TECHNICAL REPORT NO. 260

The Analysis of Hashing with Lazy Deletions

by

Pedro Celis and John Franco

Revised: November, 1989

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY

Bloomington, Indiana 47405-4101

The Analysis of Hashing with Lazy Deletions*

Pedro Celis, John Franco

Department of Computer Science,

Indiana University, Bloomington Indiana 47405

November 8, 1989

(revised)

Abstract

We present new, improved algorithms for performing deletions and subsequent searches in hash tables. Our method is based on open addressing hashing extended to allow efficient reclamation of unoccupied space due to deletions which enables dynamic shortening of probe sequence lengths. We present an analysis on the number of table probes required to locate an element in the table. Specifically, we present a formula which bounds the average number of cells visited during searches of a data element over its lifetime assuming a system in equilibrium. The formula is a function of the probability that an accessed element is deleted and is exact at the extreme points when the probability is 0 and 1. In the case that the probability is 0 and the load factor is α, the number of cell visits per search access is $-\ln(1-\alpha)/\alpha$, and in the case that the probability is 1 the number of cell visits per search access is $1/(1-\alpha)$.

*This research was supported in part by the Air Force Office of Scientific Research under grant numbers AFOSR 84-0372 and AFOSR 89-0186

1 Introduction

Hashing or scatter table storage [5] is an efficient data structure for the dictionary (*insert, delete, member*) problem. The load factor α of a table is defined as the ratio of the number n of records or elements in the table divided by the number m of elements the table can hold. The advantage of hashing schemes is that if $\alpha < 1$, the average cost of performing any of the dictionary operations is bounded from above by a function of α only and is independent of the values of n and m. This compares favorably with tree-like structures and other key comparison based schemes where the average cost of the dictionary operations is $\Omega(\log n)$.

The idea of hashing is to obtain a memory address or cell where an element is stored by applying a *hash function* to the key of the element. The candidate cell provided by the hash function is not affected by the presence or absence of any other key values in the table.

When two elements hash to the same cell we say a collision has occurred. Even if the hash function uniformly distributes all possible key values among all the available cells, collisions are likely to occur if α is a fixed constant between 0 and 1.

There are two general approaches for resolving collisions, namely chaining and open addressing. In chaining, cells are part of a data structure (usually a linked list) where all elements with the same hash value (called synonyms) are stored. There are many variants and interesting approaches in collision resolution by chaining. Our interest lies in a technique that requires no additional storage, namely open addressing.

In open addressing, collisions are resolved by rehashing one of the colliding elements. The hash function must therefore be augmented to give, for each key value, not one but a *probe sequence* of cells to be used during queries, insertions, and deletions. The probe position of an element is the cell where the element is currently stored.

There are several techniques used to generate probe sequences. The most effective is perhaps Double Hashing where the hash function is

$$H(K,i) = (h_1(K) + (i-1) * h_2(K)) \mod m,$$

h_1 and h_2 are auxiliary hash functions, and the m cells are numbered from 0 to $m-1$. The function h_1 gives the first choice (probe position 1) for the key K and the function h_2, usually referred to as the step function, gives the value that must be added to the current cell address to give the next cell in the probe sequence. As long as $h_2(K)$ is relatively prime to m, then the first m cells of the (arbitrarily long) probe sequence will form a permutation of the numbers 0 to $m-1$.

Analyzing the performance of open addressing with Double Hashing is difficult [2],[4]. A method that closely resembles the performance of Double Hashing but is much simpler to analyze is Random Probing [3]. In Random Probing each cell of the probe sequence for a key is simply a random number between 0 and $m-1$. Hence an element could be probed twice before another element is probed for the first time.

Peterson [5] proved that after a large number of insertions and deletions (maintaining the load factor at constant α) the average number of rehashes of a successful query is $1/(1-\alpha)$. This cost can be improved by dynamically shortening probe sequences. This can be accomplished by moving elements when cells become vacant due to deletions. Unfortunately, the overhead of determining which elements to move is prohibitive. However, probe sequences can be shortened lazily with very little overhead as follows. When an element is searched, if there exists a cell in its probe sequence that is unoccupied, then the element is moved to the earliest such cell in the sequence. In this paper we examine the cost of open addressing hashing using this lazy method of dynamically shortening probe sequences.

