# Performance Measures for Ordered Lists in Random-Access Files

HAIM MENDELSON AND URI YECHIALI

*Tel Aviv University, Tel Aviv, Israel*

ABSTRACT. A random-access file with $N$ storage locations is considered. Records are added to the file from time to time A record with key $\varphi \in \Omega$ is hashed to storage location $F(\omega)$. A collision is resolved by the following chaining method· All records hashed to the same location are chained to each other to form an ordered list, ordered in ascending order of the keys The first record of a list is stored either at location $F(\omega)$ or at an alternative start if location $F(\omega)$ is occupied. For this process the multidimensional time-dependent generating function is derived, and the expected values of various state variables are calculated. These values are used to obtain formulas for the expected number of I/O operations needed for retrieval, addition, or updating of a record

Two measures of retrieval performance are calculated: (i) The expected number of additional probes needed to find a record in the file This measure is uniformly bounded by $\frac{1}{2}$. (ii) The expected number of additional probes required to discover that a record is not in the file. This performance measure is always smaller than the first and is uniformly bounded by $1/e$.

Addition of a record consists of three steps. (1) checking that a record with the same key does not exist in the file, (2) finding an empty location, and (3) writing the record and updating all the pointers involved The number of I/O operations needed for record addition depends on the amount of information available on the occupancy of the file For various information levels the relevant performance measures are calculated and compared

KEY WORDS AND PHRASES: hashing, collision resolution, chaining method, ordered list, multidimensional generating function, retrieval performance

CR CATEGORIES 3.72, 3 74, 4 33, 5.5

## 1. *Introduction*

Consider a random-access file with $N$ (equivalent) storage locations 1, 2, ..., $N$. Fixed length records are added to the file from time to time. Each record is identified by a key $\omega$ belonging to a key set $\Omega$. Let $\omega_k$ $(k = 1, 2, 3, \ldots)$ denote the key of the $k$th record added to the file. We assume that $\omega_1, \omega_2, \omega_3, \ldots$ is a sequence of i.i.d. random variables with an arbitrary continuous distribution. A record with key $\omega \in \Omega$ is hashed to storage location $F(\omega)$, where the hashing function $F$ is given (see [2, 3]).

Let $F_i = F(\omega_i)$ be the storage location to which the $i$th record is mapped. It follows that $F_1, F_2, F_3, \ldots$ is a sequence of i.i.d. random variables. We assume that the distribution of $F_i$ $(i = 1, 2, 3, \ldots)$ is uniform over the set of integers $A = \{1, 2, \ldots, N\}$. A *collision* occurs when two distinct records are hashed to the same location, i.e., when $F(\omega_i) = F(\omega_j)$ for $\omega_i \neq \omega_j$.

Several methods for collision resolution are known [2–6]. In this work we analyze in detail the so-called chaining method. According to this method, all records mapped to the same location are chained to each other to form an ordered list. The first record of a list is stored either at location $F(\omega)$ or, if location $F(\omega)$ is occupied, at a randomly selected



Authors' present addresses. H Mendelson, Graduate School of Management, University of Rochester, Rochester, NY 14627; U. Yechiali, Department of Statistics, Tel Aviv University, Tel Aviv, Israel.

empty location which is termed *alternative start*. The records in each list are ordered by key.

Our aim is to calculate various measures of performance for the chaining method. Retrieval performance is evaluated via two measures: (i) the expected number of probes needed to find a record in the file, and (ii) the expected number of probes required to discover that a record is not in the file.

Measures of performance for addition of a record are also derived. Record addition is composed of three steps: (1) checking that the key does not exist in the file, (2) finding an empty location, and (3) writing the record and updating all the pointers involved. The number of I/O operations needed for the second step depends on the amount of information available on the occupancy of the file. For various information levels the relevant performance measures are calculated and compared.

Updating an existing record consists of two stages: retrieving the record and rewriting it. Since the location of the record remains the same, no pointers have to be changed. Thus, the expected number of I/O operations needed is equal to the expected number of probes needed to find the record in the file plus 1.

A special case of our general problem has been studied by Johnson [1], who treated the problem of addressing on secondary keys. He derives an approximate formula for the expected number of probes for retrieval of a record in the file. An exact formula which is easily derived from our general results shows that Johnson's approximation is a good one.

The paper is composed of the following sections. Section 2 describes the chaining method and defines the underlying stochastic process. In Section 3 we derive the multidimensional time-dependent generating function of the process and calculate the moments of various state variables. In Section 4 we analyze the length of search in an ordered table. Our results are then used in Section 5 to calculate the retrieval performance measures, and in Section 6 to analyze the process of record addition.

## 2. *The Chaining Method*

Consider a random-access file with $N$ storage locations. Let $F:\Omega \to A$ be the hashing function. We assume that $F$ is a random variable uniformly distributed over the set $A = \{1, 2, \ldots, N\}$. Records are added to the file from time to time. Suppose that a record with key $\omega \in \Omega$ is to be added to the file. The record is hashed to storage location $F(\omega)$. If this location is empty, the record is stored there. If the storage location is occupied, the record has to be assigned to some empty location which is randomly selected from the set of empty locations. From symmetry considerations the actual method of assignment is irrelevant to the future development of the process, since the empty locations are interchangeable.

For retrieval purposes it is necessary to keep track of the actual addresses of the records. All records hashed to the same location are chained to each other to form an *ordered list*. Retrieval of a record requires a search along the list generated by all records mapped by $F$ to the same location. Once the beginning of the list is found, the required key is searched along the list. However, it might require additional effort to determine the beginning of the list, since the first record hashed to the chain might have found an occupied location and may have had to be assigned to an alternative address.

We start the process at time $\tau_0 = 0$ with an empty file.

Let $0 < \tau_1 < \tau_2 < \cdots < \tau_k < \cdots$ be the sequence of arrival instants of records. $\tau_k$ is the instant of arrival of the $k$th record with key $\omega_k$.

We assume that the interarrival times $\tau_k - \tau_{k-1}$ $(k = 1, 2, \ldots)$ are independent random variables and embed the process at instants $\{\tau_k + 0\}_{k=0}^{\infty}$. We say that the system is in stage (step) $k$ when there are $k$ records in the file. Storage locations are gradually numbered (for the analysis) along with the development of the process: The location occupied in the $k$th step is denoted as the $k$th storage location. That is, at the $k$th step $k$ storage locations are occupied and numbered by the numbers 1, 2, ..., $k$. Once a number is assigned to a

location, it does not change. Addition of records to the file generates lists. We say that a record with key $\omega \in \Omega$ belongs to list $\hat{F}(\omega)$, where $\hat{F}(\omega)$ is the number given by our numbering procedure to the location to which the record is hashed. The records of the same list are chained to each other by *pointers* in ascending order of their keys.

