Vol. 38 No. 4 Transactions of Information Processing Society of Japan Apr. 1997

Regular Paper

A New Analysis of Tree Hashing Algorithm

Ryozo Nakamura,[†] Tsuyoshi Itokawa[††] and Takuo Nakashima[†]

A new mathematical analysis is proposed to evaluate exactly the average search cost of tree hashing algorithm in consideration of the frequency of access on each key. The proposed analysis clarifies the relationship between the inserting order and its locating position for each key. It is shown that the evaluation formulae derived by the proposed analysis make it possible to evaluate exactly the average search cost in accordance with any probability distribution of the frequency of access. Besides this, it can see that under the assumption that the frequency of access is uniform the proposed evaluation formula of the average search cost is more correct and concise than that of the traditional analysis.

1. Introduction

Hashing method is an important technique widely used to store and retrieve records maintained in the form of a table. The key, which uniquely identifies its record, is mapped by a hashing function to a sequence of the table location, and records are inserted and searched by using these sequences. In this case, there will probably be two or more keys hashed to one identical location. Such an occurrence is called a collision. The techniques for collision handling are chiefly classified into two categories: chaining technique and open addressing technique. Tree hashing is a technique modified to use the binary search tree instead of the linear list in the chaining method.

The time required to solve a problem is one of the most important measures in evaluating an algorithm. The search cost is defined as the product of the number of probes and the frequency of access on a key. When the frequency of access is uniform, the average search cost of both the separate chaining and the binary search tree is independent of the order of inserting key. However when the frequency of access on each key is not uniform, the inserting order plays a crucial role.

For a tree hashing algorithm, the traditional analysis has been derived by Knuth[1)] in conformity to the assumption that all keys are uniformly accessed.

However the traditional one is unable to evaluate the search cost even if the probability of the frequency of access on a key is given. Taking account of the frequency of access, it is necessary to clarify the relationship between the inserting order of a key and its locating position.

In this paper, the average search cost in a successful search for a tree hashing algorithm is analyzed mathematically according to a model that considers the frequency of access on keys. The evaluation formulae of the search cost are then derived with the concrete probability distribution of the number of probes. It is shown that the proposed evaluation formulae make it possible to evaluate correctly the average search cost in accordance with any probability distribution of the frequency of access of each key.

2. Basic Concepts of Tree Hashing

Hashing is an important and useful technique for implementing dictionaries, which require constant time for storing and retrieving records on the average. In the general separate chaining method keys with the same hash address are kept in a linked list, but the elementary list searching is not a clever method for a large amount of data. Using the binary search tree instead of the linear list in the separate chaining method is therefore proposed, because the binary search tree is well known as a simple and efficient dynamic searching method.

We show the basic data organization of tree hashing in **Fig. 1**. The hash table is indexed by the bucket number $0, 1, 2, \ldots, M-1$ according to their positions on the table of size M and contains the M headers to point the root of a binary search tree. The elements of the j-th header are a set of records whose value of the hash function $h(\mathrm{a_i})$ is equal to j, namely, where the elements have the same hash address j.

We presume that N records are distributed

† Department of Computer Science, Faculty of Engineering, Kumamoto University

†† The Graduate School of Science and Technology, Kumamoto University

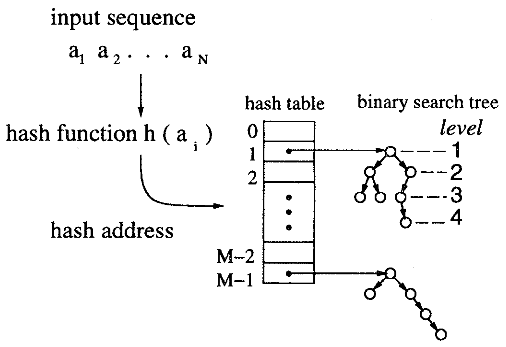


Fig. 1 Tree hashing data organization.

uniformly over the range of the hash table of size M by a hash function, and each hash sequence $a_1, a_2, \ldots, a_N$ $(0 \le a_i < M)$ is equally likely. We then have the probability P_{Nk} that exactly k of the a_i $(i = 1, 2, \ldots, N)$ are equal to j, for a fixed hash address j. It is obvious P_{Nk} is the binomial probability as follows,

$$P_{Nk} = \binom{N}{k}\left(\frac{1}{M}\right)^k\left(1-\frac{1}{M}\right)^{N-k} . \qquad (1)$$

In tree hashing algorithm the keys hashed at the same address are successively inserted into a binary search tree.

