LINEAR PROBING WITH CONSTANT INDEPENDENCE

ANNA PAGH*, RASMUS PAGH*, AND MILAN RUŽIĆ*

Abstract. Hashing with linear probing dates back to the 1950s, and is among the most studied algorithms. In recent years it has become one of the most important hash table organizations since it uses the cache of modern computers very well. Unfortunately, previous analyses rely either on complicated and space consuming hash functions, or on the unrealistic assumption of free access to a hash function with random and independent function values. Already Carter and Wegman, in their seminal paper on universal hashing, raised the question of extending their analysis to linear probing. However, we show in this paper that linear probing using a pairwise independent family may have expected *logarithmic* cost per operation. On the positive side, we show that 5-wise independence is enough to ensure constant expected time per operation. This resolves the question of finding a space and time efficient hash function that provably ensures good performance for linear probing.

Key words. Hashing, Linear Probing

1. Introduction. Hashing with linear probing is perhaps the simplest algorithm for storing and accessing a set of keys that obtains nontrivial performance. Given a hash function h, a key x is inserted in an array by searching for the first vacant array position in the sequence $h(x), h(x)+1, h(x)+2, \ldots$ (Here, addition is modulo r, the size of the array.) Retrieval of a key proceeds similarly, until either the key is found, or a vacant position is encountered, in which case the key is not present in the data structure. Deletions can be performed by moving keys back in the probe sequence in a greedy fashion (ensuring that no key x is moved to before $h(x)$), until no such move is possible (when a vacant array position is encountered).

Linear probing dates back to 1954, but was first analyzed by Knuth in a 1963 memorandum [5] now considered to be the birth of the area of analysis of algorithms [8]. Knuth's analysis, as well as most of the work that has since gone into understanding the properties of linear probing, is based on the assumption that h has uniformly distributed and independent function values. In 1977, Carter and Wegman's notion of universal hashing [2] initiated a new era in the design of hashing algorithms, where explicit and efficient ways of choosing hash functions replaced the unrealistic assumption of complete randomness. In their seminal paper, Carter and Wegman state it as an open problem to "Extend the analysis to [...] double hashing and open addressing."[1]

1.1. Previous results using limited randomness. The first analysis of linear probing relying only on limited randomness was given by Siegel and Schmidt in [9, 11]. Specifically, they show that $O(\log n)$-wise independence is sufficient to achieve essentially the same performance as in the fully random case. (We use n to denote the number of keys inserted into the hash table.) Another paper by Siegel [10] shows that evaluation of a hash function from a $O(\log n)$-wise independent family requires time $\Omega(\log n)$ unless the space used to describe the function is $n^{\Omega(1)}$. A family of functions is given that achieves space usage n^ϵ and constant time evaluation of functions, for any $\epsilon > 0$. However, this result is only of theoretical interest since the associated constants are very large (and growing exponentially with $1/\epsilon$).

*Computational Logic and Algorithms Group, IT University of Copenhagen, Denmark

[1]Nowadays the term "open addressing" refers to any hashing scheme where the data structure is an array containing only keys and empty locations. However, Knuth used th e term to refer to linear probing in [5], and since it is mentioned here together with the double hashing probe sequence, we believe that it refers to linear probing.

A potentially more practical method is the "split and share" technique described in [3]. It can be used to achieve characteristics similar to those of linear probing, still using space n^ϵ, for any given constant $\epsilon > 0$. The idea is to split the set of keys into many subsets of roughly the same size, and simulate full randomness on each part. Thus, the resulting solution would be a *collection* of linear probing hash tables.

A significant drawback of both methods above, besides a large number of instructions for function evaluation, is the use of random accesses to the hash function description. The strength of linear probing is that for many practical parameters, almost all lookups will incur only a single cache miss. Performing random accesses while computing the hash function value may destroy this advantage.

1.2. Our results. We show in this paper that linear probing using a pairwise independent family may have expected *logarithmic* cost per operation. Specifically, we resolve the open problem of Carter and Wegman by showing that linear probing insertion of n keys in a table of size $2n$ using a function of the form $x \mapsto ((ax + b) \bmod p) \bmod 2n$, where $p = 4n + 1$ is prime and we randomly choose $a \in [p]\backslash\{0\}$ and $b \in [p]$, requires $\Omega(n \log n)$ insertion steps in expectation for a worst case insertion sequence (chosen independently of a and b). Since the total insertion cost equals the total cost of looking up all keys, the expected average time to look up a key in the resulting hash table is $\Omega(\log n)$. The main observation behind the proof is that if a is the multiplicative inverse (modulo p) of a small integer m, then inserting a certain set that consists of two intervals has expected cost $\Omega(n^2/m)$.

On the positive side, we show that *5-wise independence* is enough to ensure constant expected time per operation, for load factor $\alpha \stackrel{\text{def}}{=} n/r$ bounded away from 1. Our proof is based on a new way of bounding the cost of linear probing operations, by counting intervals in which "many" probe sequences start. When beginning this work, our first observation was that a key x can be placed in location $h(x) + l \bmod r$ only if there is an interval $I \ni h(x)$ where $|I| \geq l$ and there are $|I|$ keys from S with hash value in I. A slightly stronger fact is shown in Lemma 4.1. Since the expected number of hash values in an interval I is $\alpha|I|$, long such intervals are "rare" if the hash function exhibits sufficiently high independence.

Our analysis gives a bound of $O(\frac{1}{(1-\alpha)^{13/6}})$ expected time per operation at load factor α. This implies a bound of $O(\frac{1}{(1-\alpha)^{7/6}})$ expected time on average for successful searches. These bounds are a factor $\Omega(\frac{1}{(1-\alpha)^{1/6}})$ higher than for linear probing with full independence. (The exponent can be made arbitrarily close to zero by increasing the independence of the hash function.)

In section 5 we describe an alternative to linear probing that preserves the basic property that all memory accesses of an operation are within a small interval, but improves the expected lookup time exponentially to $O(\log(\frac{1}{1-\alpha}))$.

