Reducing the Retrieval Time of Hashing Method by Using Predictors

SEIICHI NISHIHARA and KATSUO IKEDA *University of Tsukuba*

Authors' Present Address: Seiichi Nishihara and Katsuo Ikeda, Institute of Information Sciences and Electronics, University of Tsukuba, Sakura-mura, Niihari-gun, Ibaraki 305 Japan.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission. © 1983 ACM
0001-0782/83/1200-1082 75¢

***ABSTRACT:** Many methods for resolving collisions in hashing techniques have been proposed. They are classified into two main categories: open addressing and chaining. In this paper, other methods are presented that are intermediate between the two categories. The basic idea of our methods is the use of one or more predictors reserved per cell instead of a link field as in the chaining method. The predictors are used to maintain loose synonym chains. After describing the methods, the efficiencies are estimated theoretically and verified experimentally. In comparison with the chaining method, we prove that our methods significantly reduce the average number of probes necessary to retrieve a key without expending extra space.*

1. INTRODUCTION

Hash addressing is well known as an efficient technique for reducing the number of probes required to store or retrieve keys in a table. Its most remarkable feature is that the average number of probes depends just on the fraction α of the table occupied; it is not affected by the total number of keys. Since any key-to-address transformation generally makes a many-to-one mapping, it will probably happen that more than one distinct keys are hashed to the same address. Those keys with the same home address are called *synonyms*. Such an occurrence, called a *collision*, causes many kinds of clustering phenomena [1].

Many techniques for resolving collisions have been proposed. They are classified mainly into two categories: open addressing and chaining [3, 4]. In open addressing, in addition to the hash function H that determines the home address $H(k)$ of a key k to be stored, a collision-resolution function h is necessary for tracing through the table until an empty cell is encountered. The probe-sequence generated by the function h for a key k is expressed as $h(i, k)$, $i = 0, 1, \ldots, M - 1$, where M is the table size. Here, $h(0, k) = H(k)$ and $0 \leq h(i, k) \leq M - 1$, for $i = 1, 2, \ldots$. The other approach to collision resolution is the direct-chaining method [4], in which all the keys transformed to the same address are kept in a chain using simple list-processing techniques.

Both hashing techniques and many variations are surveyed in [3] and [4]; the details are omitted here. Assuming equal usage of cells, the theoretical approximation of the average number of probes necessary to retrieve a key in a hash table is given for each method [3, 4]:

$$\begin{array}{ll} -(1/\alpha)\ln(1-\alpha) & \text{for open addressing, eliminating primary and secondary clusterings [1]} \\ 1 + \alpha/2 & \text{for direct chaining} \end{array} \quad (1)$$

where α is the load factor of the table. In general, the average number of probes needed in open addressing cannot be less than the number needed in chaining. In a recent paper [2], a method named *pseudochaining*, which combines characteristics of open addressing and chaining, was proposed. The performance of pseudochaining lies between that of the other two methods.

Another combined method is proposed here, whose performance in terms of the average number of probes is essentially equal to that of chaining. In the following sections, a new method using a predictor, a several bit field assigned to each cell, and an extension of this method are described. Then the retrieval efficiencies of the predictor method and its extension, the *multiple-predictor method*, are estimated theoretically and verified by experiments. In the conclusion, it is proved that a predictor of more than four or five-bit length is always preferable to chaining from the viewpoint of efficient use of memory, and furthermore that the multiple predictor breaks through the limitation $1 + \alpha/2$ of direct chaining without expending extra space.

2. DESCRIPTION OF METHODS

2.1. The Single-Predictor Method

Our technique is applied to the open-addressing method in which secondary clustering may occur. A hash table of size M is a set of M successive cells addressed from 0 to $M - 1$. Each cell contains not only an item space (key field) but also a p bit field as a predictor. It holds a nonnegative integer q, that is used for the purpose of tracing only synonyms, that is, keys in the same cluster, where $0 \le q \le (2^b - 1)$.

