Algorithmica (1986) 1: 17–29

Algorithmica

The Complexity of Hashing with Lazy Deletion

Christopher J. Van Wyk[1] and Jeffrey Scott Vitter[2,3]

Abstract. We examine a version of the dynamic dictionary problem in which stored items have expiration times and can be removed from the dictionary once they have expired. We show that under several reasonable assumptions about the distribution of the items, hashing with lazy deletion uses little more space than methods that use eager deletion. The simple algorithm suggested by this observation was used in a program for analyzing integrated circuit artwork.

Key Words. Hashing algorithms, Complexity, Lazy deletion strategies, Birth–death process, Laplace's method

1. Introduction.

A sequence of items arrives to be stored in a dynamic dictionary. Besides the key used when searching for items, each item includes two times, a *starting* time and an *expiration* time. Items arrive in order of their starting times. Each time an item arrives, any items in the dictionary whose expiration times precede the incoming item's starting time can be deleted from the dictionary: they no longer represent valid search results.

Given data structures and algorithms for maintaining and searching the dictionary, two appropriate criteria by which to judge their performance are:

- *time complexity*: the time to insert, search for, or delete an item; and
- *space complexity*: the number of items stored by the data structure.

Lower bounds on the expected or the worst-case space complexity depend on assumptions about the input data distribution. Since the time to operate on dictionary structures usually depends on the size of the dictionary, these assumptions also affect the expected and worst-case time complexity of the algorithm.

One solution to this problem that has good space complexity is to store the items in a balanced search tree and in a priority queue (see [1], for example). Many data structures can be used for the search tree and the priority queue to give worst-case time complexity that is logarithmic in the size of the dictionary. Moreover, the priority queue makes it possible to delete items from the dictionary as soon as their expiration is implied by the entry of a new item, so the dictionary is always kept as small as possible; that is, this algorithm has *optimum* space complexity.

[1] AT & T Bell Laboratories, Murray Hill, NJ 07974, USA.
[2] Department of Computer Science, Brown University, Providence, RI 02912, USA.
[3] Support for this author was provided in part by NSF research grants MCS-81-05324 and DCR-84-03613, by an IBM research contract, by an IBM Faculty Development Award, and by ONR and DARPA under Contract N00014-83-K-0146 and ARPA Order No. 4786.

Received: March 25, 1985; revised July 16, 1985, communicated by C. K. Wong.

But balanced trees only offer logarithmic time complexity; besides, they are complicated to implement and require several pointers per item. So we might decide to replace the search tree in the above solution by a separate-chaining hash table [1]. With a suitable number of buckets, this solution offers constant expected search times, although its overall time complexity is still logarithmic because of the priority queue operations. Meanwhile, the separate chains make deletion easy, so the associated priority queue still makes it possible to attain optimum space complexity.

Suppose we go one step further and eliminate the priority queue as well, relying instead on a strategy of lazy deletion. That is, we keep the chains in each hash bucket sorted by expiration time; when inserting an element, we delete any elements in its hash bucket whose expiration times precede its starting time. What are the time and space complexities of this algorithm?

The answer obviously depends on the size of the hash table. If there is only one bucket, the algorithm degenerates to storing the items in an ordered linked list, so space complexity is optimum, but time complexity is linear in the size of the dictionary. If there are as many buckets as items, expected time complexity is constant, but expected space complexity is nearly all of the items, which may well be much larger than the optimum. To summarize, unlike other applications of hashing in which one seeks to minimize the number of collisions not caused by successful searches, in hashing with lazy deletion we hope that enough collisions happen so the table is kept mostly free of expired items.

In this paper we present hashing with lazy deletion formally and analyze its complexity. Our results indicate that the expected time and space complexities are simultaneously optimum, up to a constant factor. The expected time complexity is inversely proportional to the number of hash buckets, while average space used in excess of the optimal amount is equal to the number of hash buckets.

2. Application. Suppose that two sequences of nonvertical line segments with integral endpoints are given, each sorted by the x-coordinate of their left endpoints. The problem is to report, for each line segment in one sequence, all line segments in the other sequence that intersect it with positive length (i.e., at more than a point). Hashing with lazy deletion can be used to solve this problem: the hash key is computed by taking the segment's y-intercept modulo the size of the hash table, so collinear segments hash to the same bucket. The expected time complexity is a small constant.