List $\hat{F}(\omega) = i$ starts either at location $i$ or at some alternative location $j$ $(j \neq i)$. It starts at location $i$ if the record that initiates the list arrives at instant $\tau_i$ and is hashed to an empty location. Otherwise, if the record is hashed to an occupied location, it is assigned to an alternative location $j$, and the list starts at an *alternative start* $j$. It follows that two pointers are required for each occupied location:

(a) A *link* to the next record in the list, if any, or an indication $\varnothing$ that the record is the last (so far) in the list.

(b) An address of the alternative start, if any, or $\varnothing$, if none exists.

Addition of a record with key $\omega$ is performed as follows. After $i = F(\omega)$ has been calculated, storage location $i$ is read and checked. If it is empty, the record is stored there, with both pointers set to $\varnothing$. Otherwise, the list is scanned to find the key with the highest value that is not greater than $\omega$. Suppose the record with this key is stored at location $j$. If this key is equal to $\omega$, then the new record is not added to the file. If the key is lower than $\omega$, an empty location is selected for the new record and is chained to the list immediately following location $j$. In the special case where $\omega$ is lower than the key of the first record of list $i$ and the list has no alternative start, the first record is displaced to an empty location, and the added record is stored in location $i$.

Let $X_i^{(k)}$ be the number of records in the file belonging to list $i$ at the $k$th step, and let $Y_i^{(k)}$ be defined as follows:

$$Y_i^{(k)} = \begin{cases} 1 & \text{if location } i \text{ is occupied by a foreign record at the } k\text{th step,} \\ 0 & \text{otherwise,} \end{cases}$$

where a foreign record is one with a key $\omega$ such that $\hat{F}(\omega) \neq i$.

It follows that whenever $Y_i^{(k)} = 1$ and $X_i^{(k)} > 0$, list $i$ has an alternative start. Also, $Y_i^{(i)} = 1$ implies $Y_i^{(k)} = 1$ for $k > i$. The chaining method is best illustrated graphically. A storage location is represented by a rectangle, shown in Figure 1. In Figure 2 we demonstrate a list (list $i$, say) with no alternative start in stage $k$. In this case $X_i^{(k)} = 3$, $Y_i^{(k)} = 0$.

Now consider list $j$ of Figure 2. If no record has been mapped to list $j$ up to this step (the $k$th), then $X_j^{(k)} = 0$; yet $Y_j^{(k)} = 1$. Suppose $X_j^{(k)} = 2$; then we get the configuration shown in Figure 3, for which $X_i^{(k)} = 3$, $Y_i^{(k)} = 0$, $X_j^{(k)} = 2$, $Y_j^{(k)} = 1$.

## 3. *The Multidimensional Time-Dependent Generating Function*

Consider the $2N$-dimensional stochastic process $\{(\underline{X}^{(k)}, \underline{Y}^{(k)}), k = 0, 1, 2, \ldots\}$ where $(\underline{X}^{(k)}, \underline{Y}^{(k)}) = (X_1^{(k)}, X_2^{(k)}, \ldots, X_N^{(k)}, Y_1^{(k)}, Y_2^{(k)}, \ldots, Y_N^{(k)})$. The process $\{(\underline{X}^{(k)}, \underline{Y}^{(k)}), k = 0, 1, 2, \ldots\}$ is Markovian, but its transition probabilities are nonstationary. For each $k$, we define the $2N$-dimensional generating function

$$G^{(k)}(\underline{z}, \underline{v}) = E(\underline{z}^{\underline{X}^{(k)}} \cdot \underline{v}^{\underline{Y}^{(k)}}), \tag{1}$$

where $\underline{z}, \underline{v} \in R^N$, $\underline{z}, \underline{v} \geq \underline{0}$ and, for nonnegative $\underline{a}, \underline{b} \in R^N$, we define $\underline{a}^{\underline{b}} \equiv \Pi_{i=1}^N a_i^{b_i}$. We derive a recursive equation for $G^{(k)}(\underline{z}, \underline{v})$.

THEOREM 1. *$G^{(k)}(\underline{z}, \underline{v})$ satisfies the following recursive equation:*

$$G^{(k+1)}(\underline{z}, \underline{v}) = (1/N)v_{k+1}\left(\sum_{i=1}^{k} z_i\right) G^{(k)}(\underline{z}, \underline{v}) + (1 - k/N)z_{k+1} G^{(k)}(\underline{z}, \underline{v}) \qquad (k = 0, 1, 2, \ldots) \tag{2}$$

*where $G^{(0)}(\underline{z}, \underline{v}) = 1$.*

PROOF. $\hat{F}(\omega_{k+1}) = i$ $(i = 1, 2, \ldots, k)$ with probability $1/N$. On the other hand, $\omega_{k+1}$

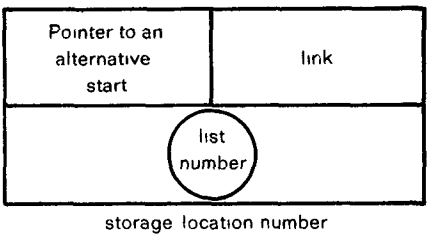


FIG 1 Graphic representation of a storage location

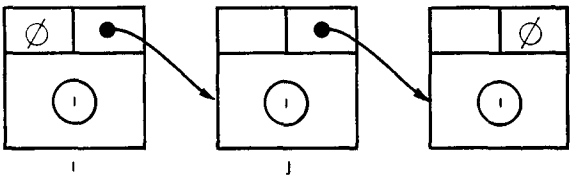


FIG 2 List with no alternative start

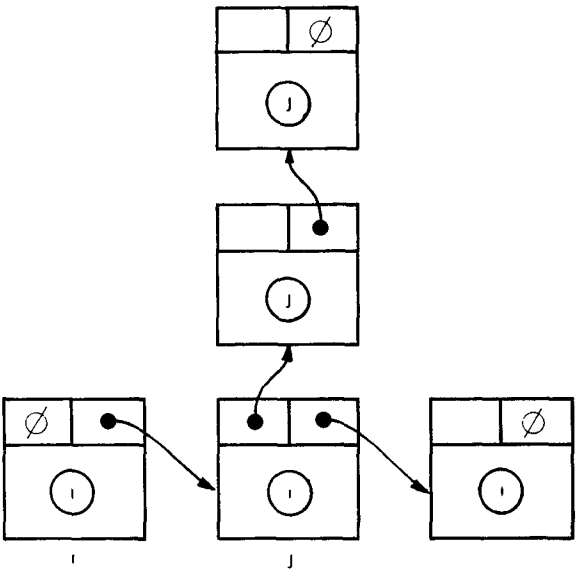


FIG 3 Lists $I$ and $J$

belongs to a nonoccupied location—which will be numbered as the $(k + 1)$st storage location—with probability (w.p.) $1 - k/N$. That is,

$$\hat{F}(\omega_{k+1}) = \begin{cases} i & \text{w p. } 1/N \\ k+1 & \text{w.p. } 1 - k/N. \end{cases} \qquad (i = 1, 2, \ldots, k),$$