Assume that the search for key k is being performed at the address $h(i, k)$, that is, none of the cells $h(0, k), \ldots, h(i, k)$ contains the key k. In the usual open-addressing method, the next search address is $h(i + 1, k)$. However, if the key in the $h(i + 1, k)$th location is not a synonym of k, there is no need to check this location. In this case, the predictor value q is used to indicate the number of probes to be skipped until the next address containing a synonym is encountered. In other words, the next synonym is found in the $h(i + q, k)$th location. Note that the function $h(i, k)$ is assumed to be directly computable; it takes a key value k and a nonnegative integer i as arguments and returns an address in the table. The predictor of the last cell of a cluster is set to zero. This means that no more synonyms exist in the table, which is the natural extension of the interpretation of predictors and is very effective for reducing reject time. In some cases, the number of probes that should be skipped in the search for another synonym is greater than the maximum predictor value $\max(=2^b - 1)$. When this occurs, after checking the $h(i + \max, k)$th location, we must repeat probing operations one by one until a synonym is encountered following the probe sequence. This phenomenon is the only factor that makes the average number of probes greater than the direct-chaining method. The additional cost is estimated in the following section.

The algorithms for storing and retrieving keys may easily be derived and so are not formulated here. The detailed algorithm given later for the multiple-predictor method includes the above algorithm as a special case.

2.2. The Multiple-Predictor Method

The multiple-predictor method employs more than one predictor and a predictor-selecting function g in addition to the collision-resolution function h. The difference from the single-predictor method is the number N of predictors reserved in each cell. The function g is used to determine which predictor should maintain the synonym cluster. Synonym here means keys with the same value of g as well as of the hash function H. Assume that the predictor number g(k) for a key k is determined independently of $H(k)$, and $1 \le g(k) \le N$ holds. The basic idea of the algorithm is much the same as that using a single predictor, except that the multiple-predictor algorithm uses the g(k)th predictor of each cell rather than the single one. Let us introduce a function h' using the collision-resolution function h as

$$h'(i, k) = \begin{cases} h(0, k)[=H(k)] & \text{for } i = 0 \\ h(i + \Delta, k) & \text{for } i \ge 1 \end{cases}$$

where Δ is a bias determined by g(k) as

$$\Delta = M[g(k) - 1]/N$$

First, the home address $h'(0, k)[=H(k)]$ of the key k to be searched for is computed. If $h'(0, k)$ does not contain the key k, then the second address to be checked is $h'\{L[h'(0, k), g(k)], k\}$, where $L(a, n)$ indicates the value of the nth predictor in the ath cell. In general, the next candidate address containing a synonym after checking the $h'(i, k)$th cell is given as $h'\{i + L[h'(i, k), g(k)], k\}$. It may occasionally happen that even the maximum predictor value cannot represent the number of probes to be skipped to reach the next candidate address, in which case the search must continue with one-by-one probing.

We now give algorithms to store or search for key k by using Pascal-like expressions. In the following, T, L, M, N, and max are nonlocal variables denoting the hash table, the table for multiple predictors, the table size, the number of predictors reserved per cell, and the maximum value of a predictor, respectively. $T[a]$ and $L[a, n]$ mean the key and the nth predictor corresponding to the ath cell.

2.2.1. The storing algorithm. Initially, the elements of T and L are all empty. The function h' and g, assumed to be defined outside the procedure, give the probe sequence and the predictor identifier for a given key k to be stored. The algorithm is as follows: where the variables a, kw, aw, i, n, b, q, and qw mean the home address of k, the key occupying the home address of k, the home address of kw, the position in the probe sequence, the predictor identifier ($\le N$) for k, a candidate address of an empty cell, a predictor value, and the temporary record of a predictor value used by 'updatepredictor' procedure, respectively.

```
procedure store (k:integer);
  label 1, 2;
  const empty = 0 {means empty};
  a, kw, aw, i, n, b, q, qw:integer;
  procedure updatepredictor;
    w:integer;
  begin if q > max then w := max else w := q;
        if w <> qw then L[a, n] := w
  end;
begin {main procedure}
  a := h'(0, k);
  if T[a] = empty then storeitem (a, k) {completed}
  else
    begin kw := T[a]; aw := h'(0, kw);
      if aw <> a then {displace the nonsynonym key kw}
        begin L[a, g(kw)] := 0; storeitem (a, k);
              k := kw; a := aw
        end;
```

```
{hereafter, k:the key to be stored, a:the home address of k}
    n := g(k); b := a; i := 0;
    l:q := L[b, n]; qw := q;
    if q = 0
    then {search empty cell}
      repeat q := q + 1; i := i + 1;
            if i > M then table-full else b := h'(i, k)
      until T[b] = empty
    else {trace the cluster}
      2:if i + q > M then table-full
        else begin b := h'(i + q, k);
            if h'(0, T[b]) = a and g(T[b]) = n
            then begin i := i + q; updatepredictor; goto 1
              end
            else begin q := q + 1;
                  if T[b] <> empty then goto 2
              end
          end;
      storeitem (b, k); updatepredictor {completed}
  end
end.
```

