ORGANIZATION OF OPERATIONS WITH BINARY TREES INDUCED BY STRUCTURES OF INDEX WORDS

G. S. Antoshchenkov

At present there are three basic approaches [1] to the solution of the problem of organizing operations with arrays ordered by collections of index words. The first of these is the establishment of horizontal connections between pairs of neighboring index words (linear lists); the second is the construction of a vertical tree of connections (binary tree); the third is the calculation of the address of the element, or of a group of elements, of the array from an index word (hashing function). The first approach, the easiest to implement, is characterized by the greatest time to retrieve an element. The third approach, optimally speedy, ordinarily only localizes a group of elements in the array and, on the final retrieval step, requires that there be hung on the field in the hash table either lists or binary trees, i.e., the problem as a whole is not thereby solved.

The method of the binary tree is optimal with respect to speed but its practical value is sharply lowered for reasons connected with the manner of constructing such trees. If, for the construction of a tree, use is made of the method of appending a new element of the data array to a free vertex, then the uniformity of the growing tree is determined by the degree of nonmonotonicity of the sequence of index words which occurs. A uniformly placed sequence of indexes generates a tree with the degree of arduousness of retrieval of $1.39 \log_2 n$ comparisons for an array with $n$ elements. However, in sequences of real occurrences of indexes, the rule of monotonicity may prevail over the rule of scattering, generating long branches. With this, the time to retrieve (find) an element becomes on the order of $n$ comparisons, just as with organization by lists.

To eliminate such nonuniform growth of binary trees, one eschews a simple method of free appending of elements to the array but turns, instead, to more complicated arrangements. The arrangement methods known at present have very arduous algorithms. For example, the construction of leveled trees can require a cardinal change of the entire binary tree. A less difficult method was suggested by Adel'son-Vel'ski and Landis [2] for constructing leveled trees. For its realization, at each vertex of the tree there is established a field of characteristic quantities and, for the introduction of a new element in the array, there is required both the usual appending of this element to the appended element. However, this algorithm also requires more time to introduce a new element than the time needed to retrieve the element.

We remark now that, in the construction of a binary tree, just as in the construction of a linear list, the index words are used only for comparison with the index of the introduced element, while the method of hash functions uses the internal properties of the index word, which also conditions its effectiveness. On the other hand, the difficulty of constructing uniform trees is connected with the fact that, when use is made only of the order of indexes, the configuration of a growing tree depends essentially on the order in which the elements appear. It turns out that the use of the internal binary structure of an index word allows one to construct a sufficiently uniform binary tree which does not depend on the order in which the elements appeared and which permits a simple and effective realization.

1. INDUCED BINARY TREES

We shall consider the intracomputer representation of the index words as Boolean vectors $W^k$ of length $k$ bits ($W^0 = \emptyset$) with the induced lexicographic relationship of the bits $W_1^k \geq W_2^k$, and, in case $k_1 \neq k_2$, the shorter word is filled with zeros on the right. We introduce the relationship of supplementation $W_1^k \equiv W_2^k$ ($W_2^k$ supplements $W_1^k$, or $W_1^k$ is supplemented by $W_2^k$) as the equality of the word composed of the first $k_1$ bits of word $W_2^k$ to word $W_1^k$. The relationship $\equiv$ defines a partial order on an arbitrary set of index words, where the equation of supplementation coincides with lexicographic equality, so that we shall denote it by $\equiv$, while strict supplementation is denoted by $\prec$. The identity relationship between words, with their lengths taken into account, we shall denote by $\equiv$.

We note that the relationship \(\equiv\) is a weakening of the relationship \(\leq\), i.e., \(\forall W_1, W_2 (W_1 \equiv W_2 \Rightarrow W_1 \leq W_2)\).

