. If there is an odd number of hyphens, the characters to the left of the leftmost hyphen as well as the hyphen itself are also removed. The last set of characters removed is put into the secondary sortkey field.

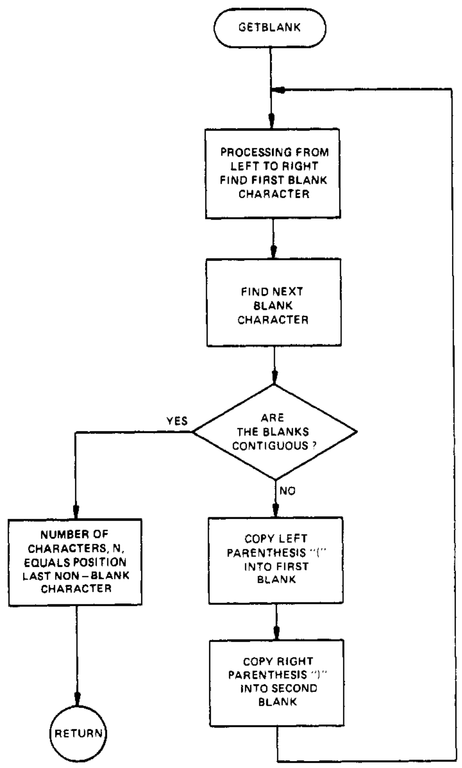
GETBLANK
PROCESSING FROM LEFT TO RIGHT FIND FIRST BLANK CHARACTER
FIND NEXT BLANK CHARACTER
ARE THE BLANKS CONTIGUOUS ?
YES
NO
NUMBER OF CHARACTERS, N, EQUALS POSITION LAST NON-BLANK CHARACTER
COPY LEFT PARENTHESIS "(" INTO FIRST BLANK
COPY RIGHT PARENTHESIS ")" INTO SECOND BLANK
RETURN

Figure 3. Flowchart of Subroutine GETBLANK.

for our computer in order to facilitate character manipulation.

Before initializing SORTKEY, copy the chemical name to a field in which the program will operate to create the sortkey. Set aside another field or a portion of the sortkey field for the

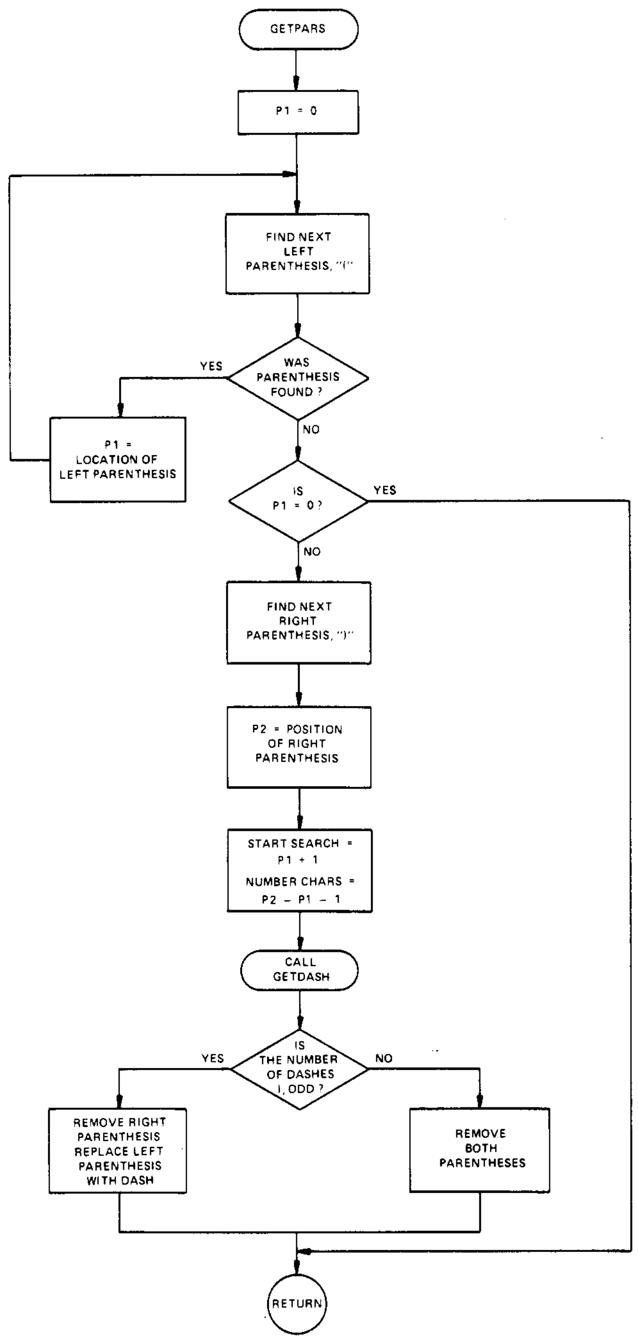
GETPARS
P1 = 0
FIND NEXT LEFT PARENTHESIS, "("
WAS PARENTHESIS FOUND ?
YES
NO
P1 = LOCATION OF LEFT PARENTHESIS
IS P1 = 0 ?
YES
NO
FIND NEXT RIGHT PARENTHESIS, ")"
P2 = POSITION OF RIGHT PARENTHESIS
START SEARCH = P1 + 1
NUMBER CHARS = P2 − P1 − 1
CALL GETDASH
IS THE NUMBER OF DASHES I, ODD ?
YES
NO
REMOVE RIGHT PARENTHESIS REPLACE LEFT PARENTHESIS WITH DASH
REMOVE BOTH PARENTHESES
RETURN

Figure 4. Flowchart of Subroutine GETPARS.