As the basic rule, starting from the home address, the cluster for the key k to be stored is traced through by using the g(k)th predictor until an empty cell is encountered. If the home address is occupied by some key whose home address is different, then that key is moved to another location (item displacement) and the predictor that pointed to the item displaced must be corrected. The procedure 'updatepredictor' is used to change the incorrect predictor.

2.2.2. The search algorithm

```
procedure search (k:integer);
  a, b, n, i:integer;
begin
  a := h'(0, k); b := a; n := g(k); i := 0;
  while T[b] <> k and L[b, n] <> 0
  {that is, not equal to k, but synonyms are not exhausted}
  do begin q := L[b, n]; i := i + q; b := h'(i, k);
    if q >= max
    then while h'(0, T[b]) <> a or g(T[b]) <> n
      [search a synonym one-by-one]
      do begin i := i + 1; b := h'(i, k) end;
    end;
  if T[b] = k then found else not-found
end.
```

Searching is much simpler than storing. The program includes two *while*-statements. The second one is executed only of $q \geq$ max holds. However, if the length of the predictor field is chosen to be more than four or five bits, such cases will be very rare. Probing is caused by evaluating the logical expression in the first of these *while*-statements. As noted earlier, the absence of the key to be retrieved is effectively treated by the final statement.

3. SEARCHING EFFICIENCY OF THE ALGORITHMS

In this section, the searching efficiency of the two methods proposed in the preceding section is analyzed in terms of the mean number of probes. First we consider the basic method using just one predictor. Let p and x be the bit length of a predictor and the load factor, respectively. Then the maximum value r of a predictor, denoted "max" in the procedures, is $2^p - 1$. Assume that each cell in the table is hit as frequently as any other. Then, the probability that i keys are hashed to any one cell is given by the Poisson approximation $P(i, x) = e^{-x}x^i/i!$.

Figure 1(a) shows the storing process for key k when the number of synonyms already stored is i, that is, the hash addresses of $k_1, \ldots, k_i$ and k are all identical. First, the tracing process of the cluster takes place, as shown by solid arrows. Then, the scanning process to find an empty cell follows, as indicated by dashed arrows.

Let us estimate the excess cost caused by those two processes shown in Figure 1 over the direct-chaining method. Starting from the last cell of a cluster, the probability that j probes are needed to find an empty cell is $x^{j-1}(1 - x)$. Whenever the number j does not exceed the maximum value r, the number of probes needed to access this key on searching is reduced to one by using the predictor. But if $j > r$, then the number of probes becomes $1 + j - r$. Therefore, the average probe number is estimated as

$$\sum_{j=0}^{r} x^j \cdot (1 - x) + \sum_{j=r+1}^{\infty} (1 + j - r)x^j(1 - x) = 1 + \frac{x^r}{1 - x}$$

Let $e_r(x)$ be the excess cost needed to traverse the gap between the two keys k_i and k compared with the cost using the chaining method. Then we have:

$$e_r(x) = \frac{x^r}{1 - x} \tag{2}$$

Next, let $1 + t_r(x)$ be the average number of probes needed to traverse between two synonym cells adjoining each other in a cluster. Since the excess cost of traversing between two keys is equal to $e_r(y)$, where y is the load factor when the second key was stored, the average excess cost $t_r(x)$ is given by integrating and averaging $e_r(y)$ as

$$\begin{aligned} t_r(x) &= \frac{1}{x}\int_0^x e_r(y)\, dy \\ &= -\frac{1}{x}\ln(1 - x) - \sum_{i=1}^{r}\frac{x^{i-1}}{i} \end{aligned} \tag{3}$$

