COMBINATORICA
Akadémiai Kiadó – Springer-Verlag

COMBINATORICA 13 (1) (1993) 83–96

MORE ANALYSIS OF DOUBLE HASHING†

GEORGE S. LUEKER* and MARIKO MOLODOWITCH**

Received October 30, 1989
Revised November 6, 1991

In [8], a deep and elegant analysis shows that double hashing is asymptotically equivalent to the ideal uniform hashing up to a load factor of about 0.319. In this paper we show how a randomization technique can be used to develop a surprisingly simple proof of the result that this equivalence holds for load factors arbitrarily close to 1.

1. Introduction

In [8], a deep and elegant analysis shows that double hashing is equivalent to the ideal uniform hashing up to a load factor of about 0.319. In this paper we give an analysis which extends this to load factors arbitrarily close to 1. We understand from [6, 7] that Ajtai, Guibas, Komlós, and Szemerédi obtained this result in the first part of 1986; the analysis in this paper is of interest nonetheless because we demonstrate how a randomization technque can be used to obtain a remarkably simple proof.

A *hash table* will consist of an array of m *slots*, indexed from 0 to $m-1$, each of which can contain a key. In a scheme called open addressing, a function called a *hash function* can be applied to a key to yield a permutation of the slot indices $[0 \ldots m-1]$, called the *probe sequence*. To insert a key K, we place K into the first empty table slot in its probe sequence. To search for a key K, we look in slots indexed by successive elements of the probe sequence for K until we find it or find an empty slot; in the latter case we know that K must not be in the table. If we have inserted n keys into the table, we say it is filled to a *load factor* of n/m.

† An earlier version of the paper was presented at the *20th Annual ACM Symposium on Theory of Computing*, Chicago, IL, May 1988.

* Supported by National Science Foundation Grants DCR 85-09667 and CCR 89-12063 at the University of California at Irvine

** Partially supported by National Science Foundation Grant DCR 85-09667 at the University of California at Irvine

AMS subject Classification Code (1991): 68 Q 25, 68 P 10, 11 B 25

Various methods can be used to determine the probe sequence for a key K. In a theoretical ideal called *uniform hashing*, we assume that the hash function and distribution of keys are such that all permutations of the slots are equally likely probe sequences. Let $C'_\alpha(m)$ be the average number of probes during an unsuccessful search of a table of size m with load factor α. As in [8], this will include the probe that found an empty table slot. (Note that this is the same as the average number of probes required to insert a key into a table with this load factor.) It has long been known that for uniform hashing the average probe length is $(m+1)/(m-\alpha m+1) = (1-\alpha)^{-1} + O(m^{-1})$; see [10] for more information and a history of research on this problem. A number of papers [14, 1, 15] have shown that uniform hashing is optimal among open addressing schemes in certain senses. Ullman [14] showed that no hash function could consistently outperform uniform hashing for average insertion cost. Ajtai, Komlós and Szemerédi [1] showed that a class of functions called single-hashing functions, in which a probe sequence is determined by its first element, could not asymptotically outperform uniform hashing for retrieval costs. Yao [15] generalized the proof of [1] to all open adressing hashing algorithms, but left open the question of lower bounds for insertion costs. (Note however that hashing algorithms which lie outside the class described above can give improved performance. In particular, by appropriately moving old keys when inserting a new one, substantial improvements are possible. For a survey of such methods see [3, Sections 3.3.7 and 3.3.8].)

Despite the good behavior of uniform hashing, it is desirable to find hash functions which are easier to compute. In a technique called *double hashing*, the probe sequence is determined by two integer-valued hash functions, namely, the *primary hash function* $h_1(K)$ and the *secondary hash function* $h_2(K)$, which determine the probe sequence

$$h_1(K) - ih_2(K) \bmod m, \quad \text{for} \quad i = 0, 1, \ldots, m-1.$$

We will let the term *hash pair* refer to the pair $(h_1(K), h_2(K))$. As in [8], we assume that these hash functions and the distribution of keys are such that