If $\hat{F}(\omega_{k+1}) = i$ $(i = 1, 2, \ldots, k)$, the $(k + 1)$st record becomes a foreign record in location $k + 1$, so $Y_{k+1}^{(k+1)} = 1$. This record belongs to the $i$th list. Hence, $X_i^{(k+1)} = X_i^{(k)} + 1$. On the other hand, if $\hat{F}(\omega_{k+1}) = k + 1$, $Y_{k+1}^{(k+1)} = 0$ and $X_{k+1}^{(k+1)} = 1$, since a new list—list $k + 1$—starts at location $k + 1$. We have

$$\begin{aligned} G^{(k+1)}(\underline{z}, \underline{v}) &= \sum_{i=1}^{k} E(\underline{z}^{\underline{X}^{(k+1)}} \cdot \underline{v}^{\underline{Y}^{(k+1)}} | \hat{F}(\omega_{k+1}) = i) \cdot \mathrm{P}(\hat{F}(\omega_{k+1}) = i) \\ &\quad + E(\underline{z}^{\underline{X}^{(k+1)}} \cdot \underline{v}^{\underline{Y}^{(k+1)}} | \hat{F}(\omega_{k+1}) = k+1) \cdot \mathrm{P}(\hat{F}(\omega_{k+1}) = k+1) \\ &= \sum_{i=1}^{k} E(\underline{z}^{\underline{X}^{(k)} + \underline{e}_i} \cdot \underline{v}^{\underline{Y}^{(k)} + \underline{e}_{k+1}}) \cdot 1/N + E(\underline{z}^{\underline{X}^{(k)} + \underline{e}_{k+1}} \cdot \underline{v}^{\underline{Y}^{(k)}})(1 - k/N), \end{aligned}$$

where $\underline{e}_j$ is the unit vector with 1 in the $j$th place. Since $\underline{a}^{\underline{e}_j} = a_j$, eq. (2) readily follows. Q.E.D.

It is interesting to write specifically the generating functions $G^{(k)}(\underline{z}, \underline{v})$ for $k = 0, 1, 2$ and interpret them. $G^{(0)}(\underline{z}, \underline{v}) = 1$, since in step 0 the file is empty. $G^{(1)}(\underline{z}, \underline{v}) = z_1$. Indeed, the first record is assigned—by our numbering method—to location 1. $G^{(2)}(\underline{z}, \underline{v}) = (1/N)z_1^2 v_2 + (1 - 1/N)z_1 z_2$, for the second record is hashed to storage location 1 with probability $1/N$ and to an empty storage location with probability $1 - 1/N$.

THEOREM 2. *$G^{(k)}(\underline{z}, \underline{v})$ is a polynomial in $\{z_i\}_{i=1}^{k}$ and $\{v_i\}_{i=1}^{k}$. The maximal power of $z_i$ is $k - i + 1$ $(i = 1, 2, \ldots, k)$, and that of $v_i$ is 1.*

PROOF. By induction on $k$. Q.E.D.

Remark. Theorem 2 represents the following facts:

(i) $X_i^{(k)} = Y_i^{(k)} = 0$ for $i = k + 1, k + 2, \ldots, N$.

(ii) $Y_i^{(k)} = 0, 1$ for all $i, k$.

(iii) $X_i^{(k)} \leq k - i + 1$ for $k = 1, 2, \ldots, N$; $i = 1, 2, \ldots, k$ since list $i$ may be augmented by records only at instants $\tau_j$ for $j \geq i$.

Given the recursive equation (2), various moments, which will be needed in the sequel, may be calculated. The first- and second-order moments are given in Corollary 1.

COROLLARY 1. *For every $k = 1, 2, \ldots, N$, $i = 1, 2, \ldots, k$,*

$$E(Y_i^{(k)}) = (i - 1)/N, \tag{3}$$

$$E(X_i^{(k)}) = 1 + (k - 2i + 1)/N, \tag{4}$$

$$E((X_i^{(k)})^2) = (k - i)(3N - 3i + k + 1)/N^2 + 1 - (i - 1)/N, \tag{5}$$

$$E(X_i^{(k)} Y_i^{(k)}) = (k - i)(i - 1)/N^2. \tag{6}$$

PROOF. Differentiating eq. (2) with respect to $v_i$ and putting $\underline{z} = \underline{v} = \underline{1}$, we get $E(Y_i^{(k+1)}) = E(Y_i^{(k)})$. (The interpretation is obvious: $Y_i^{(k)}$ is determined at step $k = i$ and does not change thereafter.) Also, using the fact that $(\partial/\partial v_{k+1})G^{(k)}(\underline{z}, \underline{v}) = 0$, we have

$$E(Y_{k+1}^{(k+1)}) = \frac{\partial}{\partial v_{k+1}} G^{(k+1)}(\underline{z}, \underline{v})\Big|_{\underline{z}=\underline{v}=\underline{1}} = \frac{k}{N},$$

which implies eq. 3. In a similar way we derive $E(X_{k+1}^{(k+1)}) = 1 - k/N$. Now,

$$E(X_i^{(k+1)}) = \frac{\partial}{\partial z_i} G^{(k+1)}(\underline{1}, \underline{1}) = \frac{1}{N} + E(X_i^{(k)}).$$

Thus,

$$E(X_i^{(k)}) = \sum_{j=i}^{k-1} 1/N + E(X_i^{(i)}) = (k - i)/N + 1 - (i - 1)/N$$
$$= 1 + (k - 2i + 1)/N.$$

To get (5), we write

$$\frac{\partial^2}{\partial z_i^2} G^{(k)}(\underline{1}, \underline{1}) = \sum_{j=i}^{k-1} \left(\frac{2}{N}\right) E(X_i^{(j)}).$$

Substituting eq. (4) in the above yields (5). Finally,

$$E(X_i^{(k+1)} Y_i^{(k+1)}) = \frac{\partial}{\partial z_i} \frac{\partial}{\partial v_i} G^{(k+1)}(\underline{1}, \underline{1})$$
$$= E(X_i^{(k)} Y_i^{(k)}) + \left(\frac{1}{N}\right) E(Y_i^{(k)}).$$

Using (3) and the fact that $X_i^{(i)} Y_i^{(i)} = 0$, we get

$$E(X_i^{(k)} Y_i^{(k)}) = \sum_{j=i}^{k-1} (i - 1)/N^2 = (k - i)(i - 1)/N^2. \qquad \text{Q.E.D.}$$

The analysis of various performance measures also requires the calculation of the moment $E[(X_i^{(k)} + 1)^{-1}]$. As it turns out, the calculation of $E[(X_i^{(k)} + 1)^{-1}]$ may be reduced to the moments of the binomial distribution given by Lemma 1.