In one application dealing with VLSI artwork analysis [2], using hashing with lazy deletion resulted in saving both time and space over the theoretical alternatives mentioned in the introduction. The data for each segment occupies six bytes, including one byte that is used for maintaining a free list. So the only space cost of hashing with lazy deletion is for the hash table, whose size can be kept quite modest while retaining constant expected search times. On the other hand, a balanced tree and priority queues require an overhead of several pointers per segment; since the principal concern in this application was to use as little memory as possible, this was not acceptable.

3. Algorithm. The sequence of items is $\{x_i | 1 \le i \le M\}$, where each item x_i is of the form (k_i, s_i, t_i), with key k_i, starting time s_i, and expiration time t_i. We assume that items have positive lifetimes, so $s_i < t_i$ for each $1 \le i \le M$. The inequalities $s_i < s_{i+1}$ for each $1 \le i < M$ express the fact that items arrive in order of starting times.

Suppose the hash table has N buckets and uses hash function η. The insertion of element x_i proceeds as follows:

1) Compute $h = \eta(k_i)$.
2) Remove from the hash chain in bucket h any items x_j with $t_j < s_i$.
3) Add x_i so the chain in bucket h remains sorted by termination time.

The analyses below use the following quantities:

- $Need(t) = |\{x_i | s_i \le t \le t_i\}|$, the number of items that must be in the data structure at time t;
- $Waste(t) = |\{x_i | t_i < t$ and there exist no x_j such that $\eta(k_i) = \eta(k_j)$ and $t_i < s_j \le t\}|$, the number of expired items still in the dictionary at time t after any insertions required at that time have been performed; and
- $Use(t) = Need(t) + Waste(t)$, the number of items in the hash table at time t.

We also define $Need_j(t)$, $Waste_j(t)$, and $Use_j(t)$, for each $1 \le j \le N$, to count only those items in the j^{th} bucket. That is, we have

$$Need(t) = \sum_{1 \le j \le N} Need_j(t), \qquad Waste(t) = \sum_{1 \le j \le N} Waste_j(t),$$

$$Use(t) = \sum_{1 \le j \le N} Use_j(t).$$

We shall denote the expected value and standard deviation of $Need(t)$ by $E(Need(t))$ and $StDev(Need(t))$; similar notation will be used for the other quantities. Even if the hash function η is constant, which reduces to the one-bucket case, the functions *Need* and *Use* do not coincide, because expired items remain in the dictionary until a new item arrives.

We assume that each assigned hash bucket $\eta(k_i)$, for $1 \le i \le M$, is uniformly distributed in the range $1 \le j \le N$ and is independent of the other bucket assignments. For each $1 \le j \le N$, we assume that the number of items that are assigned to bucket j (i.e., $|\{x_i | \eta(k_i) = j\}|$) is Poisson distributed with mean M/N. (This implies that the number of items M is Poisson distributed; for convenience, with a slight abuse of notation, we shall denote its mean by M.) These assumptions are well justified in practice. The independence allows us to concentrate our analysis on what happens in each bucket separately.

The term $Use(t)$ is the algorithm's *space complexity* at time t. The *time complexity* at time t for all operations can be bounded by the number of elements in the accessed bucket, namely, $Use_j(t)$ when the j^{th} bucket is accessed. By our assumptions above, the random variables $Use_j(t)$, $1 \le j \le N$, are independently and identically distributed, as are the random variables $Need_j(t)$ and the random

variables $Waste_j(t)$. In particular, we have $E(Use_j(t)) = E(Use(t))/N$. The expected time and space complexities are thus $E(Use(t))/N$ and $E(Use(t))$, respectively. In the next two sections, we analyze hashing with lazy deletion by determining $E(Use(t))$.