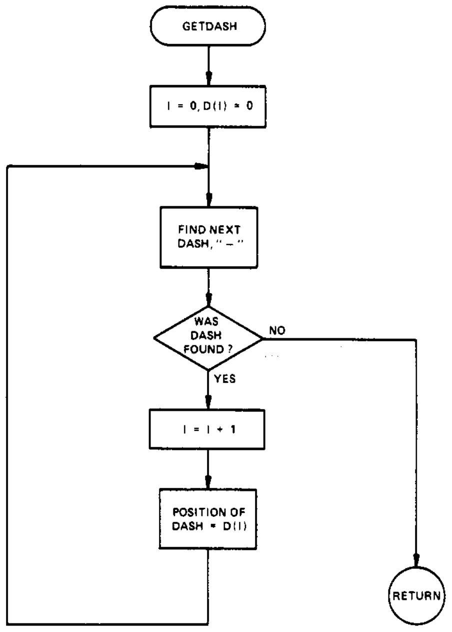

Figure 5. Flowchart of Subroutine GETDASH.

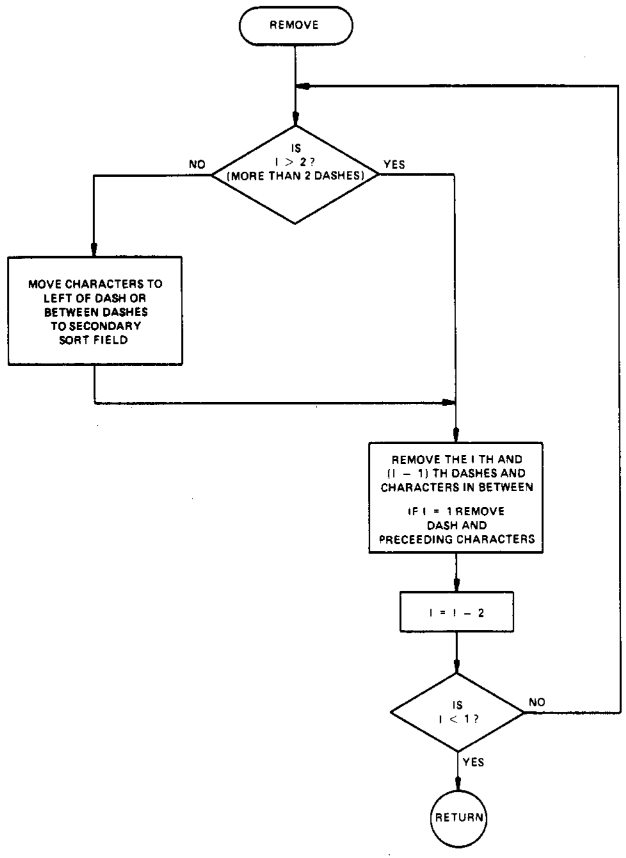
REMOVE
IS I > 2 ? (MORE THAN 2 DASHES)
NO
YES
MOVE CHARACTERS TO LEFT OF DASH OR BETWEEN DASHES TO SECONDARY SORT FIELD
REMOVE THE I TH AND (I − 1) TH DASHES AND CHARACTERS IN BETWEEN
IF I = 1 REMOVE DASH AND PRECEEDING CHARACTERS
I = I − 2
IS I < 1 ?
NO
YES
RETURN

Figure 6. Flowchart of Subroutine REMOVE.

secondary sortkey. If two of the chemical names one wishes to sort are *o*- and *p*-dichlorobenzene, "dichlorobenzene" will end up in the primary sortkey field and "*o*" or "*p*" will be in the secondary sortkey field. After the SORTKEY program is run, the records are sorted on the primary and then the secondary sortkey using the computer's utility sort routine.

A flowchart of the SORTKEY program is shown in Figure 2. The program is composed of several subroutines whose functions are explained in Table I. Flowcharts of the various subroutines appear in Figures 3–6.

In implementing the SORTKEY algorithm for the computer, we have used certain devices to facilitate programming such that the algorithm appears to be different from that which was previously presented. The end result, however, is the same. For instance, embedded blanks are replaced with parentheses, and a left parenthesis is replaced with a "hyphen" when the name fragment within parentheses contains an odd number of hyphens. In following the computer implmentation of SORTKEY, one can see that the chemical name given in the previous example would be transformed as in Figure 1b. In the first step the left parenthesis is replaced by a "hypen" and the right parenthesis removed. In the next two steps, the characters between pairs of parentheses (reading from right to left) are removed. Finally the "5" is moved to the secondary sortkey.