LEMMA 1. *Let the random variable $J$ have a binomial distribution with parameters $n$ (number of trials) and $p$ (probability for success): $J \sim B(n, p)$. Then*

$$E\left[\frac{1}{J+1}\right] = \frac{1 - q^{n+1}}{p(n+1)}, \tag{7}$$

$$E\left[\frac{1}{(J+1)(J+2)}\right] = \frac{1 - (n+2)pq^{n+1} - q^{n+2}}{p^2(n+1)(n+2)}, \tag{8}$$

*and*

$$E\left[\frac{1}{J+2}\right] = \frac{(n+2)p + q^{n+2} - 1}{p^2(n+1)(n+2)}, \tag{9}$$

*where $q = 1 - p$.*

Lemma 1 may be proved either directly or by integrating the generating function of $J$.

LEMMA 2. *For $k = i, i+1, i+2, \ldots,$*

$$E\left[\frac{1}{X_i^{(k)} + 1}\right] = E\left[\frac{1}{J+2}\right] + \frac{i-1}{N} E\left[\frac{1}{(J+1)(J+2)}\right], \tag{10}$$

*where $J \sim B(k - i, 1/N)$.*

PROOF. For $k = i, i+1, i+2, \ldots,$ we have

$$X_i^{(k+1)} = \begin{cases} X_i^{(k)} & \text{w.p. } 1 - 1/N, \\ X_i^{(k)} + 1 & \text{w.p. } 1/N. \end{cases}$$

It follows that for each $m = 1, 2, 3, \ldots,$

$$E\left[\frac{1}{X_i^{(k)} + m}\right] = \left(1 - \frac{1}{N}\right) E\left[\frac{1}{X_i^{(k)} + m}\right] + \frac{1}{N} E\left[\frac{1}{X_i^{(k)} + m + 1}\right].$$

By induction on $k$ we obtain,

$$E\left[\frac{1}{X_i^{(k)} + m}\right] = \sum_{j=0}^{k-i} \binom{k-i}{j} \left(\frac{1}{N}\right)^j \left(1 - \frac{1}{N}\right)^{k-i-j} E\left[\frac{1}{X_i^{(i)} + m + j}\right].$$

However, since $X_i^{(i)} = 0$ or 1,

$$\begin{aligned} E\left[\frac{1}{X_i^{(i)} + m}\right] &= \frac{i-1}{N} \cdot \frac{1}{m} + \left(1 - \frac{i-1}{N}\right) \cdot \frac{1}{m+1} \\ &= \frac{1}{m+1} + \frac{i-1}{N} \cdot \frac{1}{m(m+1)}. \end{aligned}$$

Hence,

$$E\left[\frac{1}{X_i^{(k)} + 1}\right] = \sum_{j=0}^{k-i} \binom{k-i}{j} \left(\frac{1}{N}\right)^j \left(1 - \frac{1}{N}\right)^{k-i-j} \left[\frac{1}{j+2} + \frac{i-1}{N} \cdot \frac{1}{(j+1)(j+2)}\right],$$

which is equivalent to (10). Q.E.D.

Combining the results of Lemmas 1 and 2 we obtain, after cancellations,

COROLLARY 2. *For $k = i, i+1, i+2, \ldots$*

$$\begin{aligned} &p^2 m(m+1) E[(X_i^{(k)} + 1)^{-1}] \\ &\quad = -[1 - (k+1)p] + [1 - (k+1)p][1 + mp]q^m + p^2 m(m+1)q^m, \end{aligned} \tag{11}$$

*where $p = 1/N$, $q = 1 - p = 1 - 1/N$, and $m = k - i + 1$.*

We note that the results given by Corollaries 1 and 2 may also be derived by using direct probabilistic arguments.

*4. Length of Search in an Ordered List*

To prepare for the analysis of the chaining method for random-access addressing, we study the retrieval performance of ordered lists. This study is needed since the search for a record with key $\omega$ is transformed by the hashing function into a sequential search along list number $i = \hat{F}(\omega)$. We consider an ordered list containing $n$ records (with different keys), and ignore for a moment the special features originating from our chaining method. The possible existence of an alternative start and the interaction among lists will be reconsidered in the following sections.

The (trivial) algorithm for sequential search in an ordered list is given by Knuth [2, p. 396]. It also appears as part of our flowchart in Figure 4. Obviously, ordering the list does not improve the performance of a successful search when all the keys in the list are equally likely to be requested. The expected length of a search in a list with $n$ records is equal to

$$(1/n)(1 + 2 + 3 + \cdots + n) = (n + 1)/2.$$

Consider now the case of an unsuccessful search. We assume that the keys of records added to the list, as well as the keys requested in unsuccessful searches, are sampled independently from an arbitrary continuous distribution $H(\cdot)$. Under these assumptions we prove

LEMMA 3. *The expected number of probes needed for an unsuccessful search in an ordered list with n different keys is* $1 + n/2 - 1/(n + 1)$.

PROOF. Let $\omega_i$ be the key of the $i$th record added to the list ($i = 1, 2, \ldots, n$). The random variables $\omega_1, \omega_2, \ldots, \omega_n$ are i.i.d. with distribution function $H(\cdot)$. Suppose $\omega$ is the requested key. By assumption, all $\omega_i$ ($i = 1, 2, \ldots, n$) are distinct from each other and from $\omega$. This information does not alter their joint distribution, since $H(\cdot)$ is continuous. Let $(\xi_1, \xi_2, \ldots, \xi_n)$ be the order statistics of $(\omega_1, \omega_2, \ldots, \omega_n)$, and set $\xi_0 = -\infty$; then $\xi_0 < \xi_1 < \xi_2 < \cdots < \xi_n$.