$$\Pr\{(h_1(K), h_2(K)) = (i,j)\} = \frac{1}{m(m-1)}$$

for all (i,j) with $0 \le i \le m-1$ and $1 \le j \le m-1$. Let $P^A_{\alpha,m}(k)$ be the probability, when using algorithm A, that the probe length during an unsuccessful search of a table, which has been filled to a load factor α, is at least k. UH will stand for uniform hashing and DH for double hashing. So that $P^A_{\alpha,m}(k)$ is defined when αm is not an integer, extend it by $P^A_{\alpha,m}(k) = P^A_{\lfloor \alpha m \rfloor/m,m}(k)$. Note that

$$C'_\alpha(m) = \sum_{k=1}^{m} P^A_{\alpha,m}(k). \tag{1.1}$$

To minimize the amount of notation in the paper, we will adopt the following convention (though we will avoid the use of this terminology in the statement of theorems). The constant c will refer to a constant whose value will be left free. We

will say that a quantity is c-polysmall if for all positive p, for large enough c the quantity is $O(m^{-p})$. Note that the sum of polynomially many quantities which are c-polysmall is again c-polysmall; if we can assert that $f(m,c)$ is c-polysmall except for small m, the phrase "except for small m" may be dropped; and if $f(m,2c)$ is c-polysmall, then so is $f(m,c)$.

Also, throughout the paper we will let α be some arbitrary but fixed constant in the range $0<\alpha<1$, and let m range only over prime values; hidden constants may depend on the choice of α.

Our goal is to prove that double hashing is asymptotically equivalent to uniform hashing for load factors arbitrarily close to 1. In fact, we can show that the distribution of the number of probes in an unsuccessful search is close to that obtained with uniform hashing.

Theorem 1. *For each fixed $\alpha\in(0,1)$ and each $p\geq 1$, we can choose a constant c so that if $\delta=cm^{-1/2}\log^{5/2}m$, then*

$$P^{\mathrm{DH}}_{\alpha,m}(k)\leq P^{\mathrm{UH}}_{(1+\delta)\alpha,m}(k)+O(m^{-p})$$

and

$$P^{\mathrm{DH}}_{\alpha,m}(k)\geq P^{\mathrm{UH}}_{(1-\delta)\alpha,m}(k)-O(m^{-p}),$$

where the hidden constants in the O-notation are independent of k and m, and m is restricted to assume only prime values.

In view of (1.1) this immediately yields

Corollary 1. *For double hashing, for each fixed α with $0<\alpha<1$,*

$$C'_\alpha(m)=\frac{1}{1-\alpha}+O(m^{-1/2}\log^{5/2}m).$$

The implied lower bound of this corollary is of course not very surprising, especially in view of [14, 1, 15].

By a hash table *configuration* we mean the set of indices of the filled slots. A key technique in our paper is the modification of the distribution of table configurations in a way which a) dominates the distribution that would be obtained by the original algorithm, except with very small probability (in a sense made more precise later), b) forces the table to be equivalent to one obtained by uniform hashing, and c) causes only a very small change in table performance. The technique is similar in principle to a resampling technique used in [9]. There a distribution which was nearly uniform was converted into a truly uniform distribution by a sampling procedure which rejected a few points. In our case, we will produce a table equivalent to uniform hashing by carefully adding a few extra items to the hash table. These extra points will be colored red, while the original items will be colored green. Our addition of red points is in some ways similar to the randomization proof strategy used in [1, 15].

The following special case of the Hoeffding bound will be useful.

Lemma 1 [4]. *Let X be the binomially distributed random variable giving the number of successes in n Bernoulli trials each having success probability p_0. Then for $\beta \geq 0$,*

$$\Pr\{X \geq (p_0 + \beta)n\} \leq \exp(-2n\beta^2),$$

and

$$\Pr\{X \leq (p_0 - \beta)n\} \leq \exp(-2n\beta^2).$$

In section 2 we observe that if a table is partially filled according to uniform hashing, but then we insert one more key according to double hashing, the distribution of the location into which the key will be inserted is, with high probability, nearly uniform. In section 3 we show how to use randomization to obtain a remarkably simple proof of Theorem 1.

2. The distribution of $\hat{\xi}(x)$ in the uniform case

In this section we note that if we have a table which has been partially filled by uniform hashing, placing the next point by double hashing is very nearly the same as placing it by uniform hashing, except with small probability. This sort of result is implicit in [8], and we make considerable use in this section of the insights and machinery of that paper. Using an elegant approach suggested to us by Pippenger [12], we give a rather simple proof of the precise form (Lemma 3) we will want.

