Hash Addressing with Conflict Flag

Koichi Furukawa*

Abstract

In this paper, a new scatter storage method using conflict flag is introduced and the mean reject time of this method is given (the reject time is the time required to identify an item as a nonmember of the given table).

The new method is compared with some other scatter storage methods. It is shown that the one-bit information reduces the reject time by more than 50% when the load factor exceeds 60%. Some applications of this method are described.

1. Introduction

The scatter storage methods are used in symbol tables of assemblers and compilers, in on-line files which are retrieved very fast, etc. The mean access time of the scatter storage table is constant regardless of the table length.

The method proposed here uses an additional one-bit flag besides the use-bit. This flag, which we call "conflict flag", is used to mark the entries which cause conflicts. The reject time is shortened by referring to the conflict flag in the retrieval phase. In addition, it is shown that this one-bit flag facilitates the deletion of records from the table.

In section 2, the new algorithms for handling the hash table are given, which use the conflict flag as well as the use-bit. In section 3, the method given here is compared with the conventional random probing method and the direct chaining method with respect to the mean reject time. In section4, some applications of this method are given.

This paper first appeared in Japanese in Joho — Shori (Journal of the Information Processing Society of Japan), Vol. 13, No.8 (1972), pp. 533～539.

* Information Systems Section, Software division, Electrotechnical Laboratory

2. Hash Table Handling Algorithms

Let N be the length of a table T, and assume that T is addressed from 1 to N. Each cell is composed of the following fields.

UFLAG	(one-bit field)
CFLAG	(one-bit field)
KEY	(key field)
DATA	(data field)

UFLAG indicates whether the cell is used(=1) or not(=0). CFLAG indicates the conflict mark, namely, whether the cell has caused conflicts(=1) or not(=0). Hash functions are represented by $h_0, h_1, h_2, \ldots$ NUMBER is a counter which counts the number of items in T.

2.1. Entering Algorithm

In the following algorithms, P0, P, Q and R are pointer variables which point to cells of the hash table, while K represent the key to be entered.

Algorithm E (Enter).

E1. [Examine number.] If NUMBER$\geq\alpha$N, the algorithm terminates. (Overflow.)

E2. [First hash.] Set P$\leftarrow h_0$[K], i$\leftarrow$1, P0$\leftarrow\Lambda$. (P0 is used to point the first empty cell.)

E3. [Is the cell used?] If UFLAG(P)=0 and P0=Λ, set P0$\leftarrow$P, and then go to E5.

E4. [Multi-entry?] If KEY(P)=K, the algorithm terminates unsuccessfully. (Multi-entry.)

E5. [Examine conflict.] If CFLAG(P)=1, set P$\leftarrow h_i$[K], i$\leftarrow$i+1,and then go back to E3.

E6. [Examine P0.] (Now, P points to the end cell of the conflict chain.) If P0=Λ, set P0$\leftarrow$P, and then go to E7; otherwise set KEY(P0)$\leftarrow$K, UFLAG(P0)$\leftarrow$1, NUMBER$\leftarrow$NUMBER+1. The algorithm terminates successfully.

E7. [Find empty cell.] Set CFLAG(P0)$\leftarrow$1, P0$\leftarrow h_i$[K], i$\leftarrow$i+1. If UFLAG(P0)=0, go back to E6 (Empty cell was found.); otherwise repeat this step. (Here, the pointer P0 is used instead of P.)

2.2 Retrieval Algorithm

The CFLAG plays the main role to reduce the reject time. When it is found that the calculated cell contains a key different from the retrieval key, the CFLAG is examined to decide whether the search should be continued or not.

Algorithm R (Retrieve).

R1. [First hash.] Set $P \leftarrow h_0[K]$, $i \leftarrow 1$.

R2. [Found?] If KEY(P)=K and UFLAG(P)=1, the algorithm terminates successfully. (P points the desired cell.)

R3. [Examine conflict.] If CFLAG(P)=0, the algorithm terminates unsuccessfully. (Not found.)

R4. [I-th hash.] Set $P \leftarrow h_i[K]$, $i \leftarrow i+1$. Then go back to R2.