3. Traditional Analysis

The traditional analysis of tree hashing algorithm has been derived by Knuth under the assumption that all keys are uniformly accessed[1]. In Knuth's analysis the probability P_{Nk} that the number of keys in a fixed hash address is k after all N keys are stored has been denoted by Eq. (1) given previously.

The average search cost of the binary search tree containing k elements has also been denoted by C_k in the successful search and by C'_k in the unsuccessful search respectively as follows,

$$C_k = 2\left(1+\frac{1}{k}\right)H_k - 3 \qquad (2)$$

$$C'_k = 2H_{k+1} - 2 \qquad (3)$$

where H_k is a harmonic number[2].

In the successful searching, the total number of probes to find all keys in the binary search trees is $M\sum_{k=1}^{N} P_{Nk}kC_k$. As a result, the average search cost of tree hashing algorithm is denoted by following S_N,

$$S_N = \frac{M}{N}\sum_{k=1}^{N} k\left(2\left(1+\frac{1}{k}\right)H_k - 3\right)P_{Nk} \qquad (4)$$

In the unsuccessful searching, either an empty binary search tree is searched or all of the keys in a binary search tree are searched. The former searching is counted as one probe, the average search cost in the unsuccessful searching of tree hashing algorithm, denoted by $\overline{S}_N$, can be expressed as follows,

$$\overline{S}_N = \sum_{k=0}^{N}(2H_{k+1} - 2 + \delta_{k0})P_{Nk} \qquad (5)$$

$$\text{where } \delta_{k0} = \begin{cases} 1 \ , & (k = 0) \\ 0 \ , & (k > 0). \end{cases}$$

From the above analysis, assuming that the frequency of access is uniform, it is unnecessary to clarify the relationship between the inserting order and its locating position for each key.

4. Proposed Analysis

An analysis is proposed for a more general situation considering the frequency of access on an individual key. Here the inserting order of key plays a crucial role, and it is necessary to clarify the relationship between the inserting order of a key and its locating position for both the separate chaining technique and the binary search tree. In the analysis, first let α_{ijk} denote the probability that the i-th key inserted of N keys will be located in the $(j+1)$st position from the head of a sequence having k keys for a fixed hash address, where the keys hashed at the same address are numbered in order of their occurrence. There are $\binom{i-1}{j}$ possible combinations to distribute the $i-1$ keys into the front j positions of the sequence with k keys and similar $\binom{N-i}{k-j-1}$ ways to distribute the $N-i$ keys into the rear $k-j-1$ positions of the sequence. Therefore the probability α_{ijk} can be expressed as follows[3],

$$\begin{aligned}\alpha_{ijk} &= \binom{i-1}{j}\binom{N-i}{k-j-1} \Big/ \sum_{j=0}^{k-1}\binom{i-1}{j}\binom{N-i}{k-j-1} \\ &= \binom{i-1}{j}\binom{N-i}{k-j-1}\Big/\binom{N-1}{k-1} \qquad (6)\\ &\text{where } 0 \le j < i .\end{aligned}$$

Here $\sum_{j=0}^{k-1}\alpha_{ijk} = 1.0$.

Next we analyze the insertion and search algorithms of a binary search tree in consideration of the frequency of access on keys. As the keys hashed at the same address are successively inserted into a binary search tree, it is necessary to derive the probability that the

j-th key of those k keys will be located in the l-th level of a binary search tree.

We have analyzed the probability β_{jl} that the $(j+1)$st key inserted into a binary search tree will be located in the $(l+1)$st level of the tree as follows[4),5)],

$$\beta_{jl} = \frac{2}{j+1}\beta_{j-1,l-1} + \frac{j-1}{j+1}\beta_{j-1,l} \ . \qquad (7)$$

The initial conditions of β_{jl} is

$$\beta_{j0} = \begin{cases} 1 \ , & \text{if } j=0 \\ 0 \ , & \text{if } j>0 \ . \end{cases} \qquad (8)$$

We can now get information about the probability β_{jl} by using a generating function $G_j(z)$,

$$G_j(z) = \beta_{j0} + \beta_{j1}z + \beta_{j2}z^2 + \cdots = \sum_l \beta_{jl}z^l \ . \qquad (9)$$