4. Analysis—Stationary Case. Let us suppose initially that η is constant, so that essentially there is only one bucket in the hash table. We assume that the starting and expiration times of items that hash to the bucket in question are drawn from the limiting distribution of a random process. Suppose that

- the starting times $\{s_i\}$ form a Poisson process with intensity λ; i.e., the probability of an item starting in an interval of length Δt is $\lambda\,\Delta t$, as $\Delta t \to 0$; and
- the lifetimes $\{t_i - s_i\}$ are independent and exponentially distributed with expectation β; i.e., the probability that an unexpired item at time t does not expire before time $t + \Delta t$ is $e^{-\Delta t/\beta}$.

With these assumptions, the random variable $Need(t)$ corresponds to the conventional memoryless birth-death process with parameters λ, β. For a sufficiently short interval length Δt, the following transitions within a single bucket are of interest:

- no items arrive and none expire; i.e., $Need(t + \Delta t) = Need(t)$ and $Waste(t + \Delta t) = Waste(t)$;
- one item expires, i.e., $Need(t + \Delta t) = Need(t) - 1$ and $Waste(t + \Delta t) = Waste(t) + 1$; and
- an item arrives and all expired items are removed; i.e., $Need(t + \Delta t) = Need(t) + 1$ and $Waste(t + \Delta t) = 0$.

Other transitions are also possible, such as the arrival of two items or the arrival of one item and the expiration of another, but their probabilities are each $o(\Delta t)$.

Let $p_{m,n}(t) = Prob\{Need(t) = m, Waste(t) = n\}$. If we define $p_{m,n}(t) = 0$ if $\min\{m, n\} < 0$, then the following recurrence relation for $p_{m,n}$ is valid for all integers m and n:

$$\begin{aligned} p_{m,n}(t + \Delta t) = {} & \left[(1 - \lambda\Delta t)(e^{-\Delta t/\beta})^m + o(\Delta t)\right] p_{m,n}(t) \\ & + \left[(1 - \lambda\,\Delta t)(m + 1)(1 - e^{-\Delta t/\beta})\right. \\ & \qquad \left.\times (e^{-\Delta t/\beta})^m + o(\Delta t)\right] p_{m+1,n-1}(t) \\ & + \delta_{n,0}[\lambda\,\Delta t + o(\Delta t)] \sum_{j \geq 0} p_{m-1,j}(t) + o(\Delta t). \end{aligned} \tag{1}$$

By rewriting $e^{-\Delta t/\beta}$ as $1 - \Delta t/\beta + o(\Delta t)$, rearranging (1), and taking the limit as

$\Delta t \to 0$, we obtain:

$$\frac{d}{dt}p_{m,n}(t) = \left(-\lambda - \frac{m}{\beta}\right)p_{m,n}(t) + \frac{m+1}{\beta}p_{m+1,n-1}(t)$$

$$+\delta_{n,0}\lambda \sum_{j\geq 0} p_{m-1,j}(t). \tag{2}$$

In the stationary (or limiting) distribution, $p_{m,n}(t)$ is constant and independent of t, so its derivative on the left hand side of (2) is zero. To emphasize the independence of $p_{m,n}$ from t, we omit t from all terms in the rest of this section. To find the distribution $p_{m,n}$, we use generating functions. Let us define

$$P(z,w) = \sum_m \sum_n p_{m,n} z^m w^n. \tag{3}$$

Multiplying (2) by $z^m w^n$, summing over all m and n, and rewriting in terms of (3), we obtain

$$\frac{(z-w)}{\beta}\frac{\partial}{\partial z}P(z,w) = -\lambda P(z,w) + \lambda z P(z,1). \tag{4}$$

Manipulating (4) can tell us several things about *Need*, *Waste*, and *Use*.