For any table configuration, for every empty slot x, let $\hat{\xi}(x)$ be the number of hash pairs which would cause x to be the next slot filled. As in [8], we can readily use this to express the probability $\xi(x)$ that x is the next position filled using double hashing:

$$\xi(x) = \frac{\hat{\xi}(x)}{m(m-1)}. \tag{2.1}$$

For convenience, we extend the domain of $\hat{\xi}(x)$ to include the full slots by defining it to be the number of hash pairs which would cause x to be filled had it been empty. Equivalently, for any x, $\hat{\xi}(x)$ is the number of hash pairs which would cause x to be probed. Note that if we have a table filled by uniform hashing, then by symmetry the distribution of $\hat{\xi}(x)$ does not depend on x. Thus in the following we refer to $\hat{\xi}(x)$ without specifying x.

Lemma 2. *Let $0 \leq p \leq (2+\alpha)/3$. Suppose that in a hash table of size m, we construct a table configuration by letting each slot, independently, be filled with probability p. Then*

$$\hat{\xi}(x) \leq \frac{m}{1-p}\left(1 + \frac{(c \log m)^{5/2}}{m^{1/2}}\right)$$

except with probability which is c-polysmall.

Proof. We will use the notion of progressions like those used in [8]. Call the set $\{x+ld, x+(l-1)d, x+(l-2)d, \ldots, x+d \pmod m\}$ of slots the *potential progression of*

length l generated by stride d coming to x. If all slots in this potential progression are filled, we can omit the word potential and say it is a *progression of length l actually generated by stride d coming to x.* Thus for fixed l and x, each stride generates just one potential progression, and actually generates a progression with probability p^l under the probability distribution of this lemma. If we adopt the convenient fiction that the $m-1$ progressions of length 0 coming to x generated by the strides $\{1,2,\ldots,m-1\}$ are distinct, we can set up a correspondence between progressions of length $l \geq 0$ generated by strides d and hash pairs $(x+ld,d)$ which would cause x to be probed. Thus $\hat{\xi}(x)$ equals the total number of progressions coming to x. For $0 \leq l \leq m-1$ let M_l be the number of actual progressions of length l coming to x; note that by our above convention $M_0 = m-1$. Then

$$\hat{\xi}(x) = \sum_{l=0}^{m-1} M_l. \tag{2.2}$$

We now fix l and x and seek to estimate M_l, which is the number of strides d which actually generate a progression of length l coming to x. Let

$$k = c \log m. \tag{2.3}$$

We first bound M_l in the case where $l \leq k$. For the present assume that

$$l \geq 2. \tag{2.4}$$

Note that there are $m-1$ possible strides, and that by the probability distribution in this lemma it is clear that each gives a progression of length l coming to x with probability p^l. Thus we might hope to use Lemma 1 to control the value of M_l, but there is of course a problem: while each slot is filled independently, the sets of slots used by various strides are not necessarily disjoint. An elegant solution to this problem, from [12], partitions the strides into subsets in which the strides do not use common slots.

First let us say that two strides d and d' *interact* if the two potential progressions of length l which they generate intersect, i.e., if

$$\begin{aligned} &\{x+ld, x+(l-1)d, x+(l-2)d, \ldots, x+d\} \\ &\cap \{x+ld', x+(l-1)d', x+(l-2)d', \ldots, x+d'\} \neq \emptyset. \end{aligned} \tag{2.5}$$

If they do not interact we will say they are *independent.* Note that the events specifying the presence of these two progressions are independent in the probabilistic sense if and only if the strides are independent in the sense just defined. One readily establishes that for any d,

$$|\{d' : d \text{ and } d' \text{ interact}\}| \leq l^2. \tag{2.6}$$

To see this, note that if one fixes d, chooses one element from each of the two sets in (2.5), and equates these elements, then there is a unique solution for d' — but there are exactly l^2 ways to choose one element from each set.

Thus if we form a graph in which vertices correspond to strides and strides which interact are adjacent, then no vertex has degree exceeding $l^2 - 1$. (Recall

that d itself is included in the bound of (2.6).) Clearly such a graph is l^2-colorable, so we can partition its vertices into l^2 independent sets $D_1, D_2, \ldots, D_{l^2}$.