From Eq. (8) we have $G_0(z) = 1$, and from Eq. (7) we have

$$G_j(z) = \frac{2z}{j+1}G_{j-1}(z) + \frac{j-1}{j+1}G_{j-1}(z) = \frac{2z+j-1}{j+1}G_{j-1}(z) \ . \qquad (10)$$

We can now obtain

$$G_j(z) = \frac{2z+j-1}{j+1}\cdot\frac{2z+j-2}{j}\cdot\frac{2z+j-3}{j-1}\cdots\frac{2z+1}{3}\cdot\frac{2z}{2} = \frac{\sum_{l=0}^{j}\left[{j \atop l}\right]2^lZ^l}{(j+1)!} \qquad (11)$$

where $\left[{j \atop l}\right]$ is Stirling numbers of the first kind[2)].

Therefore the probability β_{jl} can be expressed in terms of Stirling numbers as follows,

$$\beta_{jl} = \frac{\left[{j \atop l}\right]2^l}{(j+1)!} \ , \ (l = 0,1,2,\cdots,j) \qquad (12)$$

where $\sum_{l=1}^{j}\beta_{jl} = 1.0$.

We have found the probability α_{ijk} which the i-th key inserted will be located $(j+1)$st position from the head of a sequence having k keys for a fixed hash address when N keys are scattered over the hash table, and the probability β_{jl} which the $(j+1)$st key inserted into the binary search tree will be placed the $(l+1)$st level in the tree.

We are now in the position to analyze the search algorithm of tree hashing. By ρ_i we denote the probability that the i-th key inserted into the hash table will be retrieved, where $\rho_1 + \rho_2 + \cdots + \rho_N = 1.0$. We note that the number of probes needed to find a key is exactly one more than the number of probes that were needed when that key was entered into the tree. Therefore let γ_{kh} be the probability that a h-th level key in the binary search tree containing k keys is probed, we can derive the probability γ_{kh} as follows;

$$\gamma_{kh} = \sum_{i=1}^{N}\rho_i \sum_{j=h-1}^{k-1}\alpha_{ijk}\beta_{j\ h-1}. \qquad (13)$$

Here we get that $\sum_{h=1}^{k}\gamma_{kh} = 1.0$.

Let q_{Nh} be the probability that the number of probes needed to find a key is equal to h in searching of tree hashing, it can be expressed by

$$q_{Nh} = \sum_{j=h}^{N}\gamma_{jh}P_{Nj} \ . \qquad (14)$$

Finally we consider the number of probes as a random variable and its probability distribution in order to construct the evaluation formulae of the search cost. Here we only argue in a successful searching case, since the frequency of access on a key plays a crucial role only in this case. Besides this, it is important that only binary search trees having more than one key in the tree are probed for successful searching. We can derive the average search cost S_N and the variance V_N in a successful search as follows,

$$S_N = \sum_{k=1}^{N}kq_{Nk} \Bigg/ \sum_{k=1}^{N}P_{Nk} \qquad (15)$$

$$V_N = \sum_{k=1}^{N}k^2q_{Nk} \Bigg/ \sum_{k=1}^{N}P_{Nk} - {S_N}^2. \qquad (16)$$

5. Comparisons

In this section, the proposed analysis is compared to the traditional one under the assumption that the probability of the frequency of access on each key is equally likely.

From the above assumption, ρ_i $(i = 1,\ldots,N)$ becomes $1/N$, and then the probability γ_{kh} of Eq. (13) becomes as follows.

$$\begin{aligned}\gamma_{kh} &= \sum_{i=1}^{N}\rho_i \sum_{j=h-1}^{k-1}\alpha_{ijk}\beta_{j\ h-1} \\ &= \sum_{i=1}^{N}\frac{1}{N}\sum_{j=h-1}^{k-1}\alpha_{ijk}\beta_{j\ h-1} \\ &= \sum_{j=h-1}^{k-1}\left\{\frac{1}{N}\sum_{i=1}^{N}\binom{i-1}{j}\binom{N-i}{k-j-1}\right.\end{aligned}$$

$$\Big/ \binom{N-1}{k-1} \Big\} \beta_{j\ h-1}$$

$$= \sum_{j=h-1}^{k-1} \left\{ \frac{1}{N} \binom{N}{k} \Big/ \binom{N-1}{k-1} \right\} \beta_{j\ h-1}$$

$$= \frac{1}{k} \sum_{j=h-1}^{k-1} \beta_{j\ h-1}$$

We then can derive the average search cost S_N as follows.