The average traversing cost for keys placed earlier in a cluster is usually less than that for keys placed later; we do not, however, take this into consideration. The average excess cost to trace a cluster is $(i - 1)\cdot t_r(x)$, where i is the length of a cluster. Let $s_r(x)$ be the total excess cost to search a key that is stored when the load factor is x. Then, from the results (2) and (3), and by the assumption of Poisson approximation, it follows that

$$\begin{aligned} s_r(x) &= e_r(x) + \sum_{i=1}^{\infty}(i - 1)t_r(x)P(i, x) \\ &= -\ln(1 - x) - \sum_{i=1}^{r}\frac{x^i}{i} + \frac{x^r}{1 - x} - t_r(x)(1 - e^{-x}) \end{aligned} \tag{4}$$

Let $E(p, \alpha)$ denote the average number of probes needed to retrieve a key in the table when the load factor is α. Then, from Eqs. (1) and (4)

$$\begin{aligned} E(p, \alpha) &= 1 + \frac{\alpha}{2} + \frac{1}{\alpha}\int_0^{\alpha} s_r(x)\, dx \\ &= 1 + \frac{\alpha}{2} + \frac{1}{\alpha}\int_0^{\alpha} e_r(x)\, dx + \frac{1}{\alpha}\sum_{i=1}^{\infty}(i - 1) \end{aligned}$$

$$\cdot \int_0^{\alpha} \frac{P(i, x)}{x} \int_0^{x} e_r(y)\, dy\, dx$$

$$= 2 + \frac{\alpha}{2} - \ln(1 - \alpha) - \sum_{i=1}^{r} \frac{\alpha^{i-1}}{i} \cdot \left(1 + \frac{\alpha}{i + 1}\right)$$

$$- \frac{1}{\alpha} \int_0^{\alpha} t_r(x)(1 - e^{-x})\, dx \qquad (5)$$

where $r = 2^p - 1$.

We turn now to the case of multiple predictors. Before estimating the efficiency of multiple predictors, let us consider an extended-chaining method, called the *multiple-chaining method*, that uses more than one link field per cell.

Let N indicate the number of link fields associated with each cell. Each link field is used as a pointer to the next synonym. Consider the case of storing a key into the home cell, into which the storing algorithm has already attempted to store $i(\geq 1)$ keys (i.e., synonyms). Those synonyms, except for the one stored in the home cell, have been rescattered to the N lists by using another hashing function like the g used in the multiple-predictor method. The average length of each list is $(i - 1)/N$. Since a new key is stored after visiting the home address and all elements in a proper list selected by the secondary hashing function, the number of probes needed to retrieve this key later is $1 + (i - 1)/N + 1$ for $i \geq 1$. Assuming that keys are scattered to random locations of the table, the average search length for a key stored when the load factor is x is given as

$$c^N(x) = \sum_{i=1}^{\infty} \left(1 + \frac{i - 1}{N} + 1\right) P(i, x) + P(0, x)$$

$$= 2 - \frac{1}{N} + \left(\frac{1}{N} - 1\right) e^{-x} + \frac{x}{N}$$

Therefore, the average number of probes $E^N(\alpha)$ for a successful search is

$$E^N(\alpha) = \frac{1}{\alpha} \int_0^{\alpha} c^N(x)\, dx$$

$$= 2 - \frac{1}{N} + \frac{1}{\alpha}\left(\frac{1}{N} - 1\right)(1 - e^{-\alpha}) + \frac{\alpha}{2N} \qquad (6)$$

where α is the load factor. Note that when $N = 1$, $E^N(\alpha)$ is reduced to Eq. (1), which is as expected. Note further that

$$\lim_{N \to \infty} E^N(\alpha) = 2 - \frac{1}{\alpha}(1 - e^{-\alpha}) \qquad (7)$$

which gives the boundary of improvement by multiple chaining.