1.3. Significance. Several recent experimental studies [1, 4, 7] have found linear probing to be the fastest hash table organization for moderate load factors (30-70%). While linear probing operations are known to require more instructions than those of other open addressing methods, the fact that they access an interval of array entries means that linear probing works very well with modern architectures for which sequential access is much faster than random access (assuming that the keys we are accessing are each significantly smaller than a cache line, or a disk block, etc.). However, the hash functions used to implement linear probing in practice are heuristics, and there is no known theoretical guarantee on their performance. Since linear probing is particularly sensitive to a bad choice of hash function, Heileman and Luo [4]

advice *against* linear probing for general-purpose use. Our results imply that simple and effici ent hash functions, whose description can be stored in CPU registers, can be used to give provably good expected performance.

2. Preliminaries.

2.1. Notation and definitions. Let $[x] \stackrel{\text{def}}{=} \{0, 1, \ldots, x-1\}$. Throughout this paper S denotes a subset of some universe U, and h will denote a function from U to $R \stackrel{\text{def}}{=} [r]$. We denote the elements of S by $\{x_1, x_2, \ldots, x_n\}$, and refer to the elements of S as *keys*. We let $n \stackrel{\text{def}}{=} |S|$, and $\alpha \stackrel{\text{def}}{=} n/r$.

A family $\mathcal{H}$ of functions from U to R is k-wise independent if for any k distinct elements $x_1, \ldots, x_k \in U$ and h chosen uniformly at random from $\mathcal{H}$, the random variables $h(x_1), \ldots, h(x_k)$ are independent[2]. We refer to the variable

$$\bar{\alpha}_{\mathcal{H}} \stackrel{\text{def}}{=} n \max_{x \in U,\, \rho \in R} \Pr_{h \in \mathcal{H}}\{h(x) = \rho\}$$

as the *maximum load* of $\mathcal{H}$. When the hash function family in question is understood from the context, we omit the subscript of $\bar{\alpha}$. If $\mathcal{H}$ distributes hash function values of all elements of U uniformly on R, we will have $\bar{\alpha} = \alpha$, and in general $\bar{\alpha} \geq \alpha$.

For $Q \subseteq R$ we introduce notation for the "translated set"

$$a + Q \stackrel{\text{def}}{=} \{(a + y) \bmod r \mid y \in Q\} \ .$$

An *interval* (modulo r) is a set of the form $a + [b]$, for integers a and b. When we write $[a - b, a)$ this interval represents the set $a - 1 - [b]$. We will later use sets of the form $h(x) + Q$, for a fixed x and with Q being an interval.

2.2. Hash function families. Carter and Wegman [13] observed that the family of degree $k-1$ polynomials in any finite field is k-wise independent. Specifically, for any prime p we may use the field defined by arithmetic modulo p to get a family of functions from $[p]$ to $[p]$ where a function can be evaluated in time $O(k)$ on a RAM, assuming that addition and multiplication modulo p can be performed in constant time. To obtain a smaller range $R = [r]$ we may map integers in $[p]$ down to R by a modulo r operation. This of course preserves independence, but the family is now only close to uniform. Specifically, the maximum load $\bar{\alpha}$ for this family is in the range $[\alpha,\ (1 + r/p)\alpha]$. By choosing p much larger than r we can make $\bar{\alpha}$ arbitrarily close to α.

A recently proposed k-wise independent family of Thorup and Zhang [12] has uniformly distributed function values in $[r]$, and thus $\bar{\alpha} = \alpha$. From a theoretical perspective (ignoring constant factors) it is inferior to Siegel's highly independent family [10], since the evaluation time depends on k and the space usage is the same (though the dependence of ϵ is better). We mention it here because it is the first construction that makes k-wise independence truly competitive with popular heuristics, for small $k > 3$, in terms of evaluation time. In practice, the space usage can be kept so small that it does not matter. The construction for 4-wise independence has been shown to be particularly efficient. Though this is not stated in [12], it is not hard to verify that the same construction in fact gives 5-wise independence, and thus our analysis will apply.

[2]We note that in some papers, the notion of k-wise independence is stronger in that it is required that function values are uniformly distributed in R. However, some interesting k-wise independent families have a slightly nonuniform distribution, and we will provide analysis for such families as well.

2.3. A probabilistic lemma. Here we state a lemma that is essential for our upper bound results, described in Section 4. It gives an upper bound on the probability that an interval around a particular hash function value contains the hash function values of "many" keys. The proof is similar to the proof of [**?**, Lemma 4.19].

LEMMA 2.1. *Let $S \subseteq U$ be a set of size n, and $\mathcal{H}$ a 5-wise independent family of functions from U to R with maximum load at most $\bar{\alpha} < 1$. If h is chosen uniformly at random from $\mathcal{H}$, then for any $Q \subset R$ of size q, and any fixed $x \in U \setminus S$,*

$$\Pr\left\{\left|\{y \in S \ : \ h(y) \in (h(x)+Q)\}\right| \geq \bar{\alpha}q + d\right\} < \frac{4\bar{\alpha}q^2}{d^4} \ .$$

Proof. Denote by A the event that $\left|\{y \in S \ : \ h(y) \in (h(x)+Q)\}\right| \geq \bar{\alpha}q + d$. We will show a stronger statement, namely that the same upper bound holds for the conditional probability $\Pr\{A \mid h(x) = \rho\}$, for any $\rho \in R$. Notice that the subfamily $\{h \in \mathcal{H} \mid h(x) = \rho\}$ is 4-wise independent on $U \setminus \{x\}$, and that the distribution of function values is identical to the distribution when h is chosen from $\mathcal{H}$. The statement of the lemma will then follow from

$$\Pr(A) = \sum_{\rho \in R} \Pr\{h(x) = \rho\}\Pr\{A \mid h(x) = \rho\} < r \cdot \frac{1}{r}\frac{4\bar{\alpha}q^2}{d^4} \ .$$

Let $p_i \stackrel{\text{def}}{=} \Pr\{h(x_i) \in (h(x)+Q)\}$, and consider the random variables

$$X_i \stackrel{\text{def}}{=} \begin{cases} 1-p_i, & \text{if } h(x_i) \in h(x)+Q \\ -p_i, & \text{otherwise} \end{cases} \ .$$

Let $X \stackrel{\text{def}}{=} \sum_i X_i$ and observe that

$$\left|\{y \in S \ : \ h(y) \in (h(x)+Q)\}\right| = X + \sum_i p_i \leq X + \bar{\alpha}q \ .$$