$$S_N = \sum_{k=1}^{N} k q_{Nk} \Big/ \sum_{k=1}^{N} P_{Nk}$$

$$= \sum_{k=1}^{N} k \sum_{j=k}^{N} \gamma_{jk} P_{Nj} \Big/ \sum_{k=1}^{N} P_{Nk}$$

$$= \sum_{k=1}^{N} \sum_{j=1}^{k} j \gamma_{kj} P_{Nk} \Big/ \sum_{k=1}^{N} P_{Nk}$$

$$= \sum_{k=1}^{N} \frac{1}{k} \left\{ \beta_{00} + \sum_{i=1}^{k-1} \sum_{j=1}^{i} (j+1)\beta_{ij} \right\} P_{Nk} \Big/ \sum_{k=1}^{N} P_{Nk}$$

$$= \sum_{k=1}^{N} \frac{1}{k} \left\{ 1.0 + \sum_{i=1}^{k-1} (G_i(1) + G_i'(1)) \right\} P_{Nk} \Big/ \sum_{k=1}^{N} P_{Nk}$$

$$= \sum_{k=1}^{N} \frac{1}{k} \left\{ 1.0 + \sum_{i=1}^{k-1} (2H_{i+1} - 1) \right\} P_{Nk} \Big/ \sum_{k=1}^{N} P_{Nk}$$

$$= \sum_{k=1}^{N} \frac{1}{k} \{2(k+1)H_k - 3k\} P_{Nk} \Big/ \sum_{k=1}^{N} P_{Nk}$$

$$= \sum_{k=1}^{N} \left\{ 2(1+\frac{1}{k})H_k - 3 \right\} P_{Nk} \Big/ \sum_{k=1}^{N} P_{Nk} \tag{17}$$

In the above formulation, we use the generating function given by Eq. (10), and using differentiation of it we get

$$G_i'(z) = \frac{2}{i+1} G_{i-1}(z) + \frac{2z+i-1}{i+1} G_{i-1}'(z).$$

Therefore we find that

$$G_i'(1) = \frac{2}{i+1} + G_{i-1}'(1) ,$$

and

$$G_i'(1) = 2(H_{i+1} - 1) ,$$

where H_i is a harmonic number and

$$\sum_{i=1}^{k} H_i = (k + 1)\ H_k - k .$$

The proposed formula (17) differs from the traditional equation (4) in that M/N and the first parameter k in Eq. (4) disappear in the Eq. (17). Besides this, only binary search trees having more than one key in the tree are probed for successful searching in the proposed analysis, but Knuth did not regard to do so. As a result, the proposed equation is terser and fitter than that of the traditional analysis.

We shall discuss the reasons why these evaluation formulae are different.

First, in the proposed analysis the number of probes itself is regarded as a random variable and its probability distribution is derived concretely. On the other hand, in the traditional analysis described in Section 3, N keys are equally scattered over M binary search trees, and then using the average number of probes of the tree, the average search cost of tree hashing algorithm has been derived.

As a result, the difference is caused by the way in which a random variable and its probability distribution are defined. Besides this, in the proposed analysis, only binary search trees having more than one key are probed for successful searching. Thus it is shown that the proposed analysis is both appropriate and accurate in evaluating the search cost of tree hashing.

6. Numerical Tests

The proposed evaluation formulae (15) and (16) can evaluate the search cost in accordance with any probability distribution of the frequency of access on a key. In the numerical tests let us assume the following two probability distributions of the frequency of access on a key.

i) The probability of the frequency of access on each key is equally likely, called "Uniform", the probability ρ_i holds the relation $\rho_i = 1/N$ $(i = 1, 2, \cdots, N)$.

ii) The probability of the frequency of access on a key is reduced harmonically according to the order of inserting a key, typically called "Zipf's law", the probability ρ_i holds the relation $\rho_i = c/i$ $(i = 1, 2, \cdots, N)$, where $c = 1/H_N$ and H_N is a harmonic number.