2 Definitions and Notation

A hash table is assumed to consist of m cells. At any moment, a cell may contain a data item called a *record* or *element*. Such a cell is said to be occupied. A cell not containing an element is said to be unoccupied. The following operations may be applied to the hash table:

- insert an element into an unoccupied cell (the cell becomes occupied)
- delete an element from an occupied cell (the cell becomes unoccupied)
- query and perhaps relocate an existing element

If an element is either deleted or queried we say that the element is accessed.

When an element E is accessed or inserted, one or more cells may be visited in addition to the cell containing E. The sequence of cells visited before and including accessing or inserting E is called the *probe sequence* of E. The number of cells in the probe sequence of E is its *probe sequence length*. The probe sequence of E may change after E is accessed or inserted because E may be moved to an unoccupied cell closer to the beginning of its probe sequence. Hence, we define $\mathbf{ps}_i(E)$ and $\mathbf{psl}_i(E)$, $i > 0$, to be the probe sequence and probe sequence length at the i^{th} access of element E. Also, we define $\mathbf{ps}_0(E)$ and $\mathbf{psl}_0(E)$ to be the probe sequence and probe sequence length when E is inserted into the hash table. In what follows the dependence on E will be dropped since it will be clear which E is being referred to.

We will be interested in the probabilities of several events associated with the probe sequence of an element. Let r be a hash table cell in $\mathbf{ps}_0$. Denote by $X_i^{(r)}(E)$ the event that r is occupied by some element other than E when E is queried for the i^{th} time. Denote by $Y_i(E)$ the event that the number of accesses of E during its lifetime is at least equal to $i + 1$. After an element E is inserted into the hash table, a number of table operations are performed until E is visited for the first time and accessed. Denote by $Q_j^{(s)}(E)$ the event that the s^{th} cell in $\mathbf{ps}_0$ is occupied when the j^{th} operation after the insertion of E is performed. Denote by $R_r(E)$ the event that the element in cell r is relocated given that it is queried on the next operation. As before, the dependence on E will be dropped for simplicity. In the case of $R_r(E)$, the dependence on r will also be dropped.

The search cost of an element is the number of cells visited during its lifetime. We are interested in determining the expected search cost for all elements in the table when the table reaches equilibrium. It is sufficient to compute the expected search cost of a random element during its lifetime. Let S be a random variable equal to the search cost of a random element during its lifetime. Let $\mathbf{l}$ be a random variable equal to the number of times a random element is queried during its lifetime. For a particular random element we can write $S = \sum_{i=1}^{\infty} \mathbf{psl}_i$ where $\mathbf{psl}_i = 0$ if $i > \mathbf{l} + 1$. Then

$$\begin{aligned}\bar{S} = \mathrm{E}[S] &= \mathrm{E}[\textstyle\sum_{i=1}^{\infty} \mathbf{psl}_i] = \displaystyle\sum_{i=1}^{\infty} \mathrm{E}[\mathbf{psl}_i] \\ &= \sum_{i=1}^{\infty} (\mathrm{E}[\mathbf{psl}_i \mid Y_i]\Pr\{Y_i\} + \mathrm{E}\left[\mathbf{psl}_i \mid \bar{Y_i}\right]\Pr\left\{\bar{Y_i}\right\}) \\ &= \sum_{i=1}^{\infty}\sum_{k=0}^{\infty} \Pr\{\mathbf{psl}_i > k \mid Y_i\}\Pr\{Y_i\}. \qquad (1)\end{aligned}$$

Equation (1) gives us a way to compute $\bar{S}$. We are also interested in computing the average number of visits per access. The expected value we are seeking is then

$$\sum_{l=1}^{\infty} \mathrm{E}[\mathbf{psl}_1 + \ldots + \mathbf{psl}_l \mid \mathbf{l} = l]/l \ \Pr\{\mathbf{l} = l\}.$$

However, this is hard to compute. Therefore, we use the simple measure

$$\bar{S}/(\mathrm{E}[\mathbf{l}] + 1)$$

instead. The difference between the two is small and the simple measure is enough to afford valuable insight into the behavior of our hashing technique.