The coefficient of z^m in $P(z,1)$ is $Prob\{Need = m\}$. Substituting 1 for w in (4) gives a differential equation whose solution is

$$P(z,1) = e^{\lambda\beta(z-1)}. \tag{5}$$

This gives us the well-known result [3, eq. XVII-(7.4)] that *Need* is Poisson distributed with

$$Prob\{Need = m\} = \frac{(\lambda\beta)^m}{m!}e^{-\lambda\beta};$$

$$E(Need) = \lambda\beta;$$

$$StDev(Need) = \sqrt{\lambda\beta}. \tag{6}$$

We are really interested in information about *Use*, and this can also be obtained from (4). The coefficient of z^k in $P(z,z)$ is $Prob\{Use = k\}$. Replacing w by z in (4) causes its left hand side to vanish. By (5), we obtain

$$P(z,z) = zP(z,1) = ze^{\lambda\beta(z-1)}, \tag{7}$$

so we find that $Use - 1$ is Poisson distributed with

$$Prob\{Use = k\} = \frac{(\lambda\beta)^{k-1}}{(k-1)!} e^{-\lambda\beta};$$

$$E(Use) = 1 + \lambda\beta;$$

$$StDev(Use) = \sqrt{\lambda\beta}. \tag{8}$$

Note that this implies that $E(Waste) = E(Use) - E(Need) = 1$.

The analysis in this section applies to *Need*, *Waste*, and *Use* for the case $N = 1$, in which there is a single hash bucket. The assumptions of independence allow us to generalize to the case $N > 1$, giving the following theorem:

THEOREM 1. When the input to hashing with lazy deletion using N hash buckets is drawn from the stationary distribution of the memoryless birth-death process defined in (1), then *Need* and $Use - 1$ are Poisson distributed with mean $\lambda\beta$; hence, we have $E(Use) = N + \lambda\beta$.

PROOF. The starting times of items assigned to bucket j are a Poisson process with intensity λ/N. As shown in Section 3, the expected time and space complexities are $E(Use)/N$ and $E(Use)$. The results (8) for the case $N = 1$ show that $Need_j$ and $Use_j - 1$ are both Poisson distributed with mean $\lambda\beta/N$. The theorem follows by independence. ■

In the VLSI application mentioned above, the value of *Need* is proportional to $\sqrt{M}$. We model this situation on the unit interval by $\lambda = M$ and $\beta = c/\sqrt{M}$, for some constant c. The analysis above shows that

$$E(Need) = c\sqrt{M};$$

$$E(Use) = N + c\sqrt{M};$$

$$StDev(Need) = StDev(Use) = \sqrt{c}\, M^{1/4}. \tag{9}$$

If we choose $N = d\sqrt{M}$, the expected time and space complexities are $1 + c/d$ and $(d + c)\sqrt{M} = E(Need)(1 + d/c)$, which are simultaneously optimum, up to a constant factor.

5. Analysis—Non-Stationary Case. Suppose the starting times s_i and expiration times t_i are in the unit interval $[0, 1]$. In some applications, the value of $Need(t)$, the number of items that must be in the data structure at time t, tends to peak at some $t \in (0, 1)$. The following two models have that property:

1) Each pair (s_i, t_i) is constructed by generating two uniform random variates x and y in the unit interval and setting $s_i := \min\{x, y\}$, $t_i := \max\{x, y\}$.

2) Each s_i is a uniform random variate in the unit interval, and the corresponding t_i is a uniform random variate in $[s_i, 1]$.

All the random variates are generated independently. Model 1 is symmetric about time $t = 1/2$; it is easy to see that $E(Need(t))$ attains its maximum value $M/2$ at time $t = 1/2$. For Model 2, which is biased toward higher values of t, $E(need(t))$ attains its maximum M/e at time $t = 1 - 1/e \approx 0.63$.

In this section we determine $E(Use(t))$ and the value t_* that maximizes $E(Use(t))$ for Model 1; the results for Model 2 are similar and are stated without proof at the end of the section. We study $Use(t)$ by decomposing it into the sum $Need(t) + Waste(t)$. Let us further decompose $Need(t)$ and $Waste(t)$ into the sum of 0–1 random variables

$$Need(t) = \sum_{1 \le i \le M} Need^i(t) \quad \text{and} \quad Waste(t) = \sum_{1 \le i \le M} Waste^i(t),$$

where $Need^i(t) = 1$ iff the i^{th} item must be in the data structure at time t (i.e., $s_i \le t < t_i$), and $Waste^i(t) = 1$ iff the i^{th} item has expired but is still in the data structure at time t. For purposes of analysis, let us suppose that the indices of the M items are randomly permuted, so that the starting times of the M items are in random order, rather than nondecreasing order. Note that the random variables $Need^i(t)$ and $Waste^i(t)$, $1 \le i \le M$, are quite different from the random variables $Need_j(t)$ and $Waste_j(t)$, $1 \le j \le N$, defined in Section 3.