If $\xi_{j-1} < \omega < \xi_j$ ($j = 1, 2, \ldots, n$), then the number of probes needed to discover that key $\omega$ does not exist in the list is $j$. The probability of this event is

$$\begin{aligned} \mathrm{P}\{\xi_{j-1} < \omega, \xi_j > \omega\} &= \mathrm{P}\{\text{exactly } j-1 \text{ of the keys are lower than } \omega\} \\ &= \binom{n}{j-1} (H(\omega))^{j-1}(1 - H(\omega))^{n-(j-1)}, \end{aligned}$$

since $\mathrm{P}\{\omega_i < \omega\} = H(\omega)$ for $i = 1, 2, \ldots, n$. If $\omega > \xi_n$, the whole list has to be scanned, so that the number of probes is $n$. This happens with probability

$$\mathrm{P}\{\xi_n < \omega\} = \mathrm{P}\{\text{all keys} < \omega\} = (H(\omega))^n.$$

It follows that the number of probes is distributed like $\min\{J + 1, n\}$, where $J \sim B(n, H(\omega))$. Hence, the expected number of probes, given $\omega$, is

$$1 + nH(\omega) - (H(\omega))^n. \tag{12}$$

But, as is well known, $H(\omega)$ has a uniform distribution over the interval $[0, 1]$. Hence, the expected value of (12) is

$$1 + \frac{n}{2} - \int_0^1 u^n du = 1 + \frac{n}{2} - \frac{1}{n+1}. \qquad \text{Q.E.D.}$$

Remark. It is sometimes customary to facilitate the searching procedure by augmenting the list with a fictitious record whose key is "infinite." This increases the expected length of an unsuccessful search to $1 + n/2$, which is not significantly higher than $1 + n/2 - 1/(n + 1)$ when $n$ is large. This practice is not used in random-access addressing, since (1) the chains tend to be short so the loss in performance is considerable, and (2) the extra records further increase the occupancy of the file.

*5. Retrieval Performance Measures*

In what follows we incorporate our previous results to derive various performance measures for the chaining method. We restrict the analysis to input-output (I/O) considerations,

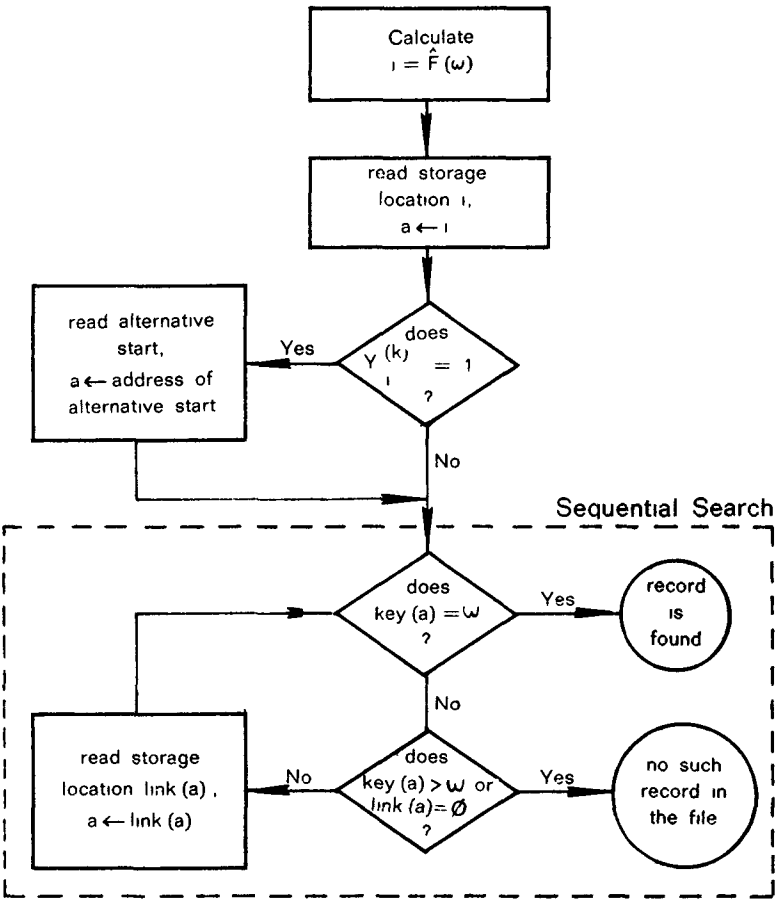


FIG 4 Flowchart for retrieval of a record with key $\omega$ The sequential search along list $i$ is shown inside the broken rectangle

which are frequently dominant in management information systems. In this section we deal with the costs associated with the retrieval of records; we measure costs by the number of probes needed for retrieval of a single record. In Section 6 we calculate the costs incurred by the addition of records to the file.

Consider a retrieval request for a record with key $\omega \in \Omega$, where $k$ records are in the file. The retrieval procedure may be described by the flowchart in Figure 4. We use the following notation:

$a$ = address of current storage location read,
$key(a)$ = key of record in storage location $a$,
$link(a)$ = address of next record in the list to which $a$ belongs.

Let $C(k)$ be the expected number of probes needed for retrieval of one of the $k$ records currently stored in the file. We assume that each of the $k$ records is equally likely to be requested. It follows that the way the records of a list are ordered does not influence $C(k)$. A record in the file belongs to list $i$ with probability $X_i^{(k)}/k$. The search for a record in list $i$ consists of (1) finding the beginning of the list, which requires $Y_i^{(k)}$ probes, and (2) moving along the list until the record is found. Since each record in the list has the same probability of being requested, the average number of moves is $(X_i^{(k)} + 1)/2$. Hence,

$$C(k) = \sum_{i=1}^{k} E\left[\frac{X_i^{(k)}}{k}\left(Y_i^{(k)} + \frac{X_i^{(k)} + 1}{2}\right)\right]. \tag{13}$$

Note that when we sum in (13) from 1 to $k$, we allow for the possibility of empty lists with $X_i^{(k)} = 0$. Since $\sum_{i=1}^{k} X_i^{(k)} = k$, we have

$$C(k) = \frac{1}{2} + \frac{1}{k} \sum_{i=1}^{k} \left[ E(X_i^{(k)} Y_i^{(k)}) + \frac{1}{2} E(X_i^{(k)})^2 \right].$$

Substituting from eqs. (6) and (5) yields

$$C(k) = 1 - \frac{1}{2NK} \sum_{i=1}^{k} (i-1) + \frac{1}{2N^2 k} \sum_{i=1}^{k} (k-i)^2 + \frac{3N-1}{2N^2 k} \sum_{i=1}^{k} (k-i).$$

By algebraic manipulation we finally obtain

THEOREM 3

$$C(k) = 1 + (k-1)(3N + k - 2)/(6N^2) \qquad (k = 1, 2, \ldots, N). \tag{14}$$

It is seen that $C(k)$ is a monotone increasing convex function of $k$ and, for fixed $k$, it is a monotone decreasing convex function of $N$.