3 Analysis of the Method

An unbounded sequence of operations shall be applied to the table for the purposes of analysis. Operations will be modeled by a series of dart throws or *hits*, as follows:

Randomly choose one of the m cells.

- If the cell is occupied,
  - with probability P_d delete the element in that cell.
  - with probability $1 - P_d$ query and perhaps relocate the element in that cell.
- If the cell is unoccupied,
  - with probability P_i insert an element there using a random hashing function.

It should be pointed out that, in the model above, we are choosing a cell which either contains an element that is to be queried or is (possibly) about to contain such an element. In other words, the cell chosen in the model above is not the beginning of a probe sequence of the interesting element. When an element is queried or inserted it is understood that the cells of the probe sequence of that element are visited first, the probe sequence having been constructed by a uniform hash function.

We must relate P_d and P_i in a special way in order to maintain equilibrium. If on a hit the table load factor is α, a delete will occur with probability αP_d, a search with probability $\alpha(1 - P_d)$, an insertion with probability $(1 - \alpha)P_i$, and no operation with probability $(1 - \alpha)(1 - P_i)$. We choose P_i and P_d such that, over a long sequence of hits, the table occupancy reaches an equilibrium load factor α. The table will be in equilibrium only if the rate at which deletions occur equals the rate at which insertions occur. Thus, we have

$$P_d = \frac{1 - \alpha}{\alpha} P_i.$$

To obtain the results of this section we assume that the following two properties hold.

Property 1: No element in the table has a probe sequence that intersects the probe sequence of designated element E in more than one cell.

Property 2: During the lifetime of element E the load factor is equal to α.

In the appendix we prove that these assumptions hold with probability tending to 1 as m goes to infinity. Property 1 implies that insertions and deletions in cells in the probe sequence of E are independent of each other, and property 2 implies that transitions for a location from occupied to unoccupied and vice versa are essentially constant for a table in equilibrium.

We wish to find $\bar{S}$ and E$[l]$. It is easy to dispense with E$[l]$ first.

Theorem 1

$$\mathrm{E[l]} = (1 - P_d)/P_d.$$

Proof: Every time E is accessed it will be queried with probability $(1 - P_d)$ and deleted otherwise. Hence **l** has a geometric distribution. □

Next, we build toward an expression for $\bar{S}$.

Lemma 1 *Let r be a cell in* $\mathbf{ps}_0$. *Then*

$$Pr\left\{X_{i+1}^{(r)} \mid X_i^{(r)}, Y_i\right\} = Pr\left\{X_i^{(r)} \mid X_{i-1}^{(r)}, Y_i\right\}$$

.

Proof: Follows from the independence of insertions and deletions due to property 1 and the constant transition probabilities due to property 2. □

In other words, the probability that an occupied cell on the i^{th} access of E is unoccupied on the $(i+1)^{st}$ access of E is not affected by the fact that the cell is in the probe sequence of E, and the number of times E has been accessed. Lemma 1 holds for all cells in the probe sequence of E and does not depend on r nor on the position of r in the probe sequence of E.

Lemma 2 *Let $r_1, r_2, \ldots, r_k$ be the first k cells in the probe sequence of element E, on its i^{th} access. Then*

$$Pr\Big\{\bigcap_{1\le l\le k} X_i^{(r_l)} \mid \bigcap_{1\le l\le k} X_{i-1}^{(r_l)}, Y_i\Big\} = Pr\Big\{\bigcap_{1\le l\le k} X_{i-1}^{(r_l)} \mid \bigcap_{1\le l\le k} X_{i-2}^{(r_l)}, Y_i\Big\}.$$

Proof: From Lemma 1 we write

$$\prod_{1\le l\le k} \Pr\Big\{X_i^{(r_l)} \mid X_{i-1}^{(r_l)}, Y_i\Big\} = \prod_{1\le l\le k} \Pr\Big\{X_{i-1}^{(r_l)} \mid X_{i-2}^{(r_l)}, Y_i\Big\}$$

