G. M. Adelson-Velskii, E. M. Landis

1962

SovietRxiv

Abstract

Full Text

G. M. Adelson-Velskii, E. M. Landis

One Algorithm for the Organization of Information

(Presented by Academician I. G. Petrovskii, 17 IV 1962)

The note will deal with the organization of information located in the cells of an automatic computing machine. For definiteness, a three-address machine will be considered.

Statement of the problem. Information from a certain store enters the machine successively. An item of information is contained in a group of consecutive cells. An item of information contains a certain number—an evaluation of the information—different for different items. It is required to organize the placement of information in the machine memory so that, at any moment, searching for information with a given evaluation and entering a new item of information should require not too large a number of operations.

In this note an algorithm is proposed in which both searching and insertion are performed in $C \lg N$ operations, where N is the number of items of information that have arrived up to the given moment.

For storing the incoming information, a part of the machine memory is allocated. The items of information are placed there in the order in which they arrive. In addition, in another part of memory a "reference table" ([1]) is created, each cell of which corresponds to one of the items of information.

The reference table is a dyadic tree (Fig. 1a): each of its cells has at most one immediately subordinate left cell and at most one immediately subordinate right cell. Immediate subordination induces subordination (a partial ordering). Moreover, for each cell of the tree, all cells subordinate to the left (right) immediately subordinate cell will be located to the left (to the right) of the given cell. In addition, we assume that there exists a cell to which all the others are subordinate (the head). By transitivity, the concepts "to the left" and "to the right" extend to the set of all pairs of cells, and this set becomes ordered. Thus, a definite order of the cells in the reference table must coincide with the order of arrangement of the evaluations of the corresponding items of information (for definiteness, we shall consider the evaluations to increase from left to right).

In the first address of each cell of the reference table is indicated the place where the corresponding item of information is located. In the second and third addresses are located the addresses of the cells of the reference table immediately subordinate to the given cell, respectively on the left and on the right. If a cell has no immediately subordinate cells on some side, then in the corresponding address there is zero. In a certain fixed cell l the address of the head is stored.

Let us call a **chain** a sequence of cells of the tree in which each subsequent cell is immediately subordinate to the preceding one. For each cell of the tree we shall call the length of the left (right) branch the maximal length of a chain consisting of cells subordinate to the given cell and located to the left (to the right) of the given cell. Any chain whose length is equal to the length of a branch is called a stem of the branch.

We shall call a tree **admissible** if, for each of its cells, the length of the left branch differs from the length of the right one by no more than one (Fig. 1b).

In each cell of the address table, two bits are allocated in the code part for information about the lengths of the branches. If the left branch is longer than the right, these bits contain 1,0. If the right branch is longer than the left, they contain 0,1, and if they are equal, 0,0. In this connection it is assumed that if on some side there are no subordinate cells, then the length of the branch on that side is zero.

Fig. 1
Fig. 1

Fig. 2
Fig. 2

The insertion algorithm is such that at every moment the reference table is an admissible tree.

Lemma 1. *Let the number of cells of an admissible tree be equal to N. Then the maximum branch length is no greater than $\frac{3}{2}\log_2(N+1)$.*

Proof. Denote by N_n the minimum number of cells in an admissible tree for a given maximum branch length n. Then it is easily proved (see Fig. 2) that $N_n = N_{n-1} + N_{n-2} + 1$.

Solving this equation in finite differences, we obtain

$$N_n = \left(1 + \frac{2}{\sqrt{5}}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(1 - \frac{2}{\sqrt{5}}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n - 1.$$

Hence

$$n < \log_{\frac{1+\sqrt{5}}{2}}(N+1) < \frac{3}{2}\log_2(N+1),$$

as was required to be proved.

The algorithm for searching for an information element with a given key is as follows. We compare the given key with the key of the information element corresponding to the head. Depending on the result of the comparison, we proceed

to compare the given key with the key of the information element corresponding to the left or right immediately subordinate head of the cell. Suppose that k comparison steps have been performed and that at the k-th step the given key m was compared with the key m_u of the information element corresponding to some cell u of the reference table. If $m < m_u$ ($m > m_u$), then at the $(k+1)$-st step the key m is compared with the key of the information element corresponding to the cell immediately subordinate to u on the left (on the right). If $m = m_u$, the search is finished.

From the uniqueness of the head it follows that if, among the accumulated information, there is an information element whose key is equal to the given one, then at some step it will be found. The number of comparisons in this case will be equal to the number of cells in some chain of the tree (assuming that the comparison gives 3 answers). Since the number of operations is proportional to the number of such comparisons, it follows from Lemma 1 that the number of operations required in this algorithm for searching does not exceed $C \log_2(N+1)$.

We now turn to the description of the algorithm for constructing the reference table in the form of an admissible tree.