The last inequality above is by the definition of maximum load. So to prove the lemma it suffices to bound $\Pr\{X \geq d\}$. We will use the 4th moment inequality

$$\Pr\{X \geq d\} \leq \mathrm{E}(X^4)/d^4 \ .$$

Clearly, $\mathrm{E}(X_i) = 0$ for any i, and the variables $X_1, \ldots, X_n$ are 4-wise independent. Therefore we have $\mathrm{E}(X_{i_1}X_{i_2}X_{i_3}X_{i_4}) = 0$ unless $i_1 = i_2 = i_3 = i_4$ or (i_1, i_2, i_3, i_4) contains 2 numbers, both of them exactly twice. This means that

$$\begin{aligned} \mathrm{E}(X^4) &= \sum_{1 \leq i_1, i_2, i_3, i_4 \leq n} \mathrm{E}(X_{i1}X_{i2}X_{i3}X_{i4}) \\ &= \sum_{1 \leq i \leq n} \mathrm{E}(X_i^4) + \sum_{1 \leq i < j \leq n} \tbinom{4}{2}\mathrm{E}(X_i^2)\mathrm{E}(X_j^2). \end{aligned}$$

The first sum can be bounded as follows:

$$\begin{aligned} \sum_i \mathrm{E}(X_i^4) &= \sum_i (p_i(1-p_i)^4 + (1-p_i)p_i^4) \\ &= \sum_i p_i(1-p_i)((1-p_i)^3 + p_i^3) \\ &< \sum_i p_i \leq \bar{\alpha}q \ . \end{aligned}$$

The second sum is:

$$\sum_{1\le i<j\le n} 6(p_i(1-p_i))(p_j(1-p_j)) < 3\sum_{1\le i,j\le n} p_ip_j$$
$$= 3(\sum_i p_i)^2 \le 3(\bar{\alpha}q)^2 \ .$$

In conclusion we have

$$\Pr\{X \ge d\} \le \mathrm{E}(X^4)/d^4 < \frac{3(\bar{\alpha}q)^2 + \bar{\alpha}q}{d^4} < \frac{4\bar{\alpha}q^2}{d^4} \ ,$$

finishing the proof. □

3. Pairwise independence. In this section we show that pairwise independence is not sufficient to ensure good performance for linear probing: Logarithmic time per operation is needed for a worst-case set. This complements our upper bounds for 5-wise (and higher) independence. We will consider two pairwise independent families: The first one is a very commonly used hash function family. The latter family is similar to the first, except that we have ensured function values to be uniformly distributed in R. To lower bound the cost of linear probing we use the following lemma.

LEMMA 3.1. *Suppose a set S of n keys is inserted in a linear probing hash table of size $r > n$. Let $\{S_j\}_{j=1}^{\ell}$ be any partition of S such that for every set S_j the set $I_j \stackrel{def}{=} h(S_j)$ is an interval (modulo r), and $|I_j| \le r/2$. Then the total number of steps to perform the insertions is at least*

$$\sum_{1\le j_1<j_2\le \ell} |I_{j_1} \cap I_{j_2}|^2/2 \ .$$

Proof. We proceed by induction on ℓ. Since the number of insertion steps is independent of the order of insertions [6, p. 538], we may assume that the insertions corresponding to S_ℓ occur last and in left-to-right order of hash values. By the induction hypothesis, the total number of steps to do all preceding insertions is at least

$$\sum_{1\le j_1<j_2\le \ell-1} |I_{j_1} \cap I_{j_2}|^2/2 \ .$$

For $1 \le j_1, j_2 \le \ell$ let $S_{j_1 j_2}$ denote the set of keys from S_{j_1} that have probe sequences starting in I_{j_2}, i.e $S_{j_1 j_2} = \{x \in S_{j_1} \mid h(x) \in I_{j_2}\}$. For any $x \in S_{\ell j}$ the insertion of x will pass all the elements of $S_{j\ell}$ "after h(x)", i.e., whose hash value is in $h(x) + [r/2]$. This means that at least $|I_j \cap I_\ell|^2/2$ steps are used during the insertions of the keys from S_ℓ to pass locations occupied by keys of S_j. Summing over all $j < \ell$ and adding to the bound from the induction hypothesis yields the desired result. □

3.1. Linear congruential hash functions. We first consider the following family of functions, introduced by Carter and Wegman [2] as a first example of a universal family of hash functions:

$$\mathcal{H}(p,r) \stackrel{\text{def}}{=} \{x \mapsto ((ax+b) \bmod p) \bmod r \mid 0 < a < p,\ 0 \le b < p\}$$

where p is any prime number and $r \le p$ is any integer. Functions in $\mathcal{H}(p,r)$ map integers of $[p]$ to $[r]$.

THEOREM 3.2. *For $r = \lceil p/2 \rceil$ there exists a set $S \subseteq [p]$, $|S| \leq r/2$, such that the expected cost of inserting the keys of S in a linear probing hash table of size r using a hash function chosen uniformly at random from $\mathcal{H}(p, r)$ is $\Omega(r \log r)$.*

Proof. We give a randomized construction of S, and show that when choosing h at random from $\mathcal{H}(p, r)$ the expected total insertion cost for the keys of S is $\Omega(r \log r)$. This implies the existence of a fixed set S with at least the same expectation for random $h \in \mathcal{H}(p, r)$. Specifically, we partition $[p]$ into 8 intervals $U_1, \ldots, U_8$, such that $\bigcup_i U_i = [p]$ and $r/4 \geq |U_i| \geq r/4 - 1$ for $i = 1, \ldots, 8$, and let S be the union of two of the sets $U_1, \ldots, U_8$ chosen at random (without replacement). Note that $|S| \leq r/2$, as required.