If at least k cells are in $\mathbf{ps}_i$ then $\mathbf{psl}_i > k$ so $r_1, r_2, \ldots, r_k$ are occupied right before and after the $(i-1)^{st}$ query of element E (otherwise element E would have been placed in the first unoccupied location among these). The probability that a cell has gone from occupied at the $(i-1)^{st}$ query to unoccupied at the i^{th} query is the same for all cells r in $\{r_1, \ldots, r_k\}$. Furthermore because of property 2, the probability that a particular cell r in $\{r_1, \ldots, r_k\}$ has gone from occupied to unoccupied is independent of the state of the other cells in $\{r_1, \ldots, r_k\}$. Thus

$$\begin{aligned}
\prod_{1\le l\le k} \Pr\Big\{X_i^{(r_l)} \mid X_{i-1}^{(r_l)}, Y_i\Big\} &= \prod_{1\le l\le k} \Pr\Big\{X_i^{(r_l)}, X_{i-1}^{(r_l)} \mid Y_i\Big\} / \Pr\Big\{X_{i-1}^{(r_l)} \mid Y_i\Big\} \\
&= \Pr\Big\{\bigcap_{1\le l\le k} X_i^{(r_l)}, X_{i-1}^{(r_l)} \mid Y_i\Big\} / \Pr\Big\{\bigcap_{1\le l\le k} X_{i-1}^{(r_l)} \mid Y_i\Big\} \\
&= \Pr\Big\{\bigcap_{1\le l\le k} X_i^{(r_l)} \mid \bigcap_{1\le l\le k} X_{i-1}^{(r_l)}, Y_i\Big\}.
\end{aligned}$$

Similarly,

$$\prod_{1\le l\le k} \Pr\Big\{X_{i-1}^{(r_l)} \mid X_{i-2}^{(r_l)}, Y_i\Big\} = \Pr\Big\{\bigcap_{1\le l\le k} X_{i-1}^{(r_l)} \mid \bigcap_{1\le l\le k} X_{i-2}^{(r_l)}, Y_i\Big\}.$$

The lemma follows. □

Lemma 3

$$Pr\{\mathbf{psl}_i > k \mid Y_i\} = (Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\})^i \, \alpha^k.$$

Proof:

$$\begin{aligned}\Pr\{\mathbf{psl}_i > k \mid Y_i\} &= \Pr\Big\{\mathbf{psl}_i > k, \mathbf{psl}_{i-1} > k \mid Y_i\Big\} \\ &= \Pr\Big\{\mathbf{psl}_i > k \mid \mathbf{psl}_{i-1} > k, Y_i\Big\}\Pr\Big\{\mathbf{psl}_{i-1} > k \mid Y_i\Big\} \\ &= \Pr\{\mathbf{psl}_0 > k \mid Y_i\} \prod_{1 \le j \le i} \Pr\Big\{\mathbf{psl}_j > k \mid \mathbf{psl}_{j-1} > k, Y_i\Big\}.\end{aligned}$$

From Lemma 2

$$\Pr\Big\{\mathbf{psl}_{i+1} > k \mid \mathbf{psl}_i > k, Y_i\Big\} = \Pr\Big\{\mathbf{psl}_i > k \mid \mathbf{psl}_{i-1} > k, Y_i\Big\}.$$

Thus,

$$\begin{aligned}\Pr\{\mathbf{psl}_i > k \mid Y_i\} &= (\Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k, Y_i\})^i \Pr\{\mathbf{psl}_0 > k \mid Y_i\} \\ &= (\Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\})^i \Pr\{\mathbf{psl}_0 > k\}.\end{aligned}$$

Since $\mathbf{psl}_0$ has a geometric distribution, its cumulative distribution is $\Pr\{\mathbf{psl}_0 > k\} = \alpha^k$. Substituting, the lemma follows. □