The efficiency of the multiple-predictor method will be proved by estimating the excess cost over the multiple-chaining method, which is similar to the method used for estimating the performance of the single-predictor method compared to the direct-chaining method. Assume each cell has N associated predictors. The excess cost of finding an empty cell is

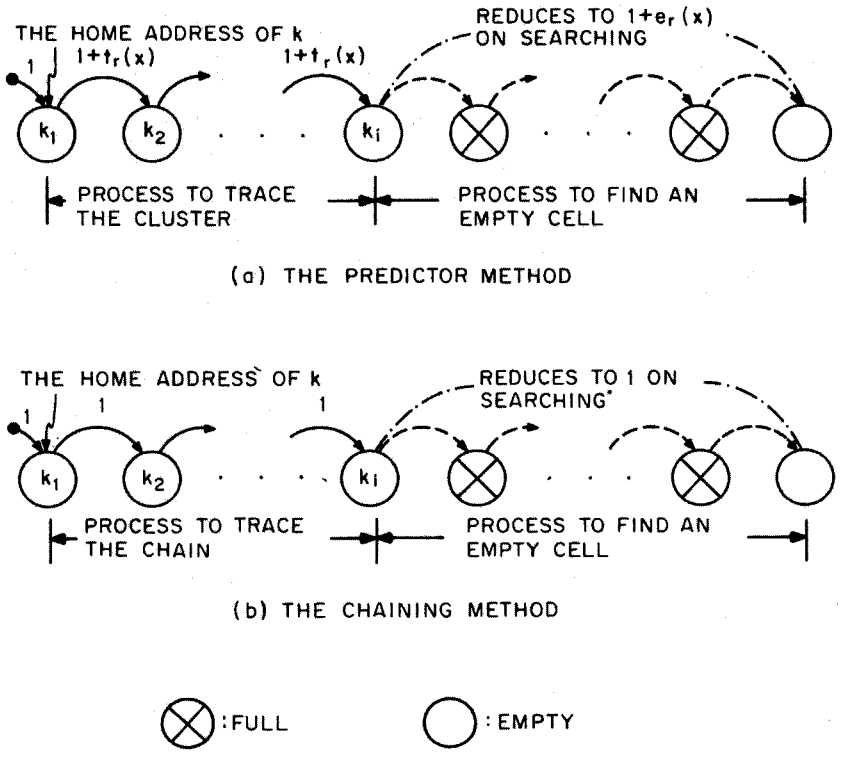

FIGURE 1. Storing process when the load factor is x.

equal to Eq. (2). By using the results of Eq. (3) and the discussion of multiple chaining, the averaged extra cost of scanning the final key in a cluster is approximated as $t_r(x)\cdot(i-1)/N$, where x is the load factor. Thus, the total excess cost $s_r^N(x)$ to search a key that was stored when the load factor was x is given as

$$s_r^N(x) = e_r(x) + \sum_{i=1}^{\infty} \frac{i-1}{N} t_r(x) \cdot P(i, x)$$

$$= -\frac{1}{N}\ln(1-x) - \frac{1}{N}\sum_{i=1}^{r}\frac{x^i}{i} + \frac{x^r}{1-x}$$

$$-\frac{1}{N}t_r(x)\cdot(1-e^{-x}) \qquad (8)$$

Let $E^N(p, \alpha)$ be the average number of probes needed for a successful search when the load factor is α. Then, from Eqs. (6) and (8)

$$E^N(p, \alpha) = E^N(\alpha) + \frac{1}{\alpha}\int_0^{\alpha} s_r^N(x)\,dx$$

$$= 2 - \frac{1}{N} + \frac{1}{\alpha}\left(\frac{1}{N} - 1\right)(1-e^{-\alpha}) + \frac{\alpha}{2N}$$

$$-\frac{1}{N\alpha}\int_0^{\alpha}\ln(1-x)\,dx$$

$$-\frac{1}{N\alpha}\sum_{i=1}^{r}\frac{1}{i}\int_0^{\alpha}x^i\,dx + \frac{1}{\alpha}\int_0^{\alpha}\frac{x^r}{1-x}\,dx$$

