



Bottom-up Self-organization for an Efficient Top-down Tree Search

メタデータ	言語: English 出版者: 北海道教育大学 公開日: 2010-07-12 キーワード: 作成者: 小原, 繁 メールアドレス: 所属:
URL	https://doi.org/10.32150/00004848

Bottom-up Self-organization for an Efficient Top-down Tree Search

Shigeru OBARA

Chemistry Laboratory, Kushiro Campus

Hokkaido University of Education, Kushiro 085 JAPAN

効率良い下方木探索のための上方自己組織化

小 原 繁

北海道教育大学釧路校化学教室

Abstract

A new sorting algorithm, named a bottom-up self-organization, is proposed. The algorithm adopts an organized structure in which data is sorted into a tree form so as to allow efficient data searches. One characteristic of the present method is that the addition of a new datum during sorting maintains the evenly balanced tree structure, ensuring search efficiency.

Introduction

Sorting of data is one of the basic steps in computer algorithms. Sorting consists of comparisons and additions of data; for a collection of data, for example all the words in a book, each datum should be compared with data in an ordered sequence, and if the same datum is not found, the datum should be added to the ordered sequence. Increasing the number of data in the sequence by the addition of new data makes comparison more difficult; 1 million data in the sequence, for example, require 1 million comparisons to confirm that a given datum is new. It is desirable to have a method which circumvents the more onerous tasks of comparison.

One improved method is a move-to-the-top method; the new datum or found datum after comparison is moved to the top of the sequence. This method is efficient if the same datum is to be compared repeatedly with the data in the sequence, but is not when all the data are distinct. It is useful when an algorithm with a high comparison efficiency and which can treat the addition of new data in an ordered sequence is obtained. The purpose of the present study is to propose such an efficient new sorting method.

The data structure of the new method is given in the next section, and the procedure for adding new data is described in the third section. The conclusions are given in the last section.

Data Structure in the Present Method

The present method adopts the tree data structure, so that a high efficiency of comparison is obtainable. The efficiency originates from the rapid reduction of the number of data to be compared after every comparison by virtue of the tree structure. The remaining problem is how the tree structure can be maintained even when new data are added. Before this point is discussed, the tree structure adopted in this study is described below.

The structure consists of "cells," which are depicted as ovals in Fig. 1. Each cell contains a maximum of 3 squares (Fig. 1), each of which directs a lower cell and specifies the range of data in the lower cell. The squares in the bottom cells contain single data, rather than ranges. The second lowest cell in Fig. 1, for example, contains 3 squares providing direction to the bottom 3 cells and denoting the data ranges $-\infty < x \leq D$, $D < x \leq F$, and $F < x \leq H$ in the bottom cells. Note that the range $D < x \leq F$ is expressed by the use of the largest data D in the lower left cell and the largest data F in the cell. The negative (positive) infinity means that there is no lower (upper) limitation.

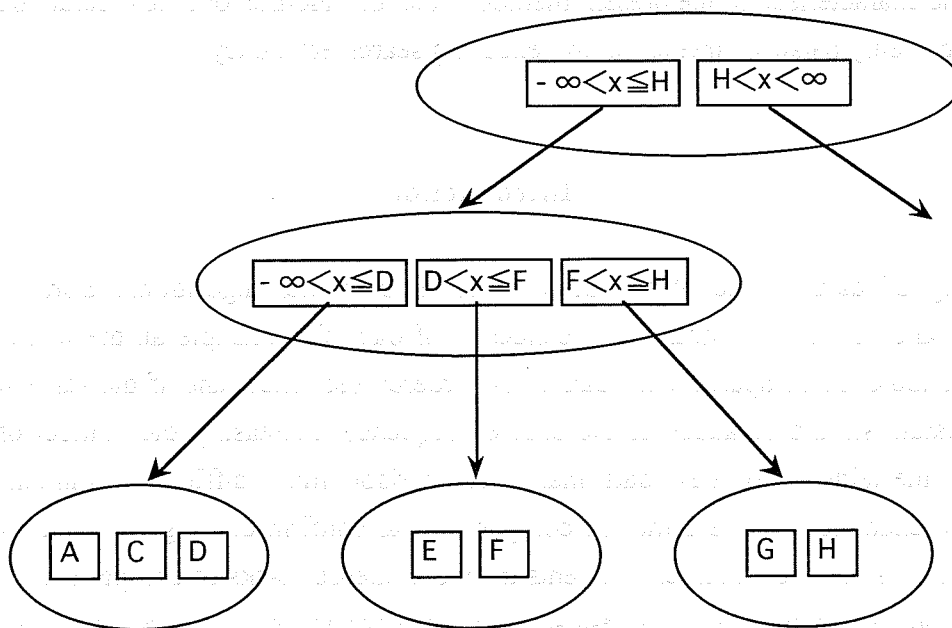


Fig. 1. Every oval denotes a "cell." The cell contains at most 3 squares. The squares in the bottom cells contain data, while those in upper cells contain the ranges of data in the lower cells. The negative and positive infinity mean that there are no lower or upper limits, respectively.

It may be helpful to provide an example of how a data search is carried out using the data structure indicated. Consider the search for data C. The search starts at the top cell. The data C is compared with the ranges in the cell, where it will be found in the range $-\infty < x \leq G$. The search

will then be carried out for the lower cell, directed by the square in the top cell. The data C is now in the range $-\infty < x \leq D$, so a further search will be performed for the bottom left cell. Finally, C will be found as the second data in the cell. Every comparison reduces the number of data to be compared, and the search applies the efficiency of the tree search.

Bottom-up Self-organization

When a new datum is found during sorting in the preceding section, the datum should be added to the set of data. The addition requires the reconstruction of the data structure, which will be discussed in this section.