Consider a particular function $h \in \mathcal{H}(p, r)$ and the associated values of a and b. Let $\hat{h}(x) \stackrel{\text{def}}{=} (ax + b) \bmod p$, and let m denote the unique integer in $[p]$ such that $am \bmod p = 1$ (i.e., $m = a^{-1}$ in GF(p)). Since $\hat{h}$ is a permutation on $[p]$, the sets $\hat{h}(U_i)$, $i = 1, \ldots, 8$, are disjoint. We note that for any x, $\hat{h}(x+m) = (\hat{h}(x)+1) \bmod p$. Thus, for any k, $\hat{h}(\{x, x+m, x+2m, \ldots, x+km\})$ is an interval (modulo p) of length $k+1$. This implies that for all i there exists a set $\hat{L}_i$ of m disjoint intervals such that $\hat{h}(U_i) = \bigcup_{I \in \hat{L}_i} I$. Similarly, for all i there exists a set L_i of at most $m+1$ intervals (not necessarily disjoint) such that we have the multiset equality $h(U_i) = \bigcup_{I \in L_i} I$. Since all intervals in $\bigcup_i \hat{L}_i$ are disjoint and their sizes differ by at most 1, an interval in $\bigcup_i L_i$ can intersect at most two other intervals in $\bigcup_i L_i$. We now consider two cases:

1. Suppose there is some i such that

$$\sum_{I_1, I_2 \in L_i, I_1 \neq I_2} |I_1 \cap I_2| \geq r/16 \ . \tag{3.1}$$

   With constant probability it holds that $U_i \subseteq S$. We apply Lemma 3.1 on the set U_i and a partition of U_i that corresponds to the interval collection L_i. The lemma gives us a lower bound of

$$\sum_{I_1, I_2 \in L_i, I_1 \neq I_2} |I_1 \cap I_2|^2/2 \tag{3.2}$$

   on the number of probes made during all insertions. This sum is minimized if all nonzero intersections have the same size. Suppose that there are $k = O(m)$ nonzero intersections. According to (3.1) the equal size of intersections would have to be $\Omega(r/k)$. Therefore the sum in (3.2) is $\Omega(r^2/k) = \Omega(r^2/m)$.
2. Now suppose that for all i,

$$\sum_{I_1, I_2 \in L_i, I_1 \neq I_2} |I_1 \cap I_2| < r/16 \ .$$

   Note that any value in $[r-1]$ is contained in exactly two intervals of $\bigcup_i L_i$. By the assumption, the number of values that occur in two intervals from the same collection L_i, for any i, is less than $8 \cdot r/16 = r/2$. Thus there exist i_1, i_2, $i_1 \neq i_2$, such that $|h(U_{i_1}) \cap h(U_{i_2})| = \Omega(r)$. With constant probability we have that $S = U_{i_1} \cup U_{i_2}$. We now apply Lemma 3.1. Consider just the terms in the sum of the form $|I_1 \cap I_2|^2/2$, where $I_1 \in L_{i_1}$ and $I_2 \in L_{i_2}$. As before, this sum is minimized if all $O(m)$ intersections have the same size, and we derive an $\Omega(r^2/m)$ lower bound on the number of insertion steps.

For a random $h \in \mathcal{H}(p,r)$, m is uniformly distributed in $\{1,\ldots,p\}$ (the mapping $a \mapsto a^{-1}$ is a permutation of $\{1,\ldots,p\}$). This means that the expected total insertion cost is:

$$\Omega\left(\frac{1}{p}\sum_{m=1}^{p} r^2/m\right) = \Omega\left(\frac{r^2}{p}\log p\right) = \Omega(r\log r) \ .$$

□

3.2. Family with uniform distribution. One might wonder if the lower bound shown in the previous section also holds if the hash function values are uniformly distributed in R. We slightly modify $\mathcal{H}(p,r)$ to remain pairwise independent and also have uniformly distributed function values. Let $\hat{p} \stackrel{\text{def}}{=} \lceil p/r \rceil r$, and define:

$$g(y,\hat{y}) \stackrel{\text{def}}{=} \begin{cases} \hat{y} & \text{if } \hat{y} \geq p \\ y & \text{otherwise} \end{cases} \ .$$

For a vector v let v_i denote the $i+1$st component (indexes starting with zero). We define:

$$\mathcal{H}^*(p,r) \stackrel{\text{def}}{=} \{x \mapsto g((ax+b) \bmod p, v_x) \bmod r \mid 0 \leq a < p,\ 0 \leq b < p,\ v \in [\hat{p}]^p\}$$

LEMMA 3.3 (Pairwise independence). *For any pair of distinct values $x_1, x_2 \in [p]$, and any $y_1, y_2 \in [r]$, if h is chosen uniformly at random from $\mathcal{H}^*(p,r)$, then*

$$\Pr\{h(x_1) = y_1 \wedge h(x_2) = y_2\} = 1/r^2 \ .$$

Proof. We will show something stronger than claimed, namely that the family

$$\mathcal{H}^{**} = \{x \mapsto g((ax+b) \bmod p, v_x) \mid 0 \leq a < p,\ 0 \leq b < p,\ v \in [\hat{p}]^p\}$$

is pairwise independent and has function values uniformly distributed in $[\hat{p}]$. Since r divides $\hat{p}$ this will imply the lemma. Pick any pair of distinct values $x_1, x_2 \in [p]$, and consider a random function $h \in \mathcal{H}^{**}$. Clearly, v_{x_1} and v_{x_2} are uniform in $[\hat{p}]$ and independent. We note as in [2] that for any $y_1', y_2' \in [p]$ there is exactly one choice of a and b that makes $(ax_1+b) \bmod p = y_1'$ and $(ax_2+b) \bmod p = y_2'$. This is because the matrix $\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \end{pmatrix}$ is invertible. As a consequence, $(ax_1+b) \bmod p$ and $(ax_2+b) \bmod p$ are uniform in $[p]$ and independent. We can think of the definition of $h(x)$ as follows: The value is v_x unless $v_x \in [p]$, in which case we substitute v_x for another random value in $[p]$, namely $(ax+b) \bmod p$. It follows that hash function values are uniformly distributed, and pairwise independent. □

COROLLARY 3.4. *Theorem 3.2 holds also if we replace $\mathcal{H}(p,r)$ by $\mathcal{H}^*(p,r)$. In particular, pairwise independence with uniformly distributed function values is not a sufficient condition for linear probing to have expected constant cost per operation.*

Proof. Consider the parameters a, b, and v of a random function in $\mathcal{H}^*(p,r)$. Since $r = \lceil p/2 \rceil$ we have $\hat{p} = p+1$, and $(p/\hat{p})^p > 1/4$. Therefore, with constant probability it holds that $a \neq 0$ and $v \in [p]^p$. Restricted to functions satisfying this, the family $\mathcal{H}^*(p,r)$ is identical to $\mathcal{H}(p,r)$. Thus, the lower bound carries over (with a smaller constant). By Lemma 3.3, $\mathcal{H}^*$ is pairwise independent with uniformly distributed function values. □

We remark that the lower bound is tight. A corresponding $O(n \log n)$ upper bound can be shown by applying the framework of section 4, but using Chebychev's inequality rather than Lemma 2.1 as the basic tool for bounding probabilities.