We now find bounds for $\Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\}$. Let random variable $\mathbf{q}_j$ be the position of the first unoccupied cell in $\mathbf{ps}_0$ when j operations have been applied to the hash table since the insertion of E. Let random variable $\mathbf{r}$ be the number of operations applied to the hash table from the insertion of E until E is either searched for the first time, or deleted. Then

$$\begin{aligned}\Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\} &= \sum_{j=1}^{\infty} \Pr\Big\{\mathbf{q}_j > k,\ \mathbf{r} = j \mid \mathbf{q}_0 > k\Big\} \\ &= \sum_{j=1}^{\infty} \Pr\Big\{\mathbf{q}_j > k \mid \mathbf{q}_0 > k, \mathbf{r} = j\Big\}\Pr\{\mathbf{r} = j \mid \mathbf{q}_0 > k\}\end{aligned}$$

The distribution of $\mathbf{r}$ is geometric and independent of $\mathbf{q}_0$ so

Lemma 4

$$Pr\{\mathbf{r} = j\} = Pr\{\mathbf{r} = j \mid \mathbf{q}_0 > k\} = \left(1 - \frac{1}{m}\right)^{j-1} \frac{1}{m}.$$

Also,

Lemma 5

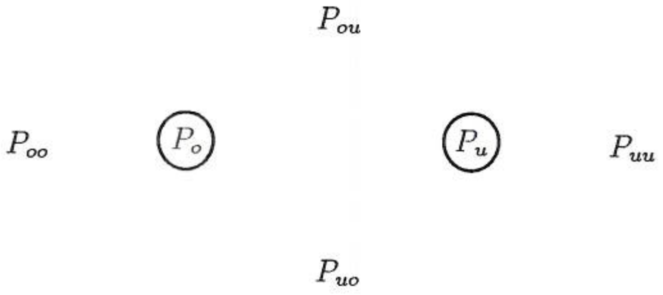
$$
\begin{aligned}
Pr\Big\{\mathbf{q}_j > k \mid \mathbf{q}_0 > k, \mathbf{r} = j\Big\} &= \left(\alpha + (1-\alpha)\left(1 - \frac{P_{ou}}{1-\alpha}\right)^j\right)^k \\
&= \sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i} (1-\alpha)^i \left(1 - \frac{P_{ou}}{1-\alpha}\right)^{ij}
\end{aligned}
$$

where P_{ou} is the transition probability that a cell in $\mathbf{ps}_0$ goes from occupied to unoccupied on the next operation.

Proof: Since the change of state between occupied and unoccupied for the cells in the probe sequence of E are independent between searches of E, we can write

$$
\Pr\Big\{\mathbf{q}_j > k \mid \mathbf{q}_0 > k, \mathbf{r} = j\Big\} = \prod_{s=1}^{k} \Pr\Big\{Q_j^{(s)} \mid Q_0^{(s)}\Big\}.
$$

The state of the s^{th} cell, between queries of E, alternates between occupied and unoccupied according to the following Markov process:



where P_{ou} is the probability that on the next hit cell s goes from occupied to unoccupied and P_{uo} is the probability of the opposite transition. The s^{th} cell can become unoccupied if the element in it is deleted or if it is queried and relocated. Then

$$
P_{ou} = \Pr\{R\}\frac{1-P_d}{m} + \frac{P_d}{m}. \tag{2}
$$

Also, since the table is α-full we can write

$$\alpha P_{ou} = (1-\alpha)P_{uo}.$$

Solving this finite Markov chain [1] we obtain the probability that the s^{th} cell in the probe sequence of E is again (or still) occupied after j hits.

$$\Pr\left\{Q_j^{(s)} \mid Q_0^{(s)}\right\} = \alpha + (1-\alpha)\left(1 - \frac{P_{ou}}{1-\alpha}\right)^j$$

The lemma follows. □

Lemma 6

$$Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\} = \sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i}(1-\alpha)^i \frac{\left(1 - \frac{(1-P_d)Pr\{R\}}{(1-\alpha)m} - \frac{P_d}{(1-\alpha)m}\right)^i}{1 + \frac{i(P_d+(1-P_d)Pr\{R\}}{1-\alpha} + O(\frac{1}{m})}.$$