2.3. Deletion Algorithm

An item can be deleted by setting the UFLAG to 0. If the CFLAG of the deleted cell is 1, the search time is not reduced by the deletion.

Algorithm D (Delete).

D1. [First hash.] Set $P \leftarrow h_0[K]$, $i \leftarrow 1$.

D2. [Found?] If KEY(P)=K and UFLAG(P)=1, set UFLAG(P)$\leftarrow$0. The algorithm terminates successfully.

D3. [Examine conflict.] If CFLAG(P)=0, the algorithm terminates unsuccessfully. (Not found.)

D4. [I-th hash.] Set $P \leftarrow h_i[K]$, $i \leftarrow i+1$. Then go back to D2.

3. Reject Time

Let $P_k(j)$ be the probability that j cells are marked when the table contains k items. We denote this state by (k,j). $P_{k+1}(j)$ can be expressed as the weighted sum of the terms $P_k(n)$, $n=0 \sim j$, as follows:

$$P_{k+1}(j) = \sum_{n=0}^{j} P_k(n) \cdot (N-k) \cdot \sum_{i=0}^{n} \binom{j-n+i}{i} \cdot \prod_{h=0}^{i-1} (n-h) \cdot \prod_{g=0}^{j-n-1} (k-n-g) \cdot \prod_{m=0}^{i+j-n} (N-m)^{-1}.$$

(Here we define $\prod_{g=0}^{-1} (k-n-g)=1$, and $P_k(k)=0$.) (1)

The coefficient of $P_k(n)$ is the transition probability from the state (k,n) to the state (k+1,j). The i-th term of this coefficient corresponds to the case that the k+1-th item accesses i marked cells, and j-n unmarked occupied cells before it accesses an empty cell.

Identity (1) can be rewritten as follows:

$$P_{k+1}(j)=\sum_{n=0}^{j} P_k(n)\cdot\frac{N-k}{N}\cdot\prod_{g=0}^{j-n-1}\frac{k-n-g}{N-g-1}\cdot\sum_{i=0}^{n}\binom{j-n+i}{i}\cdot\prod_{h=0}^{i-1}\frac{n-h}{N-j+n-h-1} \qquad (2)$$

Now we attempt to simplify (2).

Lemma 1. If

$$F_j(n)=\sum_{i=0}^{n}\binom{j-n+i}{i}\cdot\prod_{h=0}^{i-1}\frac{n-h}{N-j+n-h-1}\ , \qquad (3)$$

then

$$F_j(n)=\prod_{h=0}^{n-1}\frac{N-h}{N-j+n-h-1}\ , \quad \text{for } N>j\geq n\geq 1. \qquad (4)$$

Proof: This lemma is proved by induction on n and j, using the relation $\binom{n+1}{i}=\binom{n}{i-1}+\binom{n}{i}$. The detail of the proof is not given here.

Lemma 2.

$$P_{k+1}(j)=\frac{N-k}{\prod_{i=0}^{j}(N-i)}\cdot\sum_{n=0}^{j}P_k(n)\prod_{g=0}^{j-n-1}(k-n-g)\cdot\prod_{h=0}^{n-1}(N-h),\ \text{for } j=1,2,\dots,k \qquad (5)$$

and

$$P_{k+1}(0)=\frac{N-k}{N}\cdot P_k(0). \qquad (6)$$

Proof: (5) is easily obtained by substituting (4) to (2). (6) is the immediate result from (1). Q.E.D.

From (5), the identity

$$(N-n+1)\cdot P_{k+1}(n+1)-(k-n)\cdot P_{k+1}(n)=(N-k)\cdot(n+1) \qquad (7)$$

is deduced. Finally a very simple expression for $P_{k+1}(j)$ is obtained from (7). The following theorems hold.

Theorem 1.

$$P_{k+1}(j)=\frac{N-k}{N-j}\cdot\left(\sum_{n=0}^{j}P_k(n)-\sum_{n=0}^{j-1}P_{k+1}(n)\right)\ . \qquad (8)$$

Proof: It is easily deduced by summing (7) for n=0,1,...,j and substituting (6) to the sum. Q.E.D.