The random variables $Need^i(t)$, $1 \le i \le M$, are independently and identically distributed with

$$Prob\{Need^i(t) = 1\} = E(Need^i(t)) = 2t(1 - t). \tag{10}$$

By summing, it follows that

$$E(Need(t)) = 2Mt(1 - t). \tag{11}$$

The hard part of the analysis is to get an asymptotic approximation for $E(Waste(t))$. By our randomness assumptions, the random variables $Waste^i(t)$, $1 \le i \le M$, are identically distributed, which implies that $E(Waste(t)) = M \times E(Waste^i(t))$, for any fixed i. Without loss of generality, we consider the case $i = 1$ and derive $E(Waste^1(t))$.

In our derivations we assume that t is bounded away from 0 and 1, which is justified by the fact that the area of interest is $t \approx 1/2$. For $x < t$ and $i \ne 1$, we have

$$Prob\{x \le t_1 \le x + dx\} \to 2x\,dx, \text{ as } dx \to 0; \tag{12}$$

$$Prob\{x < s_i \le t\} = (1 - x)^2 - (1 - t)^2. \tag{13}$$

Suppose that the first item has expiration time x (i.e., $t_1 = x$). The probability that the insertion of the i^{th} item into the data structure causes the first item to be

deleted before time t is the quantity given in (13) divided by N, since there is a $1/N$ chance that $\eta(k_i) = \eta(k_1)$. By independence, the probability that none of the items i, $2 \le i \le M$, causes the first item to be deleted by time t is

$$\left(1 - \frac{(1-x)^2 - (1-t)^2}{N}\right)^{M-1} \tag{14}$$

Combining (14) and (12) gives

$$\begin{aligned} E\left(Waste^1(t)\right) &= Prob\left\{Waste^1(t) = 1\right\} \\ &= \int_0^t \left(1 - \frac{(1-x)^2 - (1-t)^2}{N}\right)^{M-1} 2x\,dx \\ &= \int_{1-t}^1 \left(1 - \frac{x^2 - (1-t)^2}{N}\right)^{M-1} 2(1-x)\,dx. \end{aligned} \tag{15}$$

The last line follows by the substitution $x \leftarrow 1 - x$. The method of approximating (15) depends on the relative values of N and M.

The Case $N = o(M)$. We assume that $N < rM$ for some constant $0 < r < 1$. Laplace's Method (see [4], for example) gives an asymptotic expression for (15) that is most useful when $N = o(M)$. The derivation is quite instructive, so we shall give the basic steps, culminating in the final approximation (17). We rewrite (15) as

$$2\int_{1-t}^1 \exp\left\{(M-1)\ln\left(1 - \frac{x^2 - (1-t)^2}{N}\right)\right\}(1-x)\,dx. \tag{16}$$

The main contribution of this integral occurs in the neighborhood of $x = 1 - t$; the integrand decreases exponentially for $x > 1 - t$. By Taylor's theorem, expanding the logarithm and the $1 - x$ terms around $x = 1 - t$, we get

$$\begin{aligned} E\left(Waste^1(t)\right) = 2\int_{1-t}^1 \exp\Bigg\{(M-1)\Bigg[&-\frac{2(1-t)}{N}(x-(1-t)) \\ &-\frac{1}{N}\left(1 + \frac{2(1-t)^2}{N}\right)(x-(1-t))^2 \\ &+O\left(\frac{(x-(1-t))^3}{N^2}\right)\Bigg]\Bigg\}(t-(x-(1-t)))\,dx. \end{aligned}$$