DISCUSSION

SORTKEY is designed primarily to alphabetize the variety of chemical names existing in the Environmental Fate Data Bases. With the exception of a few hyphenated trade names (e.g., Dichlorofop-methyl), over 3300 chemicals with environmental fate related information contained in the data bases were successfully alphabetized. In the case of hyphenated trade names, the hyphen was omitted from the name to avoid problems.

One advantage of the algorithm is that one can identify problem names by inspection. Many classes of compounds and forms of nomenclature are easily accommodated by an extension or modification of the algorithm. For example, when chemical names are written in inverted order (e.g., "acetic acid, chloro-", or "2-propenoic acid, 3-phenyl, 2-methylpropyl ester") one must modify the algorithm as follows: eliminate final hyphens or hyphens preceding commas from the sortkey, replace all commas with blanks, and then apply the standard algorithm. This modified version of SORTKEY could be used with noninverted as well as inverted nomenclature. Another situation where the algorithm can be modified to handle the chemical names is with bridged and fused ring systems. In these cases locants may appear within square bracked (e.g., "1,4-diazobicyclo[2.2.2.]octane" and "indeno[1.2.3-*cd*]-pyrene"). For these chemical names, the locant contained within the square brackets should be eliminated from the sortkey before the standard algorithm is applied.

SORTKEY may fail in situations where names of stereoisomers and optical isomers are to be sorted because there are situations where two locants separated by a hyphen are adjacent (e.g., *cis*-2-butene). These situations can be easily identified. Sometimes the chemical name can be rewritten so the algorithm applies (e.g., "2-butene (cis)" or "2-butene, cis-"). Alternatively, one can search for certain character strings (e.g., cis, dl) which remain at the beginning of the word after the algorithm is applied and remove these locants from the primary sortkey. One could further extend the algorithm in these cases by specifically searching for specific character strings that arise with stereoisomers or optical isomers throughout the chemical name, but this is beyond the intent of this paper. We have also avoided dealing with chemical names that have subscripts or superscripts because there is no simple way of entering them into a computer at a terminal without some special designation. The primary strengths of using the SORTKEY algorithm for alphabetizing computer files containing chemical names is its simplicity and the fact that its implmentation is possible for most people. The version presented is sufficient for many situations, and several modifications to the algorithm can be made to extend its applicability.

ACKNOWLEDGMENT

The support for this work from the U.S. Environmental Protection Agency, Office of Pesticides and Toxic Substances, under Cooperative Agreement CR 806902020 is gratefully acknowledged.

REFERENCES AND NOTES

(1) Howard, Philip, H.; Sage, Gloria, W.; LaMacchia, A.; Colb, Andrew. "The Development of an Environmental Fate Data Base". *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 38–44.
(2) International Union of Pure and Applied Chemistry. Organic Chemistry Division. Commission of the Nonmenclature of Organic Chemistry. "Nomenclature of Organic Chemistry, Sections A, B, C, D, E, F, and H"; Pergamon Press: Oxford, 1979.
(3) "Chemical Abstracts Index Guide"; Chemical Abstracts Service: Columbus, OH, 1982.

Computer-Assisted Examination of Compounds for Common Three-Dimensional Substructures[1]

CHRISTOPHER W. CRANDELL*[†] and DENNIS H. SMITH[‡]

Department of Chemistry, Stanford University, Stanford, California 94305, and Lederle Laboratories, Pearl River, New York 10965

Received April 28, 1983

A program for finding common three-dimensional substructures within a set of chemical compounds is described. The program allows a user to define what constitutes commonality of substructures by providing control over the importance of degree of substitution, atom type, aromaticity, and hybridization. Simple examples are used to illustrate various phases of the search process, and an application of the program to a structure/activity problem is used as a more realistic example.

(1) INTRODUCTION

Comparison of structural features within a set of chemical compounds that display a common property is a frequent problem in chemical research. This problem can usually be characterized as one of relating the structure of the compounds to some "activity", i.e., structure/activity relationships in the broadest sense of the term. For example, the activity may be of a physical nature in that the compounds all display a characteristic pattern or subpattern in a spectroscopic technique or biological in that the compounds demonstrate similar physiological effects.

One generally makes the assumption that the common activity of a set of compounds is due to some similar structural feature or features. Establishing the relationship between common features and activities is of obvious importance. In biological applications, these relationships are useful, for example, in designing new drugs. In spectroscopic applications, these relationships provide data, for example, for correlation tables. The definition of structural similarity and what constitutes a structural feature are, of course, dependent on the specific application, but the problem can be stated and solved for the general case.

This paper describes an algorithm, currently implemented in an interactive computer program, for comparing three-dimensional (3-D) representations of structures to find 3-D common substructures that represent hypothetical requirements for activity, of a general nature. Such comparisons are generally done manually by using molecular models. There are obvious limitations to manual methods, particularly when

[†]Stanford University.
[‡]Lederle Laboratories.