Proof:

$$\begin{aligned}
\Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\} &= \sum_{j=1}^{\infty} \Pr\left\{\mathbf{q}_j > k \mid \mathbf{q}_0 > k, \mathbf{r} = j\right\}\left(1 - \frac{1}{m}\right)^{j-1} \frac{1}{m} \\
&= \sum_{j=1}^{\infty}\sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i}(1-\alpha)^i \left(1 - \frac{P_{ou}}{1-\alpha}\right)^{ij} \left(1 - \frac{1}{m}\right)^{j-1} \frac{1}{m} \\
&= \frac{1}{m}\sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i}(1-\alpha)^i \sum_{j=1}^{\infty} \left(1 - \frac{P_{ou}}{1-\alpha}\right)^{ij} \left(1 - \frac{1}{m}\right)^{j-1} \frac{1}{m} \\
&= \frac{1}{m}\sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i}(1-\alpha)^i \frac{\left(1 - \frac{P_{ou}}{1-\alpha}\right)^i}{1 - \left(1 - \frac{P_{ou}}{1-\alpha}\right)^i \left(1 - \frac{1}{m}\right)}.
\end{aligned}$$

Substituting for P_{ou} using (2) we obtain,

$$\begin{aligned}
&\Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\} = \\
&= \frac{1}{m}\sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i}(1-\alpha)^i \frac{\left(1 - \frac{(1-P_d)\Pr\{R\}}{(1-\alpha)m} - \frac{P_d}{(1-\alpha)m}\right)^i}{1 - \left(1 - \frac{(1-P_d)\Pr\{R\}}{(1-\alpha)m} - \frac{P_d}{(1-\alpha)m}\right)^i \left(1 - \frac{1}{m}\right)} \\
&= \frac{1}{m}\sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i}(1-\alpha)^i \frac{\left(1 - \frac{(1-P_d)\Pr\{R\}}{(1-\alpha)m} - \frac{P_d}{(1-\alpha)m}\right)^i}{1 - \left(1 - \frac{i(1-P_d)\Pr\{R\}}{(1-\alpha)m} - \frac{P_d i}{(1-\alpha)m} + O(\frac{1}{m^2})\right) \left(1 - \frac{1}{m}\right)}.
\end{aligned}$$

The lemma follows after rearranging terms. □

We are now ready to state the main result.

Theorem 2

$$\begin{aligned}\bar{S} &= \sum_{k=0}^{\infty}\sum_{i=1}^{\infty} Pr\{\mathbf{psl}_i > k \mid Y_i\} Pr\{Y_i\} \\ &= \sum_{k=0}^{\infty} \frac{\alpha^k Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\}}{1-(1-P_d)Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\}}\end{aligned}$$

where $Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\}$ is given by Lemma 6.

Proof: Follows from Lemmas 3 and 6 and the fact that $\Pr\{Y_i\} = (1-P_d)^{i-1}$. □

Although the result of Theorem 2 is messy, it can be used to obtain a bound on $\bar{S}$. If we make $\Pr\{R\} = 0$ then we only slow the rate at which element E can be moved back in its probe sequence, therefore increasing the value of $\bar{S}$. Thus,

$$\begin{aligned}\Pr\{\mathbf{psl}_1 > k \mid \mathbf{psl}_0 > k\} &\leq \sum_{i=0}^{k}\binom{k}{i}\alpha^{k-i}(1-\alpha)^i \frac{1}{1+\frac{P_d i}{1-\alpha}} \\ &\leq \sum_{i=0}^{k}\binom{k}{i}\alpha^{k-i}(1-\alpha)^i\left(1-\frac{P_d i}{1-\alpha}+\left(\frac{P_d i}{1-\alpha}\right)^2\right) \\ &= 1-P_d k+\frac{P_d^2 k}{1-\alpha}(1+(1-\alpha)(k-1)).\end{aligned}$$

Therefore,

$$\begin{aligned}\bar{S} &\leq \sum_{x=1}^{\infty}\frac{\alpha^{x-1}}{1-(1-P_d)\left(1-P_d(x-1)+\frac{P_d^2(x-1)}{1-\alpha}(1+(1-\alpha)(x-2))\right)} \\ &= \sum_{x=1}^{\infty}\frac{\alpha^{x-1}}{P_d x - P_d^2(x-1)\left(\frac{\alpha}{1-\alpha}+x\right)+P_d^3(x-1)\left(\frac{2\alpha-1}{1-\alpha}+x\right)}.\end{aligned}$$