Substituting $y \leftarrow 2(M-1)(1-t)(x-(1-t))/N$ and using the expansion e^z

$= 1 + z + O(z^2)$, for small z, gives

$$\frac{N}{(M-1)(1-t)} \int_0^{2(M-1)t(1-t)/N} \exp\left\{ -y - \frac{Ny^2}{4(M-1)(1-t)^2} \right.$$

$$\left. \times \left(1 + \frac{2(1-t)^2}{N}\right) + O\left(\frac{Ny^3}{(M-1)^2}\right) \right\} \left(t - \frac{Ny}{2(M-1)(1-t)}\right) dy$$

$$= \frac{Nt}{(M-1)(1-t)} \int_0^{2(M-1)t(1-t)/N} e^{-y} \left[1 - \frac{Ny}{2(M-1)t(1-t)} \right.$$

$$\left. - \frac{Ny^2}{4(M-1)(1-t)^2} \left(1 + \frac{2(1-t)^2}{N}\right) + O\left(\frac{N^2 y^3}{M^2}\right) \right] dy$$

$$= \frac{Nt}{M(1-t)} \left[1 - \frac{N}{2Mt(1-t)^2} + O\left(\frac{N^2}{M^2}\right) \right]. \tag{17}$$

The last line follows from the formula $\int_0^\infty y^k e^{-y}\, dy = k!$; the error incurred by extending the integrals to ∞ is $O(e^{-cM/N}) = O(N^2/M^2)$, for $N \le M$ and some constant c.

Multiplying (17) by M gives an asymptotic expression for $E(Waste(t))$. Adding the result to (11) yields a formula for $E(Use(t))$:

$$E(Use(t)) = 2Mt(1-t) + \frac{Nt}{(1-t)} \left[1 - \frac{1}{2Mt(1-t)^2} + O\left(\frac{N^2}{M^2}\right) \right]. \tag{18}$$

THEOREM 2. When the input to hashing with lazy deletion using N hash buckets is M items whose endpoints are drawn from a uniform distribution (Model 1), then the maximum value of $E(Use(t))$ is achieved at $t = t_*$, where

$$t_* = \frac{1}{2} + \frac{N}{M} + O\left(\frac{N^2}{M^2}\right); \tag{19}$$

$$E(Use(t_*)) = \frac{M}{2} + N - \frac{4N^2}{M} - \frac{N}{M} + O\left(\frac{N^3}{M^2}\right). \tag{20}$$

PROOF. We can bound $E(Use(t))$ from below and from above by replacing the $O(N^2/M^2)$ term in (18) by $c_1 N^2/M^2$ and $c_2 N^2/M^2$, respectively, for some constants $c_1 < c_2$. Let $U_i(t)$, for $i = 1, 2$, be (18) with $O(N^2/M^2)$ replaced by

$c_i N^2/M^2$. We have

$$U_1(t) \leq E(Use(t)) \leq U_2(t). \tag{21}$$

We can maximize $U_i(t)$, for $i = 1, 2$, by setting

$$\frac{d}{dt} U_i(t) = 2M - 4Mt + \frac{N}{(1-t)^2}\left[1 - \frac{3N}{2M(1-t)^2} + \frac{c_i N^2}{M^2}\right] = 0,$$

which has the solution $t = t_{U_i}$, where

$$t_{U_i} = \frac{1}{2} + \frac{N}{4M(1-t_{U_i})^2}\left[1 - \frac{3N}{2M(1-t_{U_i})^2} + \frac{c_i N^2}{M^2}\right]. \tag{22}$$

We can get an asymptotic expression for t_{U_i} by iteratively improving estimates via a bootstrapping process. To prime the pump, we know that $t_{U_i} < 1 - \varepsilon$, for some $\varepsilon > 0$. Substituting this bound into the right hand side of (22) and simplifying, we get $t_{U_i} = 1/2 + O(N/M)$. Substituting our new estimate of t_{U_i} into the right hand side, we get

$$t_{U_i} = \frac{1}{2} + \frac{N}{M} + O\left(\frac{N^2}{M^2}\right), \tag{23}$$

for $i = 1, 2$. Substituting this into the definition of $U_i(t)$, we find that $U_i(t_*)$, for $i = 1, 2$, is of the form given by the right hand side of (20). Formula (20) then follows from the two-way bound (21).