4. 5-wise independence. We want to bound the expected number of probes into the table made during any single operation (insertion, deletion, or lookup of a key x) when the hash table contains the set S of keys. It is well known that for linear probing, the set P of occupied table positions depends only on the set S and the hash function, independent of the sequence of insertions and deletions performed. An operation on key x makes no more than

$$1 + \max\{l \mid h(x) + [l] \subseteq P\}$$

probes into the table, because the iteration stops when the next unoccupied position is found (or sooner in case of a successful search). We first show a lemma which intuitively says that if the operation on the key x goes on for at least l steps, then there are either "many" keys hashing to the interval $h(x) + [l]$, or there are "many" keys that hash to some interval having $h(x)$ as its right endpoint.

LEMMA 4.1. *For any $l > 0$ and $\bar{\alpha} \in (0, 1)$, if $h(x) + [l] \subseteq P$ then at least one of the following holds:*

1. $\left|\{y \in S \setminus \{x\} \ : \ h(y) \in (h(x) + [l])\}\right| \geq \frac{1+\bar{\alpha}}{2} l - 1$, *or*
2. $(\exists \ell)\ \left|\{y \in S \ : \ h(y) \in [h(x) - \ell, h(x))\}\right| \geq \ell + \frac{1-\bar{\alpha}}{2} l$.

Proof. Suppose that $\left|\{y \in S \setminus \{x\} \ : \ h(y) \in (h(x) + [l])\}\right| < \frac{1+\bar{\alpha}}{2} l - 1$. Then in either case, $x \in S$ or $x \notin S$, it holds that $\left|\{y \in S \ : \ h(y) \in (h(x) + [l])\}\right| < \frac{1+\bar{\alpha}}{2} l$. Let

$$l' \stackrel{\text{def}}{=} \max\{\ell \ : \ [h(x) - \ell, h(x)] \subseteq P\} \ .$$

Now, fix any way of placing the keys in the hash table, e.g., suppose that keys are inserted in sorted order. Consider the set $S^* \subseteq S$ of keys stored in the interval $I = [h(x) - l', h(x) + l - 1]$. By the choice of l' there must be an empty position to the left of I, so $h(S^*) \subseteq I$. This means:

$$\begin{aligned}
\left|\{y \in S \ : \ h(y) \in [h(x) - l', h(x))\}\right| &\geq \left|\{y \in S^* \ : \ h(y) \in [h(x) - l', h(x))\}\right| \\
&\geq |S^*| - \left|\{y \in S^* \ : \ h(y) \in (h(x) + [l])\}\right| \\
&> |I| - \tfrac{1+\bar{\alpha}}{2} l \\
&= l' + \tfrac{1-\bar{\alpha}}{2} l \ .
\end{aligned}$$

□

4.1. A simple bound. We start out with a bound that is simpler to derive than our final bound in section 4.2. It is possible to skip this section, but we believe that reading it makes it easier to understand the more complicated argument in section 4.2.

The next lemma upper bounds the probability that there exists some interval of form $[h(x) - \ell, h(x))$ having $\ell + d$ keys hashing into it, with d being a parameter. The derived bound will later be used to cover the case 2 from Lemma 4.1.

LEMMA 4.2. *Let $S \subseteq U$ be a set of size n, and $\mathcal{H}$ a 5-wise independent family of functions from U to R with a maximum load of $\bar{\alpha} < 1$. If h is chosen uniformly at random from $\mathcal{H}$, then for any $x \in U$ and $\lambda > 0$,*

$$\Pr\left\{\max_{\ell} \left(\left|\{y \in S \ : \ h(y) \in [h(x) - \ell, h(x))\}\right| - \ell\right) \geq \frac{\lambda + 1}{(1 - \bar{\alpha})^{3/2}}\right\} < \frac{8\bar{\alpha}}{\lambda^2} \ .$$

Proof. We will use the symbol Δ to denote $\lceil \frac{\lambda}{2}(1 - \bar{\alpha})^{-3/2} \rceil$. Let A_i be the event that

$$\left|\{y \in S \ : \ h(y) \in [h(x) - i\Delta, h(x))\}\right| - i\Delta \geq \Delta \ .$$

We claim that it is sufficient to show $\Pr\left(\bigcup_{i>0} A_i\right) < \frac{8\bar{\alpha}}{\lambda^2}$. To see this, suppose that $\left|\{y \in S \,:\, h(y) \in [h(x)-\ell,\, h(x))\}\right| - \ell \geq \frac{\lambda+1}{(1-\bar{\alpha})^{3/2}}$, for some ℓ. Let $i' = \lceil \frac{\ell}{\Delta} \rceil$. Then

$$\begin{aligned}\left|\{y \in S \,:\, h(y) \in [h(x)-i'\Delta,\, h(x))\}\right| &\geq \ell + \frac{\lambda+1}{(1-\bar{\alpha})^{3/2}} \\ &\geq i'\Delta - (\Delta-1) + \lambda(1-\bar{\alpha})^{-3/2} + 1 \\ &\geq i'\Delta + \frac{\lambda}{2}(1-\bar{\alpha})^{-3/2} + 1 \geq i'\Delta + \Delta \ .\end{aligned}$$

In this lemma we use a simple upper bound $\Pr(\bigcup_{i>0} A_i) \leq \sum_{i>0} \Pr(A_i)$. We use Lemma 2.1 to estimate each value $\Pr(A_i)$. Note that intersections of any interval $[h(x)-\ell,\, h(x))$ with the sets $h(S \setminus \{x\})$ and $h(S)$ are the same.

$$\sum_{i>0} \Pr(A_i) \leq \sum_{i>0} \frac{4\bar{\alpha}(i\Delta)^2}{((1-\bar{\alpha})i\Delta + \Delta)^4} \leq \frac{4\bar{\alpha}}{\Delta^2} \sum_t \frac{(\frac{t}{1-\bar{\alpha}})^2}{(t+1)^4}$$

We used the substitution $t = (1-\bar{\alpha})i$. The last sum is over $t \in \{1-\bar{\alpha},\ 2(1-\bar{\alpha}), \ldots\}$. The function $\frac{t^2}{(1+t)^4}$ is first increasing and then decreasing on $[0,\infty)$. Thus the sum can be bounded by the integral $\frac{1}{1-\bar{\alpha}} \int_{1-\bar{\alpha}}^{\infty} \frac{t^2}{(1+t)^4} dt$ plus the value of the biggest term in the sum.