Now suppose a record with key $\omega$, which is *not* in the file, is requested at the $k$th step. Let $D(k)$ be the expected number of probes needed to discover that the required key is not there. The key is hashed to an empty location with probability $1 - k/N$. In such an event the number of probes is one. For each $i = 1, 2, \ldots, k$, $\hat{F}(\omega) = i$ with probability $1/N$. We claim that if $\hat{F}(\omega) = i$ $(i = 1, 2, \ldots, k)$, then the expected number of I/O operations is

$$Y_i^{(k)} + 1 + \frac{1}{2} X_i^{(k)} - \frac{1}{X_i^{(k)} + 1}. \tag{15}$$

Two cases have to be considered.

(i) If $X_i^{(k)} = 0$, then (15) reduces to $Y_i^{(k)}$, which must be equal to 1. Indeed, the first probe suffices to discover that the list is empty.

(ii) If $X_i^{(k)} > 0$, then Lemma 3 may be invoked to find the expected length of search along the records belonging to list $i$. It requires $Y_i^{(k)}$ additional probes to reach the beginning of the list.

It follows that

$$D(k) = \left(1 - \frac{k}{N}\right) \cdot 1 + \frac{1}{N} \sum_{i=1}^{k} \left[ 1 + E\left[ Y_i^{(k)} + \frac{1}{2} X_i^{(k)} \right] \right] - \frac{1}{N} \sum_{i=1}^{k} E\left[ \frac{1}{X_i^{(k)} + 1} \right].$$

Substitution from (3) and (4) yields

$$D(k) = 1 + \frac{k}{2N} + \frac{k(k-1)}{2N^2} - \frac{1}{N} \sum_{i=1}^{k} E\left[ \frac{1}{X_i^{(k)} + 1} \right].$$

Using (11) and the sums

$$\sum_{m=1}^{k} \frac{1}{m(m+1)} = \sum_{m=1}^{k} \left( \frac{1}{m} - \frac{1}{m+1} \right) = 1 - \frac{1}{k+1},$$

$$\sum_{m=1}^{k} q^m = \frac{q}{p}(1 - q^k),$$

$$\sum_{m=1}^{k} \frac{1 + mp}{m(m+1)} q^m = \sum_{m=1}^{k} \left( \frac{q^m}{m} - \frac{q^{m+1}}{m+1} \right) = q - \frac{q^{k+1}}{k+1},$$

we obtain (letting, as in Corollary 2, $m = k - i + 1$):

$$\sum_{i=1}^{k} E\left[ \frac{1}{X_i^{(k)} + 1} \right] = -\frac{1}{p^2}[1 - (k+1)p] \left[ 1 - \frac{1}{k+1} \right] + \frac{1}{p^2}[1 - (k+1)p] \left[ q - \frac{q^{k+1}}{k+1} \right] + \frac{q}{p}(1 - q^k).$$

Simplifying, we have

$$-\frac{1}{N}\sum_{i=1}^{k} E\left[\frac{1}{X_i^{(k)}+1}\right] = 1 - kp - \frac{1-q^{k+1}}{p(k+1)}.$$

We have thus proved

THEOREM 4

$$D(k) = 2 - \frac{k}{2N} + \frac{k(k-1)}{2N^2} + \frac{N}{k+1}\left[\left(1-\frac{1}{N}\right)^{k+1} - 1\right]. \tag{16}$$

In order to obtain the qualitative properties of $D(k)$, we use the binomial expansion of $(1-(1/N))^{k+1}$:

$$\begin{aligned} D(K) &= 2 - \frac{k}{2N} + \frac{k(k-1)}{2N^2} + \frac{N}{k+1} \\ &\quad \times \left[-\frac{k+1}{N} + \frac{k(k+1)}{2N^2} - \frac{k(k+1)(k-1)}{6N^3} + \sum_{i=4}^{k+1}\binom{k+1}{i}\left(-\frac{1}{N}\right)^i\right] \\ &= 1 + \frac{k(k-1)}{3N^2} + \sum_{i=3}^{k}\binom{k}{i}\frac{(-1)^{i+1}}{(i+1)N^i}. \end{aligned} \tag{17}$$

It follows from (17) and from the identity

$$\binom{k+1}{i} - \binom{k}{i} = \binom{k}{i-1}$$

that

$$\Delta D(k) = D(k+1) - D(k) = \frac{2k}{3N^2} + \sum_{i=3}^{k+1}\binom{k}{i-1}\frac{(-1)^{i+1}}{(i+1)N^i} \tag{18}$$

and

$$\Delta^2 D(k) = \Delta D(k+1) - \Delta D(k) = \frac{2}{3N^2} + \sum_{i=3}^{k+2}\binom{k}{i-2}\frac{(-1)^{i+1}}{(i+1)N^i}. \tag{19}$$

Since the absolute values of the terms of (17), (18), and (19) are decreasing as $i$ increases, we have

COROLLARY 3. *$D(k)$ is an increasing convex function of $k$ and a decreasing convex function of $N$.*

In Figure 5 we illustrate the behavior of $C(k)$ and $D(k)$ for a file with $N = 1000$ storage locations.

From Figure 5 it is evident that $C(k)$ and $D(k)$ possess the following properties:

COROLLARY 4

(*i*) $C(k) \geq D(k)$, $k = 1, 2, \ldots, N$.
(*ii*) $C(k) < \frac{5}{3}$, $k = 1, 2, \ldots, N$
(*iii*) $D(k) < 1 + 1/e$, $k = 1, 2, . \ . , N$.
(*iv*) *For* $k \ll N$, $C(k) \sim 1 + (1/2N) \cdot k$.

PROOF. (i) It follows from (17) that

$$\begin{aligned} D(k) &\leq 1 + \frac{k(k-1)}{3N^2} + \binom{k}{3}\cdot\frac{1}{4N^3} \\ &= 1 + \frac{k(k-1)}{3N^2} + \frac{k(k-1)(k-2)}{24N^3}. \end{aligned}$$