We shall prove (19) by contradiction. Suppose that (19) is not true; that is, suppose

$$t_* = \frac{1}{2} + \frac{N}{M} + \frac{c(N, M)N^2}{M^2},$$

where $c(N, M)$ is an unbounded function. Substituting this into (18), we find that $E(Use(t_*))$ is not equal to the right hand side of (20), which is a contradiction. This proves (19). ■

Note that we can compute t_{U_i} up to terms of order $O(N^3/M^3)$ by another iteration of bootstrapping; but without greater accuracy in (18), that would not give a better estimate for t_*.

The Case $N = rM$, for Any Constant $0 < r < 1$. Theorem 2 is also valid when $N = rM$, for fixed r, but we can get a better approximation using the following approach. Substituting the approximation $\ln(1 + y) = y + O(y^2)$, for small y,

into (16), we get

$$E(Waste^1(t)) = 2\int_{1-t}^{1} \exp\left\{\frac{(1-t)^2}{r} - \frac{x^2}{r} + O\left(\frac{1}{r^2M}\right)\right\}(1-x)\,dx$$

$$= 2e^{(1-t)^2/r}\left(\int_{1-t}^{1} e^{-x^2/r}\,dx - \int_{1-t}^{1} xe^{-x^2/r}\,dx\right)\left[1 + O\left(\frac{1}{r^2M}\right)\right]$$

$$= 2e^{(1-t)^2/r}\left[\sqrt{\pi r}\,\mathfrak{N}\left((1-t)\sqrt{2/r}, \sqrt{2/r}\right)\right.$$

$$\left. + \frac{r}{2}\left(e^{-1/r} - e^{-(1-t)^2/r}\right)\right]\left[1 + O\left(\frac{1}{r^2M}\right)\right], \tag{24}$$

where $\mathfrak{N}(a, b)$ is the normal distribution function $1/\sqrt{2\pi}\int_a^b e^{-x^2/2}\,dx$. Let $f(t)$ denote the expression given in (24) without the $(1 + O(1/(r^2M)))$ term; that is, $E(Waste^1(t)) = f(t)(1 + O(1/(r^2M)))$. We define $t' = \min\{1.0, t_f\}$, where $t_f > 0.5$ is defined implicitly by the formula

$$f(t_f) = \frac{N}{M} = r. \tag{25}$$

We have $t_f < 1$ when $r < 0.84$. If we multiply formula (24) by M, we get an asymptotic expression for $E(Waste(t))$. Adding (11) to the result gives us $E(Use(t))$. By techniques similar to those we used for proving Theorem 2, we can show that $E(Use(t))$ is maximized at time $t = t_*$, where

$$f(t_*) = r\left[1 + O\left(\frac{1}{r^2M}\right)\right];$$

$$t_* = t' + O\left(\frac{1}{rM}\right).$$

Substituting our results, we find that

$$E(Waste(t_*)) = f(t_*)M\left[1 + O\left(\frac{1}{r^2M}\right)\right]$$

$$= rM\left[1 + O\left(\frac{1}{r^2M}\right)\right]; \tag{26}$$

Hence, $E(Waste(t)) \sim rM = N$ when $E(Use(t))$ is maximized. Adding $E(Need(t_*))$ to (26) gives the final result:

THEOREM 3. When M items whose endpoints are drawn from a uniform distribution (Model 1) are input to hashing with lazy deletion using $N = rM$ hash buckets, then the maximum value of $E(Use(t))$ is achieved at $t = t_*$, where

$$t_* = t_f + O\left(\frac{1}{rM}\right);$$

$$E(Use(t_*)) = 2Mt_*(1 - t_*) + rM\left[1 + O\left(\frac{1}{r^2M}\right)\right],$$

where t_f is defined implicitly by (25).

Comparing this with (20) for the case $N = rM$, for fixed r, reveals that the error term $O(1/r)$ is much smaller for large M than the $O(M)$ error term from our previous approach. For $r = 0.1$, we have $t_* \sim t_f \approx 0.59$; hence, $E(Use(t_*)) \approx 2M(0.59)(0.41) + 0.1M \approx 0.584M$.