The reconstruction is performed in a bottom-up manner ("bottom-up self-organization"). Consider a search of data B in the data structure shown in Fig. 1. The data B is in the range between A and C, so it should be added as a new data into the bottom left cell. After the addition, the cell contains 4 squares, as shown in Fig. 2a, and should then be divided. Divided cells are shown in Fig. 2b, and they are directed by 2 squares in the second lowest cell. In the present example, the second-lowest cell also contains 4 squares (Fig. 2b) and should also be divided into 2 cells, as shown in Fig. 2c. The top cell in Fig. 2c has 3 squares, so no further cell division is necessary. The scheme of the reconstruction is said to be

**division of a cell containing four squares,
addition of a square into the upper cell,
and modification of the ranges written in the squares.**

If the number of squares in the top cell is increased to four, this cell also should be divided into two cells and one cell is to be established above them as a new top cell.

The sorting starts with a single bottom cell with no square. Three distinct data are stored in the cell. When a fourth distinct datum is also stored in the cell, however, the cell will be divided into two cells with an addition of a upper cell. The fifth and following distinct data are treated in the manner described above.

Conclusions

A new method of sorting data has been proposed. The method constructs a data structure in a tree form in a bottom-up manner. Characteristics of the data structure and its construction are that (1) all the data have the same depth, namely, the same number of cells are passed through in comparison and, (2) the construction of the data structure is rather simple. The method has been used to make a dictionary of items in files in the Quantum Chemistry Literature Data Base (QCLDB).

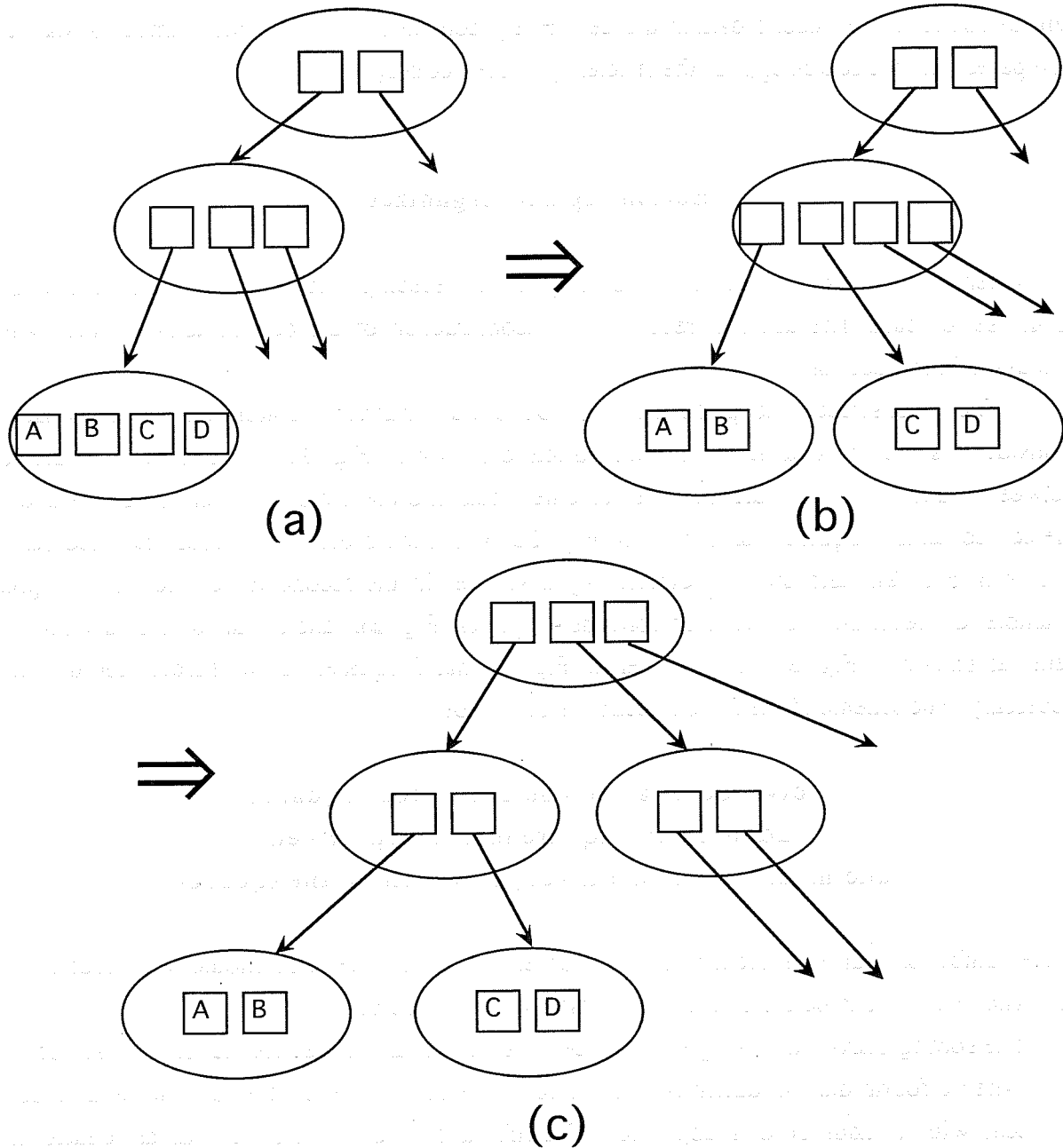


Fig. 2. (a) The bottom cell contains four data, so it is divided into two cells. (b) Divided bottom cells are directed by two squares in their upper cell. The second-lowest cell now contains four squares, so it is also divided into two cells. (c) Divided second-lowest cells are directed by two squares in the top cell. In all figures, the ranges in squares are omitted for simplicity.