Let $s_1, s_2, \ldots, s_{l^2}$ be the number of strides in $D_1, D_2, \ldots, D_{l^2}$ (respectively) which actually generate progressions, so that

$$M_l = \sum_{i=1}^{l^2} s_i. \tag{2.7}$$

Now we know that each of the s_i is binomially distributed, and we can use Lemma 1 to assert that for each i, if we choose

$$\beta_i = \sqrt{c \log m}/\sqrt{|D_i|},$$

then

$$\Pr\left\{s_i \geq (p^l + \beta_i)|D_i|\right\}$$

is c-polysmall. Hence by (2.7) we can assert that except with c-polysmall probability

$$\begin{aligned} M_l &= \sum_{i=1}^{l^2} s_i \\ &\leq \sum_{i=1}^{l^2} p^l |D_i| + \sum_{i=1}^{l^2} \beta_i |D_i| \\ &\leq p^l \sum_{i=1}^{l^2} |D_i| + \sum_{i=1}^{l^2} \sqrt{(c \log m)|D_i|} \\ &\leq p^l m + \sqrt{c \log m} \sum_{i=1}^{l^2} \sqrt{|D_i|} \\ &\leq p^l m + \sqrt{c \log m} l^2 \sqrt{l^{-2} \sum_{i=1}^{l^2} |D_i|} \leq p^l m + l\sqrt{cm \log m}, \end{aligned} \tag{2.8}$$

where we have used the concavity of the square root function to obtain the last line. We can now drop the assumption (2.4) since if $l=0$ then $M_l = m-1$ so (2.8) is trivial, and if $l=1$ then M_l is just the number of filled slots (other than x) so (2.8) holds except with c-polysmall probability from Lemma 1.

For $l > k$, a stronger bound on M_l is readily obtained. For such l the probability that any given stride actually generates a progression of length l is bounded by p^k, which is c-polysmall since $p < (2+\alpha)/3 < 1$ and $k = c \log m$. Hence, for $l > k$ the probability that M_l exceeds 0 is c-polysmall.

Thus by (2.2) we can now conclude that, except with c-polysmall probability,

$$\begin{aligned}\hat{\xi}(x) &\le \sum_{l=0}^{k} M_l \\ &\le \sum_{l=0}^{k} \left(p^l m + l(cm\log m)^{1/2}\right) \\ &\le \frac{m}{1-p} + \frac{k(k+1)}{2}(cm\log m)^{1/2} \\ &\le \frac{m}{1-p} + (c\log m)^{5/2} m^{1/2}\end{aligned}$$

where we have used (2.3) in the last step. ∎

In the following and in the next section, the idea of dominating configurations is crucial. We say a configuration C_1 *dominates* configuration C_2 if $C_1 \supseteq C_2$. Note that $\hat{\xi}$ is *monotonic* in the configuration, in the sense that adding elements to the configuration can not decrease $\hat{\xi}(x)$.

Lemma 3. *Let $0 \le \nu \le (1+\alpha)/2$. Suppose we construct a hash table configuration by starting with an empty table of size m and adding νm points according to uniform hashing. Then*

$$\hat{\xi}(x) \le \frac{m}{1-\nu}\left(1 + \frac{(c\log m)^{5/2}}{m^{1/2}}\right)$$

except with probability which is c-polysmall.

Proof. If we pick $p = \nu(1 + cm^{-1/2}\log m)$, then the distribution of Lemma 2 will include at least νm points except with probability which is c-polysmall. The lemma then follows from the fact $\hat{\xi}$ is monotonic. ∎

An elegant alternative proof of Lemma 3 was suggested to us by the referee. We begin with the following lemma.

Lemma 4. *Let X be a random variable having a binomial distribution with mean μ. Let γ be a monotone nondecreasing function defined on the range of X. Then $\gamma(\lfloor\mu\rfloor) < 2\mathsf{E}[\gamma(X)]$.*

Proof. Using the fact that γ is a monotone nondecreasing function, we write

$$\gamma(\lfloor\mu\rfloor) = \mathsf{E}[\gamma(X) | X = \lfloor\mu\rfloor] \le \mathsf{E}[\gamma(X) | X \ge \lfloor\mu\rfloor].$$

Now since

$$\Pr\{A|H\} = \frac{\Pr\{AH\}}{\Pr\{H\}} \le \frac{\Pr\{A\}}{\Pr\{H\}},$$

we can show that

$$\mathsf{E}[\gamma(X) | X \ge \lfloor\mu\rfloor] \le \frac{\mathsf{E}[\gamma(X)]}{\Pr\{X \ge \lfloor\mu\rfloor\}}.$$

From [5, equation (1.4)] we conclude that $\Pr\{X \geq \lfloor \mu \rfloor\} > \frac{1}{2}$, so $\gamma(\lfloor \mu \rfloor) < 2\mathsf{E}[\gamma(X)]$. ∎

We can now give the following alternate proof of Lemma 3.