Substitution of $C(k)$ from (14) yields

$$C(k) - D(k) \geq \frac{k-1}{24N^2}(7N-8) \geq 0.$$

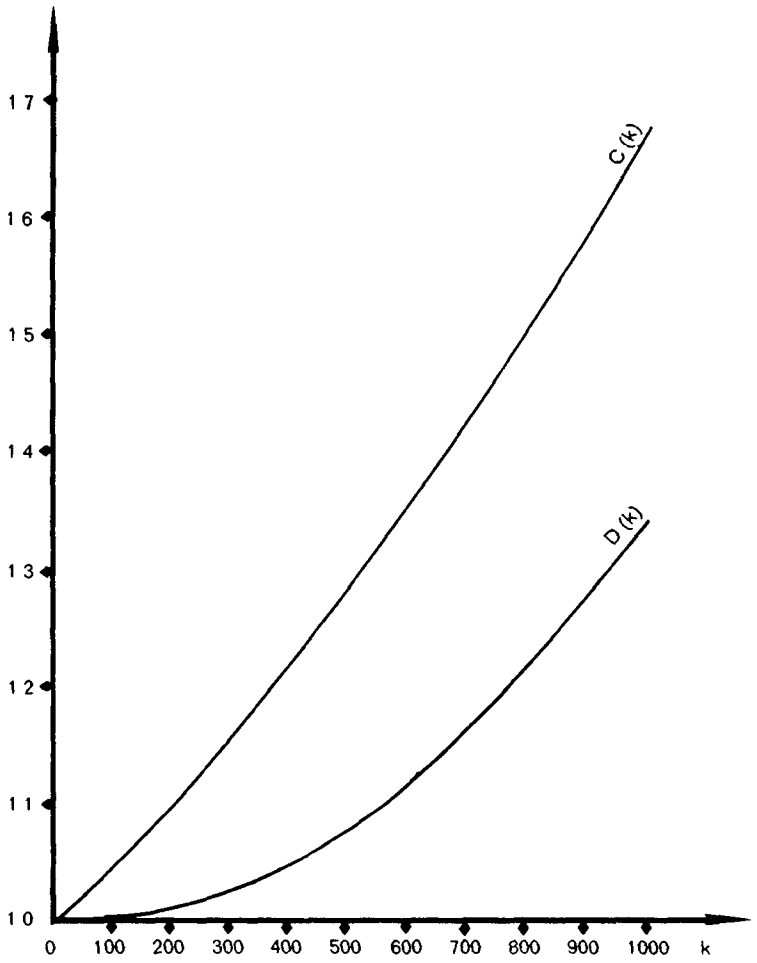


FIG. 5 Behavior of $C(k)$ and $D(k)$

(ii) From (14) we have

$$C(k) \leq C(N) = 1 + (N-1)(4N-2)/(6N^2) < \tfrac{5}{3}.$$

(iii) Use of eq. (16) with $k = N$ yields

$$D(k) \leq D(N) = 2 + \frac{N-1}{N+1}\left(1 - \frac{1}{N}\right)^N - \frac{2N^2+N+1}{2N(N+1)}.$$

It is well known that $(1 - 1/N)^N < 1/e$; hence

$$\begin{aligned} D(k) &< 2 + \left(1 - \frac{2}{N+1}\right)\cdot\frac{1}{e} - \frac{2N^2+N+1}{2N(N+1)} \\ &= 2 + \frac{1}{e} - \frac{2N(N+1) + ((4/e)-1)N + 1}{2N(N+1)} < 1 + \frac{1}{e}. \end{aligned}$$

(iv) Rewrite eq. (14) as

$$C(k) = 1 + \frac{k-1}{N}\left(\frac{1}{2} + \frac{1}{6}\frac{k}{N} - \frac{1}{3N}\right).$$

For $k \ll N$, the result follows. Q.E.D.

The above results may be compared with Johnson's work [1]. Johnson treats the problem of addressing on secondary keys, using an "indirect" chaining method. Since a record's location is a function of the primary key, a pointer to the beginning of each list is maintained. In our presentation this means that $Y_i^{(k)} = 1$ for all $i \leq k$. Obviously, this reduces the dimension and complexity of the problem.

Johnson calculates $C(k)$ using a Poisson approximation with parameter $k/N$ for the length of each list. He obtains the formula $C(k) = 2 + k/2N$.

The exact result may be obtained from eq. (13) when $Y_i^{(k)} = 1$. Substituting the values of $E(X_i^{(k)})$ and $E((X_i^{(k)})^2)$ from (4) and (5) yields

$$C(k) = 2 + (k - 1)/2N. \tag{20}$$

Comparing eqs. (20) and (14), it is clear that Johnson's method requires more I/O operations. For $k = 1$, the difference is one probe; when $k$ approaches $N$, the difference approaches $\frac{5}{6}$.

## 6. *Addition of Records*

Consider the $(k + 1)$st record arriving at instant $\tau_{k+1}$ where there are already $k$ records in the file. The addition of the record is composed of three stages: (1) checking that the key $\omega_{k+1}$ does not exist in the file, (2) searching for an empty location, and (3) storing the record and updating the relevant pointers. Each of these steps may be analyzed separately. Yet, there is some information flow from step to step.

First consider the checking procedure. This is simply a search for a record with key $\omega_{k+1}$. If $\omega_{k+1}$ exists in the file, then either the addition request is rejected, or the existing record is updated in place. The expected number of reading probes is $C(k)$; an update in place requires only one additional I/O operation, since no pointers have to be changed. In either case the addition procedure is terminated.

Now suppose that key $\omega_{k+1}$ does not exist in the file The expected number of probes needed in this case is $D(k)$. As a by-product of the search, it is known where (and how) the new record has to be chained in its list. This information is transferred onto the third stage, where the actual chaining is performed.

Next, we consider the stage of searching for an empty location. Let $i = \hat{F}(\omega_{k+1})$. If storage location $i$ is empty, the number of I/O operations needed in this step is 0 Otherwise, an alternative location has to be found.

The amount of effort needed to select the alternative location depends on the level of information available on the occupancy of the file. Three different levels of information will be analyzed:

(1) *No information.* In this case a location is selected randomly among all $N$ locations in the file. If the location so selected is empty, the record will be stored there. Otherwise, another similar independent trial is repeated until an empty location is found. Since no information is gathered during the process it might happen that an occupied location will be selected more than once. We indicate this information level with the subscript $n$.

(2) *Partial information.* Here we keep track of the locations that have been traversed and found occupied during this addition process. This includes all occupied locations encountered either during stage (1) or in previous trials of the present stage. These locations are no longer candidates for storing the $(k + 1)$st record. This information level will be indicated by the subscript $p$.

(3) *Full information.* With this level of information the addresses of all $k$ occupied locations are known. (Such information may compactly be maintained in a bit map.) The subscript $f$ will indicate this level of information.