Theorem 2. When the table of the length N contains k items, the mean reject time $E_c(k)$ is given by

$$E_c(k)=\sum_{j=0}^{k-1}P_k(j)\cdot E_n(j)\ , \qquad (9)$$

where

$$E_n(j)=\frac{N+1}{N-j+1}\ . \qquad (10)$$

Proof: When the table contains j marked cells, the mean reject time is equal to the average probe number to enter a j+1-th item, and is given by (10).[1] Q.E.D.

Next, we compare this result with that of the direct chaining method.[1] Let $E_d(k)$ be the mean reject time of the direct chaining method. The following theorem holds:

Theorem 3.

$$E_d(k) \approx e^{-\alpha}+\alpha, \tag{11}$$

where $\alpha = \frac{k}{N}$.

Proof: If the table contains k items and there are n_i chains with length i, the mean reject time $t_d(k)$ is given by

$$t_d(k)=\frac{n_0}{N} + \sum_{i=1}^{k} \frac{n_i}{N}\cdot i = \frac{n_0}{N} + \frac{k}{N} ,$$

where n_0 is the number of empty cells.

If N and k are large enough, it is known[2] that the probability $P_{n_0}(k,N)$ of finding exactly n_0 cells empty is given by the Poisson distribution

$$P(n_0;\lambda)=e^{-\lambda}\frac{\lambda^{n_0}}{n_0!} ,$$

where $\lambda = Ne^{-\alpha}$. The expectation of n_0 is λ and

$$E_d(k)=\frac{1}{N}\cdot Ne^{-\alpha} +\alpha = e^{-\alpha} +\alpha . \quad \text{Q.E.D.}$$

Figure 1 shows $E_c(k)$, $E_d(k)$ and $E_n(k)$.

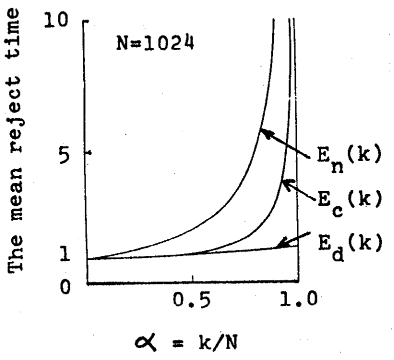

Figure 1. The mean reject time of each method.

4. Some Applications of This Method

The direct application of the scatter storage method using conflict flag is the membership testing problem. It is very likely that many items are not included in a given set. In such cases, it is very effective to reduce the reject time for the improvement of the system performance.

Let us consider the instruction table of an assembler. The table is assumed to contain only the mnemonic codes of machine instructions and some pseudo instructions. Whenever the assembler encounters macro instructions, it fails in searching the instruction table and this gives the situation of the membership testing problem.

This method can also be used in handling the overflow problem, which is the most cumbersome problem of the scatter storage techiques. There are two different methods to resolve this problem. One is rehashing all items in the new larger table. This is a very time-consuming operation. The other method is to put the overflowed items in other space. Some new aspects are introduced in this method. The extra-space again constitutes a hash table with the same hash function as used in the first table. This table is linked to the first table. If the second table overflows, then still a new table is constracted. The average probe number E_n when n tables are full is given by

$$E_n = \frac{k-1}{2} \cdot t_\alpha + E_1 \qquad (12)$$

where α is the load factor of the overflow point, and E_1 and t_α are the average probe number and the mean reject time of the table with the load factor α , respectively. E_k also depends on the reject time, and the scatter storage method with conflict flag works very effectively in this scheme.

Acknowledgement

The author wishes to express his sincere thanks to Dr. H. Nishino, the head of the Pattern Recognition division, Dr. O. Ishii, the head of the Computer Science division, and Y. Kato, the chief of the Information Systems Section. He is also indebted to M. Arisawa and the members of the Information Systems Section for suggesting many stimulating discussions and H. Hasegawa for refining this paper.

References

[1] Morris, R.: Scatter storage techniques. CACM 11, 1 (Jan. 1968), pp. 38 44.

[2] Feller, W.: An Introduction to Probability Theory and Its Applications, Vol.1. John Wiley & Sons, New York, 1950.