The tree is constructed as information arrives in the following way. When the first element of information arrives, the location of this element in memory is indicated in some cell in the first address. Zeros are entered in the remaining addresses and in the distinguished bits of the code. This cell is declared the head. Accordingly, its address is entered in the cell allotted for this purpose. Suppose an admissible tree has been constructed for N elements of information and the $(N+1)$-st element of information has arrived. We apply the search algorithm to the estimate m of this element, at the same time remembering the addresses of the cells of the chain along which we pass in this algorithm. In what follows we shall call this chain the **marked** one. If it turns out that the new element is already contained among the previous information, then the tree is not changed. Otherwise we shall reach a cell u having the following property. If $m < m_u$ ($m > m_u$), where m_u is the estimate of the information element corresponding to u, then the cell u has no immediate subordinate on the left (right). Then a new cell v, immediately subordinate on the left (right) to the cell u, is added to the directory column. In the first address of the cell v the place where the new information element is located is indicated; in the remaining addresses and in the distinguished bits there are zeros. The address of the cell v is placed in the corresponding address of the cell u.

Thus the directory column, augmented by the cell v, remains a tree, but it may be inadmissible. In addition, the distinguished bits in the cells of the marked chain need correction.

First of all, the distinguished bits in the cell u are corrected. There are two possibilities: 1) the distinguished bits of the cell u contained 1 0 (0 1) and 2) these bits contained 0 0. In the first case, a new branch is added to the cell u on the free side; in this case the lengths of both branches become equal (equal

Fig. 3

Figure 1: Fig. 3

to 1), and the length of any other branch in the tree is not changed. By putting 0 0 in the distinguished bits of the cell u, we complete the insertion, since the tree obtained is admissible. In the second case we change the distinguished bits of u to 1 0 if $m < m_u$, and to 0 1 if $m > m_u$.

Lemma 2. *A chain* $\mathfrak{C}$ *of an admissible tree, in all cells of which the distinguished bits contain* 0 0*, and whose last cell has no subordinates (zeros stand in the 2nd and 3rd addresses), is the stem of a branch for the cell immediately preceding the first cell* w *of the chain* $\mathfrak{C}$*.*

Proof. Suppose that the lemma is false. From the set of all chains beginning at w and having length greater than the length of $\mathfrak{C}$, choose a chain $\mathfrak{D}$ having the maximum number of common cells with $\mathfrak{C}$. Let t be the last common cell of the chains $\mathfrak{C}$ and $\mathfrak{D}$. Then its branches have different lengths; consequently, the distinguished bits of the cell t cannot contain 0 0. Thus Lemma 2 is proved.

Now consider the maximal connected part $\mathfrak{C}$ of the marked chain, going upward, beginning with the cell u, and consisting of cells in whose distinguished bits there is 0 0. Correction of the distinguished bits in the cells of $\mathfrak{C}$ consists in changing 0 0 to 1 0 or 0 1, depending on whether the next cell of the chain $\mathfrak{C}$ is immediately subordinate on the left or on the right.

It follows from Lemma 2 that: 1) the cells subordinate to any cell of $\mathfrak{C}$ form an admissible tree; 2) for the cells of $\mathfrak{C}$ the correction of the distinguished bits has been performed correctly; and 3) the length of any branch containing cells from the marked chain has increased by 1.

Moreover, note that the cells subordinate to any cell not belonging to the marked chain form an admissible tree which has not changed from the addition of the cell v.

There are 3 possible cases:

1) $\mathfrak{C}$ is the entire marked chain. Then we already have an admissible tree.

2) C lies in the short branch immediately preceding it, of cell s_0. Then, after adding cell v, an admissible tree has been obtained, and in the selected categories of cell s_0 it is only necessary to put 00.

3) C lies in the long branch of cell s_0. After adding cell v, this branch of cell s_0 became longer than the short branch by 2. Consider the three possible (up to symmetry) arrangements of the cells subordinate to s_0, shown in Fig. 3. A, B, C, D, E, F are branches not shown in the drawing. The number in parentheses denotes the length of the corresponding branch.

Fig. 3

Fig. 4

Figure 2: Fig. 4

Fig. 4

In this case the restructuring of the tree shown in Fig. 4 is performed. s is the cell immediately preceding s_0. The circles enclose the cells in which the indications of the immediately subordinate cells are changed. In square brackets are indicated the values of the selected categories in the corresponding cells, if they have to be changed. After this restructuring, as is clear from the drawing, the tree becomes admissible. If s_0 is the head (there is then no cell s), it is also necessary to change the fixed cell in which the address of the head is indicated.

Since the search requires $C \log_2(N+1)$ operations, moving upward along the marked chain up to cell s_0 requires no more than $C_1 \log_2(N+1)$ operations, and, finally, restructuring the tree requires a constant number of operations, in all $C \log_2(N+1)$ operations are required.

Received
13 IV 1962

CITED LITERATURE

1. Windley, *Comp. J.*, **3**, No. 2, 84 (1960).

Note: Figure translations are in progress. See original paper for figures.