Model 2. Model 2 can be analyzed in a similar way. We have

$$E(Need(t)) = -(1 - t)\ln(1 - t)M;$$

$$E(Waste(t)) = M\int_{1-t}^{1}\left(1 - \frac{x - (1 - t)}{N}\right)^{M-1}(-\ln x)\,dx. \tag{27}$$

By Laplace's Method and analysis similar to that above, we find for $N = o(M)$ that

$$t_* \sim 1 - \frac{1}{e};$$

$$E(Need(t_*)) \sim \frac{M}{e};$$

$$E(Waste(t_*)) \sim N. \tag{28}$$

The techniques for the case $N = rM$, for fixed r, can also be applied to get similar results.

6. Conclusion. We have shown that under several different assumptions about the input data distribution, the average excess space used by hashing with lazy deletion is equal to the number N of hash buckets. This makes hashing with lazy deletion a practical algorithm even when space complexity is as much of concern as time complexity.

Hashing with lazy deletion is a simple algorithm for dynamic dictionaries whose occupants expire spontaneously. It is also compatible with the need to accommodate explicit user requests to delete elements. Even if the expiration

times of active occupants change, one can use hashing with lazy deletion: if expiration times only increase, they can be changed by moving items further down hash chains; if they may also decrease, they may need to move back in their hash chain or even be deleted immediately. (An alternative would be to forget about keeping the hash chains in expiration-time order; this would make updating expiration times simpler, and would not change the asymptotic time complexity.) Expired occupants cannot be resurrected, though, except by explicit re-insertion.

In this paper we have derived expressions for $E(Use(t))$ and $\max_t\{E(Use(t))\}$. An interesting open problem is to compute $E(\max_t\{Use(t)\})$, which corresponds to the total space complexity of hashing with lazy deletion over the course of the algorithm; this has direct relevance to queueing theory. In typical applications of hashing with lazy deletion, we have $E(\max_t\{Use(t)\} \approx \max_t\{E(Use(t))\}$.

One approach to getting good upper and lower bounds on $E(\max_t\{Use(t)\})$ is to study a potentially easier problem. Starting from equilibrium at $t = 0$ in the stationary model for the one-bucket case, we have $\max_{s_j \le t \le T}\{Use_j(t)\} = \max_{s_j \le t \le T}\{Need_j(t)\}$, for $1 \le j \le N$, where $s_j > 0$ is the minimum positive starting time over all intervals that hash to bucket j, and T is positive and finite. We believe that $E(\max_{0 \le t \le T}\{Use_j(t)\}) \sim E(\max_{0 \le t \le T}\{Need_j(t)\})$ as T gets large. If this were true, it would suffice to compute $E(\max_{0 \le t \le T}\{Need_j(t)\})$, because multiplying this by N would give an asymptotic upper bound on $E(\max_{0 \le t \le T}\{Use(t)\})$.

With respect to obtaining a lower bound, note that if we can evaluate $E(\max_{0 \le t \le T}\{Need_j(t)\})$, then we can also evaluate $E(\max_{0 \le t \le T}\{Need(t)\})$, since $Need_j$ and $Need$ are both birth-and-death processes, with intensities λ/N and λ, respectively. If the above conjecture is true, then $E(\max_{0 \le t \le T}\{Need(t)\})$ gives a tight lower bound on $E(\max_{0 \le t \le T}\{Use(t)\})$. A similar approach could be used for the non-stationary models.

Acknowledgements. Thanks to Don McClure, John Morrison, and Andrew Odlyzko for interesting discussions. Don also suggested looking at the stationary model. Thanks also to Brenda Baker, Brian Kernighan, Ian Munro, and an anonymous referee for helpful comments on the manuscript.

References

1. Donald E. Knuth, *Sorting and Searching*, Addison-Wesley, Reading, Massachusetts (1973). Volume 3 of *The Art of Computer Programming*.
2. Thomas G. Szymanski and Christopher J. Van Wyk, "Space-efficient algorithms for VLSI artwork analysis," *Proc. IEEE Design Automation Conference*, pp. 743–749 (June, 1983).
3. William Feller, *An Introduction to Probability Theory and Its Applications*, John Wiley & Sons, New York (third edition 1968).
4. Nicolaas G. de Bruijn, *Asymptotic Methods in Analysis*, Dover Publications, New York (1981).