$$-\frac{1}{N\alpha}\int_0^{\alpha}t_r(x)\cdot(1-e^{-x})\,dx$$

$$= 2 + \frac{1}{\alpha}\left(\frac{1}{N} - 1\right)(1-e^{-\alpha}) + \frac{\alpha}{2N}$$

$$+\frac{1}{\alpha}\left(\frac{1-\alpha}{N} - 1\right)\ln(1-\alpha)$$

$$-\frac{1}{N}\sum_{i=1}^{r}\frac{\alpha^i}{i(i+1)} - \sum_{i=1}^{r}\frac{\alpha^{i-1}}{i}$$

$$-\frac{1}{N\alpha}\int_0^{\alpha}t_r(x)(1-e^{-x})\,dx \qquad (9)$$

where $r = 2^p - 1$. If we let N approach infinity, $E^N(p, \alpha)$ gives the boundary of improvement by multiple predictors whose size is p as:

$$\lim_{N\to\infty} E^N(p, \alpha) = 2 - \frac{1-e^{-\alpha}}{\alpha} - \frac{1}{\alpha}\ln(1-\alpha)$$

$$-\sum_{i=1}^{r}\frac{\alpha^{i-1}}{i} \qquad (10)$$

When $N = 1$, $E^N(p, \alpha)$ naturally reduces to Eq. (5).

The integral in the last term of Eq. (5) or Eq. (9) is easily evaluated by a standard numerical-integration method. Figure 2 shows the estimated average probe numbers for successful

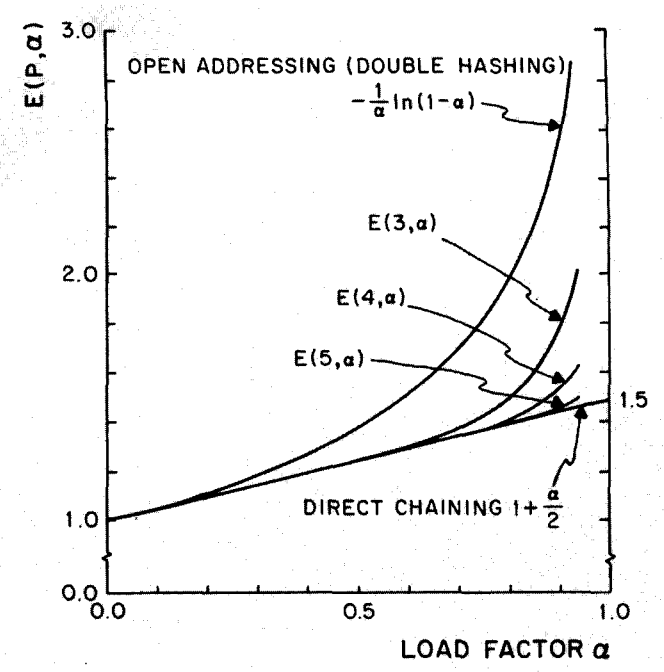

FIGURE 2. $E(p, \alpha)$ of the single-predictor method.