Let \(D = \{W_{i_1}^j\}_{i \in I}\) be a family of pairwise distinct, nonvacuous, and nonzero index words. We shall consider the binary tree \(T = \{t_{i_1}, t_{i_2}, \ldots\}\) where each vertex \(t_i\) is connected to word \(W_i \in D\) and not more than two other vertices \(t_{i_1}\) and \(t_{i_2}\) starting out from \(t_i\). The collection of all vertices connected with \(t_i\) by chains leading from one branch to another we call branch \(B_i\) of vertex \(t_i\). The branches \(B_i^l\) and \(B_i^r\) of vertices \(t_i^l\) and \(t_i^r\) we call respectively, the left and right branches of vertex \(t_i\). For sets \(A \subset D\) containing no fewer than two words, we define prefix \(P_A\) as the longest of those index words \(W_k\) such that \(\forall W_i^j \in A (W_k^j \equiv W_i^j)\). For \(A = \{W_m\}\) we set \(P_A = W_m\) and, if \(A = \emptyset\), then \(P_A = \emptyset\).

We now introduce the basic concepts. We shall say that tree \(T\) is induced by the structure of the index words if the following two conditions hold for it.

1. \(\forall i \in I (t_{i_1} \in B_i^l, t_{i_2} \in B_i^r (W_{i_1} < W_i < W_{i_2})).\)
2. \(\forall i \in I ((P_{B_i^l} \neq P_{B_i^r}) \& (P_{B_i^r} \neq P_{B_i^l})).\)

The first condition induces in the binary tree the order of the index words while the second induces their binary structure. We now consider certain properties of induced trees.

Property 1. Let \(k\) be the length of \(P_{B_i^l}\) for the vertices with word \(W_i\). We denote by \(w_{j}^m\) the \(m\)-th bit of word \(W_j\). Then, \(\forall t_i \in B_i^l, w_{i+1}^{k+1} = 0, \forall t_i \in B_i^r, w_{i+1}^{k+1} = 1\). Indeed, dropping the case of vacuity of \(B_i^l\) and \(B_i^r\), it is clear from the definition of the prefix that the \(k + 1\) bits of words of \(B_i\) must take both 0 and 1 values. It follows from condition 1 that we can find \(B_i^l\) in \(t_{i_1}\) with \(w_{i+1}^{k+1} = 0\), and \(t_{i_2}\) in \(B_i^r\) with \(w_{i+1}^{k+1} = 1\).

We now assume that, for \(t_{i_1} \in B_i^l, w_{i+1}^{k+1} = 1\), and then \(P_{B_i^l} = P_{B_i^r}\) despite condition 2, so that then the equation on the left will have been proved. The equation on the right is proven analogously.

Property 2. If the length of all the index words is bounded by the number \(M\), then the length of the induced tree also does not exceed \(M\). This follows from the capability of a vertex to increase the length of a prefix for its own left and right branches. Property 2 limits the number of comparisons in the search for an element of the array to the maximum number of bits in the index word.

Property 3. For a set of indices, taking the form of a sequence of binary numbers with a constant step, the induced binary tree is almost leveled, i.e., is optimal with respect to retrieval time. The regular monotonic application of a uniform distribution of indices occurs very frequently: in the numbering of the rows of algorithmic languages, in the images of maps in archives, and in data blocks of a retrieval system.

Property 4. The characteristic property of induced trees determines the value of the \(k + 1\) bits of the words found at the vertices of the right and left branches of vertex \(t_i\). The value of the \(k + 1\) bit of word \(W_i\) is not defined and may be arbitrary. We note here that if we postulate that this value is 1, then, for all the vertices with one branch incident out, this branch will leave on the left. We call such a tree induced on the left. For trees \(w_{i+1}^k = 0\), the nonterminal vertices will always have right branches, and we call such trees induced on the right. Each set \(D\) of index words has one tree induced on the left and one induced on the right. The left- and right-induced trees coincide only for vacuous, one-element, or complete sets of index words. We shall henceforth, for brevity's sake, call a binary tree induced on the left a BI-tree.

In Fig. 1 we give an example of a BI-tree for the set of all the basic symbols of ALGOL-60 with their English names. Each letter is codes by its five-bit binary ordinal number in the Latin alphabet, starting with 00001, while code 00000 denotes the absence of letters. In the matrix of the binary representations of the indices there are singled out the unit \((k + 1)\)-st bits and the prefixes corresponding to them.

2. PROCEDURE FOR CONSTRUCTING A BI-TREE

We now write, in the language of ALGOL-68 [3], the procedure "put" for introducing into a BI-tree \(T\).