Let $S(k)$ be the number of input operations needed for the search until an empty location is found for the $(k + 1)$st record. For each of the above levels of information, denote the expected value of $S(k)$ by $L_n(k)$, $L_p(k)$, or $L_f(k)$, respectively. We have

THEOREM 5. *For* $k = 0, 1, 2, \ldots, N-1$,

(*i*) $$L_n(k) = \frac{k}{N-k}, \tag{21}$$

(*ii*) $$L_p(k) = \frac{k+1-D(k)}{N+1-k}, \tag{22}$$

(*iii*) $$L_f(k) = k/N. \tag{23}$$

PROOF. (i) It is readily seen that $S(k)$ has a geometric distribution with a probability for success (= finding an empty location) $1 - k/N$. Hence

$$L_n(k) = \frac{k/N}{1-k/N} = \frac{k}{N-k}.$$

(ii) Let $V(k, n)$ be the expected length of a search for an empty location among $n$ equivalent locations, $k$ of which are occupied ($k = 0, 1, 2, \ldots, n-1$). $V(k, n)$ satisfies the recursive equation

$$V(k, n) = (1 - k/n)\cdot 1 + (k/n)\cdot[1 + V(k-1, n-1)],$$

or

$$V(k, n) = 1 + (k/n)V(k-1, n-1). \tag{24}$$

Note that if a solution of (24) exists, it is unique. It follows by induction on $n$ for each $k = 0, 1, 2, \ldots, n-1$ that

$$V(k, n) = (n+1)/(n+1-k).$$

Now if $\omega_{k+1}$ is hashed to an empty location, then $S(k) = 0$. If $\hat{F}(\omega_{k+1}) = i$ ($i = 1, 2, \ldots, k$), let $U_i$ be the number of probes needed for stage (1) of record addition. Then

$$E[S(k)|\hat{F}(\omega_{k+1}) = i,\ U_i = u] = V(k-u, N-u) = \frac{N+1-u}{N+1-k}.$$

It follows that

$$\begin{aligned} L_p(k) &= \left(1 - \frac{k}{N}\right)\cdot 0 + \frac{1}{N}\cdot\sum_{i=1}^{k} E\left[\frac{N+1-U_i}{N+1-k}\right] \\ &= \frac{k(N+1)}{N(N+1-k)} - \frac{1}{N+1-k}\cdot\frac{1}{N}\sum_{i=1}^{k} E[U_i]. \end{aligned}$$

But

$$D(k) = 1 - \frac{k}{N} + \frac{1}{N}\sum_{i=1}^{k} E[U_i];$$

hence,

$$L_p(k) = (k+1-D(k))/(N+1-k).$$

(iii) Location $\hat{F}(\omega_{k+1})$ is occupied with probability $k/N$. Hence

$$L_f(k) = (1 - k/N)\cdot 0 + (k/N)\cdot 1 = k/N. \qquad \text{Q.E.D.}$$

It is easy to show the following:

COROLLARY 5. $L_f(k) \le L_p(k) \le L_n(k)$.

As was expected, the number of I/O operations increases as the level of information decreases. The improvement obtained by partial information relative to no information is negligible when $k$ is not close to $N$. When $k$ approaches $N$, both $L_n(k)$ and $L_p(k)$ increase rapidly while $L_f(k)$ increases only linearly. These facts are seen in Figure 6 for the case of $N = 1000$.

We complete the analysis by calculating the number $Q(k)$ of output operations required for the last stage of record addition—i.e. for storing the record and updating the pointers

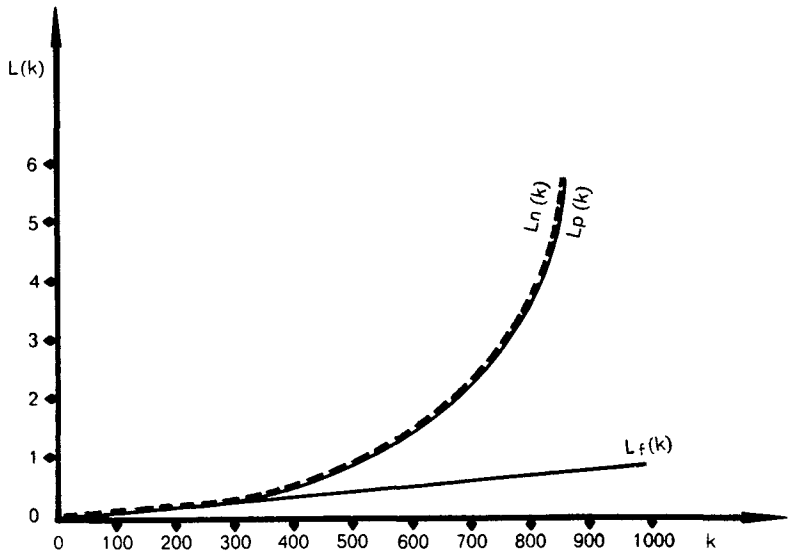


FIG 6 $L_n(k)$, $L_p(k)$, and $L_f(k)$

(all the necessary input operations have been performed in stages (1) and (2)). If location $\hat{F}(\omega_{k+1})$ is empty, all that remains is to write the added record; so $Q(k) = 1$. Otherwise, let $\iota = \hat{F}(\omega_{k+1})$ $(\iota = 1, 2, \ldots, k)$. The new record has to be chained into list $i$ following the last key with value less than $\omega_{k+1}$.

If a record with such a key exists in the list, it has been located and read in stage (1). Suppose that record was located in location $j$. Then *link*($j$) is moved to *link*($k$ + 1), *link*($j$) is set to point to the new record (at location $k$ + 1), record ($k$ + 1) is stored, and record $j$ is updated. Hence $Q(k) = 2$.

When $\omega_{k+1}$ is lower than all the keys in list $\iota$ (if any) and $Y_\iota^{(k)} = 1$, then the alternative start pointer of record $\iota$ is moved to link ($k$ + 1) and then updated to point to the new record. Locations $\iota$ and $k$ + 1 are then stored in the file; thus $Q(k) = 2$.

If $\omega_{k+1}$ is lower than all the keys in list $i$ and $Y_i^{(k)} = 0$ (this implies $X_i^{(k)} > 0$), then record $i$ is displaced to the new storage location and chained following the added record, which is stored in location $i$. Again, $Q(k) = 2$.

It follows that

$$Q(k) = \begin{cases} 1 & \text{w.p.} \quad 1 - k/N, \\ 2 & \text{w.p.} \quad k/N. \end{cases}$$

Thus, we conclude:

THEOREM 6. $E[Q(k)] = 1 + k/N$.

## REFERENCES

1 JOHNSON, L R An indirect chaining method for addressing on secondary keys *Comm. ACM 4*, 5 (May 1961), 218–222.

2 KNUTH, D E *The Art of Computer Programming, Vol 3 Sorting and Searching* Addison-Wesley, Reading, Mass., 1973

3 LUM, V Y, YUEN, P S T, AND DODD, M Key-to-address transform techniques: A fundamental performance study on large existing formatted files *Comm. ACM 14*, 4 (April 1971), 228–239

4 MORRIS, R Scatter storage techniques *Comm ACM 11*, 1 (Jan 1968), 38–44

5 PETERSON, W W Addressing for random-access storage *IBM J Res and Develop 1* (1957), 130–146

6 VAN DER POOL, J A Optimum storage allocation for initial loading of a file *IBM J Res and Develop 16* (1972), 579–586

RECEIVED JUNE 1977, REVISED SEPTEMBER 1978