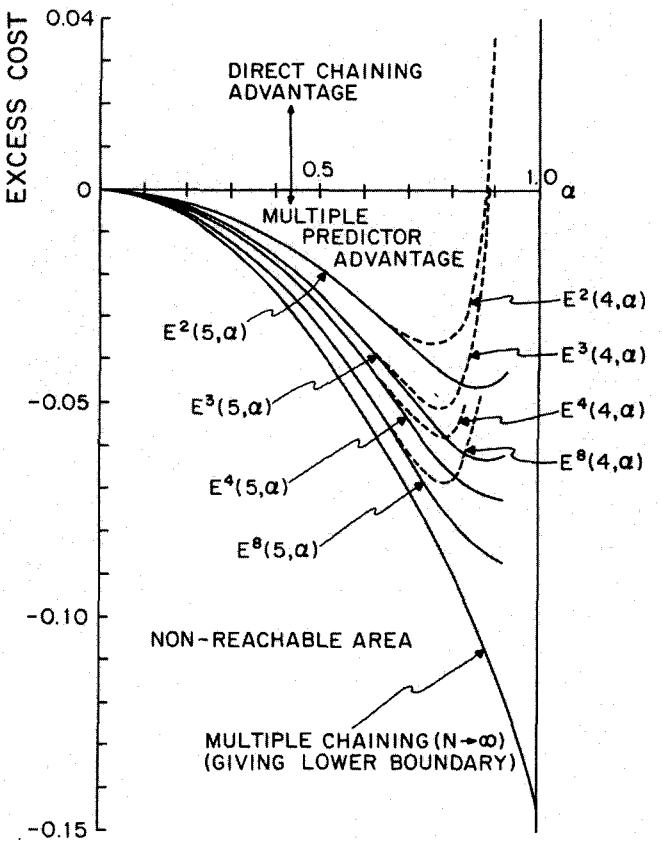

FIGURE 3. The excess cost of the multiple-predictor method over the direct-chaining method, that is, $E^N(p, \alpha) - (1 + \alpha/2)$.

TABLE I. Summary of results of simulations and theoretical values $E(p, \alpha)$ or $E^N(p, \alpha)$.

N: No. of Preds. / α: load fact.	Methods	Predictor $p=3$		$p=4$		$p=5$		Chain
		$E^N(3, \alpha)$	Observed	$E^N(4, \alpha)$	Observed	$E^N(5, \alpha)$	Observed	$E^N(\alpha)$
$N=1$ single	0.1	1.050	1.049	1.050	1.049	1.050	1.049	1.050
	0.2	1.100	1.099	1.100	1.099	1.100	1.099	1.100
	0.3	1.150	1.154	1.150	1.154	1.150	1.154	1.150
	0.4	1.200	1.203	1.200	1.203	1.200	1.203	1.200
	0.5	1.252	1.253	1.250	1.252	1.250	1.252	1.250
	0.6	1.308	1.312	1.300	1.304	1.300	1.303	1.300
	0.7	1.379	1.389	1.351	1.354	1.350	1.351	1.350
	0.8	1.498	1.521	1.409	1.412	1.400	1.398	1.400
	0.9	1.809	1.832	1.543	1.545	1.460	1.457	1.450
$N=2$	0.5	1.233	1.235	1.232	1.234	1.232	1.234	1.232
	0.6	1.282	1.282	1.274	1.275	1.274	1.275	1.274
	0.7	1.344	1.346	1.316	1.315	1.315	1.314	1.315
	0.8	1.453	1.462	1.365	1.365	1.356	1.351	1.356
	0.9	1.750	1.785	1.487	1.504	1.405	1.409	1.395
$N=3$	0.5	1.227	1.229	1.225	1.228	1.225	1.228	1.225
	0.6	1.273	1.274	1.265	1.267	1.265	1.267	1.265
	0.7	1.332	1.329	1.305	1.303	1.304	1.303	1.304
	0.8	1.438	1.434	1.350	1.384	1.341	1.336	1.341
	0.9	1.730	1.760	1.469	1.489	1.387	1.387	1.377
$N=4$	0.5	1.224	1.226	1.222	1.225	1.222	1.225	1.222
	0.6	1.269	1.271	1.261	1.263	1.261	1.263	1.261
	0.7	1.326	1.330	1.299	1.299	1.298	1.297	1.298
	0.8	1.431	1.446	1.343	1.345	1.334	1.331	1.334
	0.9	1.721	1.744	1.460	1.474	1.378	1.377	1.368
$N=6$	0.5	1.221	1.223	1.219	1.221	1.219	1.221	1.219
	0.6	1.264	1.266	1.257	1.259	1.257	1.259	1.257
	0.7	1.320	1.321	1.293	1.292	1.292	1.291	1.287
	0.8	1.423	1.425	1.336	1.335	1.327	1.322	1.326
	0.9	1.711	1.745	1.450	1.477	1.369	1.375	1.359
$N=8$	0.5	1.219	1.221	1.218	1.220	1.218	1.220	1.218
	0.6	1.262	1.263	1.255	1.257	1.255	1.256	1.255
	0.7	1.318	1.319	1.290	1.290	1.289	1.289	1.289
	0.8	1.419	1.417	1.332	1.330	1.323	1.319	1.323
	0.9	1.706	1.716	1.446	1.455	1.364	1.366	1.354
$N \to \infty$	0.1	1.048		1.048		1.048		1.048
	0.2	1.094		1.094		1.094		1.094
	0.3	1.136		1.136		1.136		1.136
	0.4	1.176		1.176		1.176		1.176
	0.5	1.215		1.213		1.213		1.213
	0.6	1.256		1.248		1.248		1.248
	0.7	1.309		1.282		1.281		1.281
	0.8	1.408		1.321		1.312		1.312
	0.9	1.691		1.432		1.350		1.341
	1.0	∞		∞		∞		1.368