The numerical results with two probability

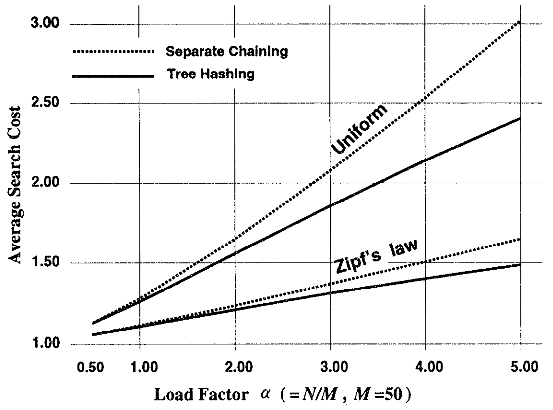


Fig. 2 Comparisons of the average search cost in a successful search under consideration of the probability distributions.

distributions are shown in **Fig. 2**, where the average search cost of tree hashing (the solid line) is compared to the separate chaining method (the dotted line)$^{3)}$. We can see that the performance of tree hashing is better than that of the separate chaining method with increasing of the number of records.

7. Conclusions

We have analyzed mathematically the average search cost of a tree hashing algorithm in consideration of the frequency of access on an individual key. Taking account of the frequency of access on a key it is necessary to clarify the relationship between the inserting order and its locating position for each key.

In the proposed analysis we have clarified two relationships. The first is the relation between the inserting order into the hash table and its locating position in a sequence with the same hash address, and the second is one between the inserting order into a binary search tree and its locating level in the tree. Compounding the above two relations we have proposed a new analysis of a tree hashing algorithm in consideration of the frequency of access on each key. The proposed evaluation formulae have been derived from the concrete probability distribution of the number of probes. It has been shown that the proposed analysis makes it possible to evaluate exactly the performance of tree hashing.

Finally, a tree hashing algorithm may seem so obvious we should not bother to analyze it, but analyzing of the algorithm in detail is an attractive study, and makes a good manner in which more complicated algorithms may be analyzed.

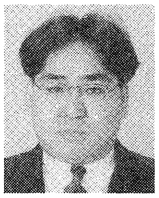
Acknowledgments The authors would like to thank the anonymous referees for their valuable comments to improve the clarity of this paper.

References

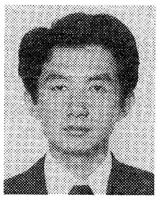
1) Knuth, D.E.: The Art of Computer Programming Vol.3, Sorting and Searching, pp.506–549, Addison-Wesley, Reading, MA (1973).
2) Knuth, D.E.: The Art of Computer Programming Vol.1, Fundamental Algorithms, pp.51–73, Addison-Wesley, Reading, MA (1973).
3) Nakamura, R.: An Alternative Analysis of the Algorithm for Separate Chaining Technique of the Hashing Method, *Trans. IPS Japan*, Vol.34, No.1, pp.10–15 (1993).
4) Nakamura, R.: An Analysis of Insertion and Search Algorithms of a Binary Search Tree, *Trans. IPS Japan*, Vol.26, No.6, pp.1106–1112 (1985).
5) Nakamura, R. and Nakashima, T: An Analysis of Search Algorithm of Binary Search Tree, *Trans. IEICE*, Vol.J75-D-I, No.9, pp.830–835 (1992).

(Received April 1, 1996)
(Accepted January 10, 1997)

Ryozo Nakamura received the M.E. degree from Kumamoto University in 1968 and the D.E. degree in computer science from Kyushu University in 1985. From 1968 to 1974, he joined Chubu Electric Power Company. Since 1975 he has joined in Faculty of Engineering of Kumamoto University, and is presently a professor in Department of Computer Science. His current research interests include the design and analysis of algorithms and data structures.



Tsuyoshi Itokawa received the B.E. degree from Kumamoto University in 1993. From 1993 to 1994 he joined Fujitsu Kyushu System Engineering LTD., and received the M.E. degree from Kumamoto University in 1997. He is presently working for the D.E. degree in the Graduate School of Science and Technology, Kumamoto University. His current research interests include the design and analysis of algorithms and data structures.



Takuo Nakashima received the M.E. degree from Kumamoto University in 1986. From 1986 to 1988 he joined Fujitsu Company. Since 1991 he˙ has joined in the Faculty of Engineering, Kumamoto University, and is presently a research associate in Department of Computer Science. His research interests include the design and analysis of algorithms and data structures, and computer network.