$$\begin{aligned}\sum_{i>0} \Pr(A_i) &< \frac{4\bar{\alpha}}{\Delta^2} \frac{1}{(1-\bar{\alpha})^2} \left(\max_{t>0} \frac{t^2}{(1+t)^4} + \frac{1}{1-\bar{\alpha}} \int_{1-\bar{\alpha}}^{\infty} \frac{t^2}{(1+t)^4} dt \right) \\ &< \frac{4\bar{\alpha}}{\Delta^2} \frac{1}{(1-\bar{\alpha})^3} \left(\frac{1}{10} + \int_0^{\infty} \frac{t^2}{(1+t)^4} dt \right) \\ &< \frac{2\bar{\alpha}}{\Delta^2} (1-\bar{\alpha})^{-3} \leq \frac{8\bar{\alpha}}{\lambda^2} \ .\end{aligned}$$

□

THEOREM 4.3. *Consider any sequence of operations (insertions, deletions, and lookups) in a linear probing hash table where the hash function h used has been chosen uniformly at random from a 5-wise independent family of functions $\mathcal{H}$. Let n and $\bar{\alpha} < 1$ denote, respectively, the maximum number of keys in the table during a particular operation and the corresponding maximum load. Then the expected number of probes made during that operation is $O((1-\bar{\alpha})^{-5/2})$.*

Proof. We refer to x, S, and P as defined previously in this section. As argued above, the expected probe count is bounded by

$$1 + \sum_{l>0} \Pr\{h(x) + [l] \subseteq P\} \ .$$

Let $l_0 = \frac{10}{(1-\bar{\alpha})^{5/2}}$. For $l \leq l_0$ we use the trivial upper bound $\Pr\{h(x) + [l] \subseteq P\} \leq 1$. In the following we consider the case $l > l_0$.

Let A_l be the event that $\left|\{y \in S \setminus \{x\} \,:\, h(y) \in (h(x) + [l])\}\right| \geq \frac{1+\bar{\alpha}}{2} l - 1$, and let B_l be the event that $(\exists \ell)\ \left|\{y \in S \,:\, h(y) \in [h(x)-\ell,\, h(x))\}\right| \geq \ell + \frac{1-\bar{\alpha}}{2} l$. Lemma 4.1 implies that

$$\sum_{l>l_0} \Pr\{h(x) + [l] \subseteq P\} \leq \sum_{l>l_0} \big(\Pr(A_l) + \Pr(B_l)\big) \ .$$

Estimates of $\Pr(A_l)$ and $\Pr(B_l)$ are obtained from Lemma 2.1 and Lemma 4.2 respectively:

$$\begin{aligned}\sum_{l>l_0}\big(\Pr(A_l)+\Pr(B_l)\big) &< \sum_{l>l_0}\left(\frac{4\bar{\alpha}l^2}{(\frac{1-\bar{\alpha}}{2}l-1)^4}+\frac{8\bar{\alpha}}{(\frac{(1-\bar{\alpha})^{5/2}}{2}l-1)^2}\right)\\ &= O\Big(\bar{\alpha}\sum_{l>l_0}\Big((1-\bar{\alpha})^{-4}l^{-2}+(1-\bar{\alpha})^{-5}l^{-2}\Big)\Big)\\ &= O\Big((1-\bar{\alpha})^{-5}/l_0\Big) = O((1-\bar{\alpha})^{-5/2}) \ .\end{aligned}$$

□

4.2. Improving the bound. By inspecting the proof of Theorem 4.3, one notices that an improvement to the result of Lemma 4.2 directly leads to an improvement of the main upper bound. The following lemma gives a bound with better dependence on α, which is significant for high load factors. The stated constant factor is far from being tight. Showing a considerably better constant factor would require a more tedious proof with inelegant calculations.

LEMMA 4.4. *Let $S \subseteq U$ be a set of size n, and $\mathcal{H}$ a 5-wise independent family of functions from U to R with a maximum load of $\bar{\alpha}$. If h is chosen uniformly at random from $\mathcal{H}$, then for any $x \in U$,*

$$\Pr\left\{\max_{\ell}\big(\big|\{y\in S\ :\ h(y)\in[h(x)-\ell,\,h(x))\}\big|-\ell\big)\geq\frac{\lambda+2}{(1-\bar{\alpha})^{7/6}}\right\}<\frac{500\bar{\alpha}}{\lambda^2} \ .$$

Proof. We will use the symbol Δ to denote $\lceil\frac{\lambda}{3}(1-\bar{\alpha})^{-7/6}\rceil$. Let A'_i be the event that

$$\big|\{y\in S\ :\ h(y)\in[h(x)-i\Delta,\,h(x))\}\big|-i\Delta\geq 2\Delta \ .$$

It is sufficient to find a good upper bound on $\Pr(\bigcup_{i>0}A'_i)$. To see this, suppose that $\big|\{y\in S\ :\ h(y)\in[h(x)-\ell,\,h(x))\}\big|-\ell\geq\frac{\lambda+2}{(1-\bar{\alpha})^{7/6}}$, for some ℓ. Let $i'=\lceil\frac{\ell}{\Delta}\rceil$. Then

$$\begin{aligned}\big|\{y\in S\ :\ h(y)\in[h(x)-i'\Delta,\,h(x))\}\big| &\geq \ell+\frac{\lambda+2}{(1-\bar{\alpha})^{7/6}} \geq i'\Delta-(\Delta-1)+\lambda(1-\bar{\alpha})^{-7/6}+2\\ &\geq i'\Delta+2\frac{\lambda}{3}(1-\bar{\alpha})^{-7/6}+2\geq i'\Delta+2\Delta \ .\end{aligned}$$

We define the events A_i by $A_i = A'_i \setminus \bigcup_{j>i}A'_j$. It holds that $A_i\cap A_j=\emptyset$, $i\neq j$, and $\bigcup_{i>0}A_i=\bigcup_{i>0}A'_i$. Therefore, $\Pr(\bigcup_{i>0}A'_i)=\sum_{i>0}\Pr(A_i)$.