Note: $E^1(p, \alpha) \equiv E(p, \alpha)$

searching by the single-predictor method. It is seen that just a four or five bit predictor field is sufficient to get efficiency very close to that of direct chaining. Figure 3 shows the excess cost of the multiple-predictor method over the direct-chaining method.

4. EXPERIMENTAL VERIFICATION

Applying our methods to the quadratic search method, which is a typical open-addressing method eliminating primary clustering, we made the following set of experiments.

Many cases of the size of a predictor field and the number of predictors were tested. Each simulation run was repeated 10 times and was averaged for a table of length 2048 using pseudorandom keys. The results obtained for each case were compared with the theoretical values, that is, $E(p, \alpha)$ or $E^N(p, \alpha)$ in Table I. Estimated efficiencies $E^N(\alpha)$ of the multiple-chaining method and the ultimate values when N approaches infinity are also listed in Table I. It is seen that the experiments give results very close to the expected values.

The greater the bit length p of each predictor field is chosen, the closer the value of $E^N(p, \alpha)$ becomes to that of chaining, that is, $E^N(\alpha)$. In the chaining method, the length of a link field must be at least $\log_2 M$ bits, where M is the table size. In general, the size of a cell of the predictor method is less than that of the chaining method. In practical usage, with respect to the space/time trade-offs, the predictor method is always preferable to the other as long as the size of each predictor field is chosen to be more than four or five bits.

In particular, when the table size is very large and the entire required bit length of the link field is used instead for multiple predictors, the expected number of probes to look up a key of the multiple-predictor method becomes less than that of the direct-chaining method, that is, $1 + \alpha/2$.

5. CONCLUSION

We have proposed two methods, the single-predictor method and the multiple-predictor method, that use several bit fields as predictors, to reduce the average number of probes necessary to search a key in a hash table. The efficiency of each method was analyzed theoretically and verified experimentally.

The single-predictor method whose predictor size is more than four or five bits is in practice preferable to the chaining method with respect to space/time trade-offs. Furthermore, when the table size is great, the multiple-predictor method gives a smaller average number of probes than that of the chaining method, that is, $1 + \alpha/2$. Notice that the length of each predictor field need not be changed even when the table size varies. The multiple-predictor method is, in a sense, an extension of the single-predictor method, similar to the double-hashing method [1, 3]. Finally, note that the two proposed methods are also very effective in reducing the reject time when the key to be retrieved does not exist in the table.

Acknowledgment. The authors would like to thank M. Mori and J. W. Higgins for their many suggestions and review of the manuscript.

REFERENCES

1. Bell, J.R. The quadratic quotient method: A hash code eliminating secondary clustering. *Comm. ACM, 13*, 2 (Feb. 1970) 107–109.
2. Halatsis, C. and Philokyprou, G. Pseudochaining in hash tables. *Comm. ACM, 21*, 7 (July 1978) 554–557.
3. Knuth, D.E. *The Art of Computer Programming, Vol. 3: Sorting and Searching*. Addison-Wesley, Reading, MA, 1973.
4. Morris, R. Scatter storage techniques. *Comm. ACM, 11*, 1 (Jan. 1968), 38–44.

CR Categories and Subject Descriptors: E.2 [**Data**]: Data Storage Representations—*hash-table representations*
General Terms: Algorithms
Additional Key Words and Phrases: hashing, scatter storage, open addressing, chaining, collision resolution, clustering, predictor, synonym, searching

Received 10/80; revised 6/81; accepted 11/82

COMING IN JANUARY

SELF-ASSESSMENT PROCEDURE XII

a self-assessment dealing with computer architecture

Robert I. Winner and Edward M. Carter