Splitting the sum into two intervals, one from $x = 1$ to $x = \sqrt{1/P_d}$ and the other from $x = \sqrt{1/P_d}+1$ to ∞ we get

$$\bar{S} \leq \sum_{x=1}^{\sqrt{1/P_d}}\frac{\alpha^{x-1}}{P_d(1-\epsilon(P_d))x}+\alpha^{\sqrt{1/P_d}}\bar{S}$$

where $\epsilon(P_d) < \sqrt{P_d}$. Extending the upper summation limit to infinity we get

$$\bar{S} \leq \frac{-\ln(1-\alpha)}{\alpha(1-\epsilon(P_d))P_d} + \alpha^{\sqrt{1/P_d}}\bar{S}$$

so

$$\bar{S} \leq \frac{-\ln(1-\alpha)}{\alpha(1-\epsilon(P_d))P_d(1-\alpha^{\sqrt{P_d}})}. \tag{3}$$

If P_d is very small the upper bound of (3) is approximately $-\ln(1-\alpha)/(\alpha P_d)$. An upper bound on the number of visits per access (that is $\bar{S}/(\mathrm{E}[l]+1)$) is, therefore, $-\ln(1-\alpha)/\alpha$ in this case. If $P_d = 1$ then we have

$$\bar{S} = \sum_{x=1}^{\infty} \alpha^{x-1} = \frac{1}{1-\alpha}.$$

4 Summary

In this paper we have presented a very simple algorithm for reorganizing a hash table where deletions have occured during searches of table elements. We have presented an analysis that proves that if searches are much more frequent than insertions and deletions the average cost of successful searches is the same as if the deleted elements were never inserted into the table. We have derived a formula that may be used to bound the average number of cell visits in an element's lifetime in a system in equilibrium with any constant load factor and probability of deletion.

References

[1] **Feller, W.,** *An Introduction to Probability Theory and its Applications, Vol. I,* John Wiley & Sons, New York, 1968

[2] **Guibas, L.J.,** "The Analysis of Hashing Techniques that Exhibit K-ary Clustering", *Journal of the ACM,* Vol. 25, No. 4, pp.544-555, October 1978

[3] **Knuth, D.E.,** *The Art of Computer Programming, Vol. III: Sorting and Searching,* Addison-Wesley, Reading, Massachusetts, 1973

[4] **Lueker, G.S. and M. Molodowitch** "More Analysis of Double Hashing", *Proc. 20th ACM Symposium on Theory of Computing,* Chicago, Illinois, May, 1988

[5] **Peterson, W. W.,** "Addressing for Random-Access Storage", *IBM Journal of Research and Development* Vol. 1, No. 2, pp.130-146, April 1957

5 APPENDIX

In this section we justify assumptions 1 and 2 under the conditions that P_d, P_i, and therefore $\alpha = n/m$ are fixed. First we show

Lemma 7 *During the lifespan of element E, the fraction of occupied cells is between $\alpha - 1/n^{1/4}$ and $\alpha + 1/n^{1/4}$ with probability tending to 1.*

Proof: Let $q_i(t)$ be the probability that i cells are occupied at time t. We may use a well known birth and death model to develop recurrence relations for $q_i(t)$. We represent the equilibrium probabilities as q_i and find

$$q_i = \binom{n}{i} \alpha^i (1-\alpha)^{n-i}.$$

Let Q_β be the event that the number of occupied cells is not between $(1-\beta)\alpha n$ and $(1+\beta)\alpha n$ at any time assuming eqilibrium. By using the Chernoff bound for binomials we have

$$pr(Q_{1/\alpha n^{1/4}}) < 2e^{-\sqrt{n}/\alpha}.$$

Thus, one of every $2e^{\sqrt{n}/\alpha}$ hits either witnesses fewer than $(1 - 1/\alpha n^{1/4})\alpha n$ occupied cells or greater than $(1 + 1/\alpha n^{1/4})\alpha n$ occupied cells.

Consider an interval of n^2 consecutive hits with start point chosen randomly. The probability that $Q_{1/\alpha n^{1/4}}$ occurs in this interval is maximized if all hits which witness $Q_{1/\alpha n^{1/4}}$ are evenly spaced. Then the probability is at most $n^2 e^{-\sqrt{n}/\alpha}$ which tends to zero.