Proof. (Sketch.) Let $p = \nu$ and define $h(C)$ by

$$h(C) = \begin{cases} 1 & \text{if } \hat{\xi}(x) > \frac{m}{1-\nu}\left(1 + \frac{(c \log m)^{5/2}}{m^{1/2}}\right) \\ 0 & \text{otherwise,} \end{cases}$$

so that if we let the distribution of configurations C be as in Lemma 2, then the excluded probability in Lemma 2 is $\mathsf{E}[h(C)]$ and the excluded probability in Lemma 3 is $\mathsf{E}[h(C) \mid |C| = \nu m]$. Now apply Lemma 4 with $X = |C|$, $\mu = \nu m$, and $\gamma(i) = \mathsf{E}[h(C) \mid |C| = i]$; note that γ is a nondecreasing function because $\hat{\xi}$ is monotonic in the configuration. ∎

3. Proof of the Theorem

We now show how Lemma 3 of the previous section combined with a randomization technique can be used to give a simple proof of Theorem 1.

The following observation will be crucial. As observed in [8, p. 255], double hashing preserves the dominance relationship of configurations: If C_1 dominates C_2, and we insert a key K into C_1 (respectively C_2) to obtain C_1' (respectively C_2'), then C_1' dominates C_2'. In particular, this means that if we occasionally add a fictitious extra point to the table, we will never cause some slot to remain empty that would otherwise have been filled.

Proof of Theorem 1. Let

$$\delta = \frac{(c \log m)^{5/2}}{m^{1/2}}. \tag{3.1}$$

To obtain the first inequality in the theorem, we will show that

$$P^{\mathrm{DH}}_{\alpha,m}(k) \leq P^{\mathrm{UH}}_{(1+2\delta)\alpha,m}(k) + (\text{terms which are } c\text{-polysmall}). \tag{3.2}$$

(By the properties of the c-polysmall notation noted in Section 1, the factor of 2 on δ in (3.2), and the movement of c inside the parenthesis in (3.1), are unimportant.)

Consider the algorithm *UsuallyDoubleHash* given below, for inserting points into the table. Let $K_1, K_2, \ldots, K_n$, where $n = \lfloor \alpha m \rfloor$, be the keys to be inserted; we will call these the *original keys* to avoid possible confusion with extra points to be added. Usually an inserted point will be one of the original keys (which will be colored green), but with small probability it will be an extra point (colored red) which we add to facilitate the analysis. We henceforth let UDH stand for *UsuallyDoubleHash*.

In order to make the structure of the randomization argument clear, we now describe the probability spaces which will be used. Fix n and m. The sequence of original keys defines a sequence of hash pairs $((i_1, j_1), (i_2, j_2), \ldots, (i_n, j_n))$ with

```
procedure UsuallyDoubleHash(n)
begin
  k:=0; /* Number of original keys inserted do far */
  f:=0; /* Number of table positions filled so far */
  while k<n and f<⌊(1+2δ)n⌋ do
    begin
      if there exists an empty slot x for which ξ(X)>(1+δ)/(m−f)
        then VeryUnlikely: exit this while-loop
        else UsualCase: if flip(1/(1+δ))
          then begin
                 k:=k+1;
                 /* Note that the probability that slot x is filled by the statement
                 below is ξ(x) */
                 insert K_k into the table according to double hashing, and color
                 it green;
                 end
          else begin
                 choose an empty table location x to be filled according to the
                 probability distribution

                                  g(x) = δ^{-1} ( (1+δ)/(m−f) − ξ(x) ),

                 and color it red;
                 end;
      f:=f+1;
    end; /* of while-loop */
  while f<⌊(1+2δ)n⌋ do
    insert an extra red point into the table according to uniform hashing;
end; /* of procedure */
```