For the purpose of determining certain constraints that values $\Pr(A_i)$ must satisfy, we define the events B_i and C_{ij} by

$$B_i=\left\{h\in\mathcal{H}\ :\ \big|\{y\in S\ :\ h(y)\in[h(x)-i\Delta,\,h(x))\}\big|\geq\frac{1+\bar{\alpha}}{2}i\Delta+\Delta\right\} \ ,$$

and

$$C_{ij}=\left\{h\in\mathcal{H}\ :\ \big|\{y\in S\ :\ h(y)\in[h(x)-i\Delta,\,h(x)-j\Delta\}\big|\geq\frac{1-\bar{\alpha}}{2}j\Delta+(i-j+1)\Delta\right\} \ ,$$

for $i>j>0$. Intuitively, B_i is the event that A'_i *nearly* holds, with no more than $(1+\frac{1-\bar{\alpha}}{2}i)\Delta$ elements missing in the interval. C_{ij} is the event that the interval

$[h(x) - i\Delta,\, h(x) - j\Delta\big)$ contains the hash values of at least $\frac{1-\bar{\alpha}}{2} j\Delta$ more elements than the size of the interval. For $k < i$, it holds that

$$A_i \subseteq A'_i \subseteq B_{i-k} \cup C_{i,\,i-k} \ . \tag{4.1}$$

Hence, for a fixed j,

$$\bigcup_{i>j} (A_i \setminus C_{ij}) \subset B_j \ ,$$

and as a result $\sum_{i>j} \Pr(A_i \setminus C_{ij}) \leq \Pr(B_j)$. Summing over $j > 0$ and re-expressing the sums we get:

$$\sum_{i>0} \sum_{j>0} \Pr(A_i \setminus C_{ij}) \leq \sum_{i>0} \Pr(B_i) \ . \tag{4.2}$$

We will first estimate $\sum_i \Pr(B_i)$ using Lemma 2.1 (note that intersections of any interval $[h(x) - \ell,\, h(x))$ with the sets $h(S \setminus \{x\})$ and $h(S)$ are the same):

$$\begin{aligned}
\sum_{i>0} \Pr(B_i) &\leq \sum_{i>0} \frac{4\bar{\alpha}(i\Delta)^2}{(\frac{1-\bar{\alpha}}{2} i\Delta + \Delta)^4} \\
&< \frac{4\bar{\alpha}}{\Delta^2} \frac{8}{(1-\bar{\alpha})^3} \left(\max_{t>0} \frac{t^2}{(1+t)^4} + \int_{1-\bar{\alpha}}^{\infty} \frac{t^2}{(1+t)^4} dt \right) \\
&< \frac{4\bar{\alpha}}{\Delta^2} \frac{8}{(1-\bar{\alpha})^3} \left(\frac{1}{10} + \int_{0}^{\infty} \frac{t^2}{(1+t)^4} dt \right) \\
&< \frac{16\bar{\alpha}}{\Delta^2} (1-\bar{\alpha})^{-3} \ .
\end{aligned}$$

Again using Lemma 2.1 to estimate $\Pr(C_{ij})$, for $i > j > 0$, we get:

$$\Pr(C_{ij}) \leq \frac{4\bar{\alpha}(i-j)^2\Delta^2}{(\frac{1-\bar{\alpha}}{2}(2i-j)\Delta + \Delta)^4} < \frac{4\bar{\alpha}}{\Delta^2} \frac{(i-j)^2}{(\frac{1-\bar{\alpha}}{2} i + 1)^4} \ .$$

For the probability of the event $A_i \setminus C_{ij}$ we use a trivial lower bound of $\Pr(A_i) - \Pr(C_{ij})$. Hence, for any $i > 0$,

$$\sum_{k=1}^{i-1} \Pr(A_i \setminus C_{i,\,i-k}) > \sum_{k=1}^{i-1} \max \left\{ 0,\ \Pr(A_i) - \frac{4\bar{\alpha}}{\Delta^2} \frac{k^2}{(\frac{1-\bar{\alpha}}{2} i + 1)^4} \right\} \ .$$

Let $p_i = \Pr(A_i)$, $\gamma_i = \frac{4\bar{\alpha}}{\Delta^2}(\frac{1-\bar{\alpha}}{2} i + 1)^{-4}$, and $k_i = \lfloor \sqrt{p_i/\gamma_i} \rfloor$. Lemma 2.1 gives $p_i < \frac{4\bar{\alpha}(i\Delta)^2}{((1-\bar{\alpha})i\Delta + 2\Delta)^4}$, and so $k_i \leq \lfloor i/4 \rfloor \leq i - 1$. We further have that

$$k_i p_i - \gamma_i \sum_{k=1}^{k_i} k^2 > p_i(\sqrt{p_i/\gamma_i} - 1) - \gamma_i \left(\frac{(p_i/\gamma_i)^{3/2}}{2} + 1 \right) = \frac{p_i^{3/2}}{2\sqrt{\gamma_i}} - p_i - \gamma_i \ .$$

An upper bound on $\sum_{i>0} p_i$ can be obtained through solving an optimization problem over real variables x_1, x_2, …, $x_{\lfloor r/\Delta \rfloor}$. The problem is to maximize $\sum_{i>0} x_i$ subject to the constraint that

$$\sum_{i>0} \frac{x_i^{3/2}}{2\sqrt{\gamma_i}} - x_i \leq \frac{16\bar{\alpha}}{\Delta^2} (1-\bar{\alpha})^{-3} + \sum_{i>0} \gamma_i \ .$$

The vector $(p_1, p_2, \ldots)$ is a feasible solution to the problem and thus $\sum_{i>0} p_i$ is not larger than the optimal value. By employing the method of Lagrange multipliers we find that the optimal solution is $x_i = \gamma_i y^2$, where y is a value that satisfies:

$$y^3 - y^2 = \frac{4}{(1-\bar{\alpha})^3} \frac{1}{\sum_{i>0} (\frac{1-\bar{\alpha}}{2} i + 1)^{-4}} + 1 \ .$$

We have that $\frac{2}{3(1-\bar{\alpha})} < \sum_{i>0} (\frac{1-\bar{\alpha}}{2} i + 1)^{-4} < 1 + \frac{2}{3(1-\bar{\alpha})}$. Suppose that $y > 2.5$, so we may write $\frac{1}{2} y^3 < y^3 - y^2 - 1$. It follows that $y < (\frac{16}{(1-\bar{\alpha})^2})^{1/3}$. Since $\sqrt[3]{16} > 2.5$ the calculated upper bound on y is true in general. Finally,

$$\sum_{i>0} p_i < \sum_{i>0} \gamma_i y^2 < \left(\frac{16}{(1-\bar{\alpha})^2} \right)^{2/3} \frac{72\bar{\alpha}}{\lambda^2} (1-\bar{\alpha})^{14/6-1} < \frac{500\bar{\alpha}}{\lambda^2} \ .$$

□

By changing l_0 to $\frac{50}{(1-\bar{\alpha})^{13/6}}$ in the proof of Theorem 4.3, and utilizing the previous lemma, the following result is proved.