Finally, because the lifespan of E is geometrically distributed, the probability that the lifespan of E includes more than n^2 hits is $(1 - P_d/n)^{n^2}$ which tends to zero. Hence the probability that the fraction of occupied cells is between $\alpha - 1/n^{1/4}$ and $\alpha + 1/n^{1/4}$ during the lifespan of E tends to one. □

Next we show

Lemma 8 *During the lifespan of E, with probability tending to 1 there is no element in the hash table which has a probe sequence that intersects two or more cells of* $\mathbf{ps}_0$.

Proof: Suppose λ percent of the cells of $\mathbf{ps}_0$ are occupied, ps_0 has y cells, and β percent of all other cells are occupied. Then $\beta(n-y) = \alpha n - \lambda y$. The probability that the probe sequence, if any, of the next hit has x cells, none of which intersecting ps_0, is $(\beta(n-y)/n)^{x-1}(1-\alpha-(1-\lambda)y/n)$. The probability that the probe sequence of the next hit has x cells, the last of which intersects $\mathbf{ps}_0$, is $(\beta(n-y)/n)^{x-1}(1-\lambda)b/n$. The probability that the probe sequence of the next cell has x cells, one of which, but not the last one, intersecting $\mathbf{ps}_0$ is $(\beta(n-y)/n)^{x-2}(x\lambda b/n)(1-\alpha-(1-\lambda)b/n)$. Summing these three probabilities gives the probability that the probe sequence of the next hit, if any, has x cells and intersects $\mathbf{ps}_0$ in at most one cell. Using $\beta(n-y) = \alpha n - \lambda y$ and simplifying, this sum is

$$\left(\beta(1-\alpha) + \frac{1}{n}\left(x\lambda y(1-\alpha) - \beta y(1-\alpha)\right) + \frac{1}{n^2}\left(x\lambda^2 y^2 - x\lambda y^2\right)\right)\left(\frac{\beta(n-y)}{n}\right)^{x-2}.$$

Summing over all x gives the probability that the probe sequence of the next hit, if any, intersects $\mathbf{ps}_0$ in at most one cell. This is

$$\frac{1-\alpha}{1-\alpha+\frac{\lambda y}{n}} + \frac{\frac{\lambda y(1-\alpha)}{n} - \frac{\lambda y(1-\lambda)}{n^2}}{\left(1-\left(\alpha-\frac{\lambda y}{n}\right)\right)^2}.$$

This probability is minimum for $\lambda = 1$. In this case its value is

$$\begin{aligned} & 1 - \frac{y}{n(1-\alpha)} + O\left(\frac{y^2}{n^2}\right) + \frac{y}{n(1-\alpha)\left(1+\frac{y}{n(1-\alpha)}\right)^2} \\ = \; & 1 - \frac{y}{n(1-\alpha)} + O\left(\frac{y^2}{n^2}\right) + \frac{y}{n(1-\alpha)}\left(1 - \frac{y}{n(1-\alpha)} + O\left(\frac{y^2}{n^2}\right)\right)^2 \\ = \; & 1 - O\left(\frac{y^2}{n^2}\right). \end{aligned}$$

Hence, the probability that the probe sequence of the next hit, if any, intersects $\mathbf{ps}_0$ of size y more than once is $O(y^2/n^2)$. Therefore, the probability that the probe sequence of the next hit intersects $\mathbf{ps}_0$ in more than one cell is less than

$$\sum_{y=0}^{n} O\left(\frac{y^2}{n^2}\right)\alpha^y(1-\alpha) = O\left(\frac{1}{n^2}\right).$$

The probability that all probe sequences intersect $\mathbf{ps}_0$ in no more than one cell during the lifespan of E is less than

$$\sum_{j=0}^{\infty} O\left(\frac{1}{n^2}\right)\left(1-\frac{P_d}{n}\right)^j \frac{P_d}{n} = O\left(\frac{1}{n}\right)$$

which tends to zero as n tends to infinity. This proves the Lemma. □