THEOREM 4.5. *Consider any sequence of operations (insertions, deletions, and lookups) in a linear probing hash table where the hash function h used has been chosen uniformly at random from a 5-wise independent family of functions $\mathcal{H}$. Let n and $\bar{\alpha} < 1$ denote, respectively, the maximum number of keys in the table during a particular operation and the corresponding maximum load. Then the expected number of probes made during that operation is $O((1-\bar{\alpha})^{-13/6})$.*

5. Improving the lookup cost. In this section we briefly describe an alternative to standard linear probing that improves the cost of lookups exponentially, without significantly changing the characteristics of linear probing. Update operations exhibit the same memory access pattern as before. Lookups perform jumps in memory, but still access only memory locations within as small interval – at most twice the length of the interval that would be inspected by the standard lookup procedure.

The idea is to order the keys of each maximal interval of occupied positions according to values of the hash function, in order to be able to do a *doubling search* during lookups. In other words, if there is a choice of more than one key to be placed at slot i then we choose the key having the hash value farthest from i (in the metric $(i - h(x)) \bmod r$). If there is more than one key with the most distant hash value, the smallest such key is stored at slot i. This invariant can be maintained during insertions and deletions at no asymptotic cost in running time, and the analysis of all operations stays the same.

Now consider a search for a key x, and assume for simplicity of exposition that r is a power of 2, and that the table is not full. Instead of searching for x sequentially we do a doubling search in the interval $h(x) + [r-1]$ (which must contain x). For this to work we must argue that inspecting a location $(h(x)+i) \bmod r$ allows us to determine whether we should continue the search for x before or after. If $(h(x)+i) \bmod r$ is an empty location, it is clear that we must search before. By the invariant the same is true if location $(h(x)+i) \bmod r$ contains a key x' such that $h(x') \in h(x) + 1 + [i]$ or $h(x') = h(x) \wedge x' > x$. Otherwise, x cannot be in $h(x) + [i]$. The doubling search finds an interval $h(x) + [i, 2i]$ that contains x in case $x \in S$. Binary search is then applied on this interval. This means that any search that would take time l using standard linear probing now takes time $O(\log l)$. Specifically, the expected search time goes down to $O(\log \frac{1}{1-\bar{\alpha}})$.

6. Open problems. An immediate question is whether the dependence on α for linear probing with constant independence matches the dependence on α in the case of full independence, up to a constant factor. It is unclear whether 4 or even 3-wise independence can guarantee good expected performance for linear probing. If one could prove a sufficiently strong tail bound in a style of the bound from Lemma 2.1 it could be plugged into the framework of Section 4; the bound would have to be polynomially stronger than the bound that results from Chebyshev's inequality.

In general, the problem of finding practical, and provably good hash functions for a range of other important hashing methods remains unsolved. For example, cuckoo hashing [7] and its variants presently have no such functions. Also, if we consider the problem of hashing a set into $n/\log n$ buckets such that the number of keys in each bucket is $O(\log n)$ w.h.p., there is no known explicit class achieving this with function descriptions of $O(\log |U|)$ bits. Possibly, such families could be designed using efficient circuits, rather than a standard RAM instruction set.

Acknowledgments. We thank Martin Dietzfelbinger and an anonymous reviewer for numerous suggestions improving the presentation. In particular, we thank Martin Dietzfelbinger for showing us a simplified proof of Lemma 2.1.

REFERENCES

[1] John R. Black, Charles U. Martel, and Hongbin Qi, *Graph and hashing algorithms for modern architectures: Design and performance*, in Algorithm Engineering, Kurt Mehlhorn, ed., Max-Planck-Institut für Informatik, 1998, pp. 37–48.

[2] J. Lawrence Carter and Mark N. Wegman, *Universal classes of hash functions*, J. Comput. System Sci., 18 (1979), pp. 143–154.

[3] Martin Dietzfelbinger and Christoph Weidling, *Balanced allocation and dictionaries with tightly packed constant size bins*, in ICALP, vol. 3580 of Lecture Notes in Computer Science, Springer, 2005, pp. 166–178.

[4] Gregory L. Heileman and Wenbin Luo., *How caching affects hashing*, in Proceedings of the 7th Workshop on Algorithm Engineering and Experiments (ALENEX05), SIAM, 2005, pp. 141–154.

[5] Donald E. Knuth, *Notes on "open" addressing*, July 22 1963. Unpublished memorandum. Available at `http://citeseer.ist.psu.edu/knuth63notes.html`.

[6] ———, *Sorting and Searching*, vol. 3 of The Art of Computer Programming, Addison-Wesley Publishing Co., Reading, Mass., second ed., 1998.

[7] Rasmus Pagh and Flemming Friche Rodler, *Cuckoo hashing*, Journal of Algorithms, 51 (2004), pp. 122–144.

[8] H. Prodinger and W. Szpankowski (eds.), *Special issue on average case analysis of algorithms*, Algorithmica, 22 (1998). Preface.

[9] Jeanette P. Schmidt and Alan Siegel, *The analysis of closed hashing under limited randomness (extended abstract)*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC '90), ACM Press, 1990, pp. 224–234.

[10] Alan Siegel, *On universal classes of extremely random constant-time hash functions*, SIAM J. Comput., 33 (2004), pp. 505–543.

[11] A. Siegel and J. Schmidt, *Closed hashing is computable and optimally randomizable with universal hash functions*, Technical Report TR1995-687, New York University, Apr., 1995.

[12] Mikkel Thorup and Yin Zhang, *Tabulation based 4-universal hashing with applications to second moment estimation.*, in Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '04), 2004, pp. 615–624.

[13] Mark N. Wegman and J. Lawrence Carter, *New hash functions and their use in authentication and set equality*, J. Comput. System Sci., 22 (1981), pp. 265–279.