Fig. 1. Procedure *UsuallyDoubleHash*

$i_l = h_1(K_l)$ and $j_l = h_2(K_l)$. Let $\mathcal{H}$ denote the set of all possible such sequences, of length n, of hash pairs. Once we have placed the n keys, we wish to determine the expected probe length for an unsuccessful search; let $\mathcal{U}$ be the set of $m(m-1)$ possible hash pairs for this new key. We have $|\mathcal{H}\times\mathcal{U}| = (m(m-1))^{n+1}$, and each of these elements is equally likely. On the probability space $\mathcal{H}\times\mathcal{U}$ we define the random variable $Y^{\mathrm{DH}}_{\alpha,m}$; for $(h,u)\in\mathcal{H}\times\mathcal{U}$, $Y^{\mathrm{DH}}_{\alpha,m}(h,u)$ is the integer computed as follows: fill a table according to double hashing using the hash pairs specified by h, and then return the number of probes double hashing would use to search unsuccessfully for a key with hash pair u in the resulting configuration. Then the quantity $P^{\mathrm{DH}}_{\alpha,m}(k)$

appearing in the Theorem is simply

$$P^{\mathrm{DH}}_{\alpha,m}(k) = \Pr\left\{Y^{\mathrm{DH}}_{\alpha,m} \geq k\right\}.$$

Let $\mathcal{T}$ be a probability space used to determine the values returned by **flip**, and the choices of locations for red points, in UDH. (The symbol $\mathcal{T}$ is intended as a mnemonic for the *tossing* of coins.) The probability space used in the analysis will be the product space $\mathcal{S} = \mathcal{H} \times \mathcal{U} \times \mathcal{T}$. The random variable $Y^{\mathrm{DH}}_{\alpha,m}$ can easily be extended to be defined on this space by simply ignoring the component from $\mathcal{T}$. The variable $Y^{\mathrm{UDH}}_{\alpha,m}$ maps an element $(h,u,t) \in \mathcal{S} = \mathcal{H} \times \mathcal{U} \times \mathcal{T}$ to a value computed in the following way: fill a table according to h and t by the algorithm UDH, and then return the number of probes that double hashing would use to search for a key with hash pair u in the resulting configuration. We define

$$P^{\mathrm{UDH}}_{\alpha,m}(k) = \Pr\left\{Y^{\mathrm{UDH}}_{\alpha,m} \geq k\right\}.$$

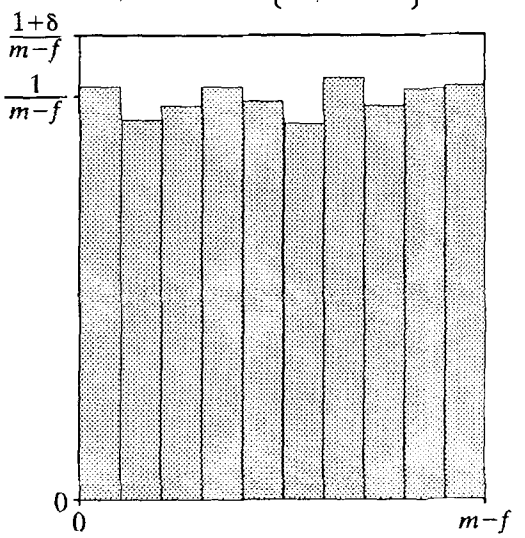

Fig. 2. A plot of $\xi(x)$, for the $m-f$ empty table positions

Figure 2 depicts a simplified possible plot of $\xi(x)$ when for all empty slots x, $\xi(x) < (1+\delta)/(m-f)$ (the "UsualCase" in the algorithm). The scales on the axes in that figure are not equal: For convenience we have scaled the horizontal axis so that each of the vertical bars has width 1. Note that the shaded region in that figure has area 1, and the white region (between $\xi(x)$ and $(1+\delta)/(m-f)$) has area δ. The probability distribution $g(x)$ used in the algorithm is just the white region, normalized to have area 1.

Note that the block labelled "UsualCase:" can be viewed as drawing a point uniformly from the rectangle shown in Figure 2, coloring it green if it is drawn from the shaded portion of that rectangle and red otherwise. (The procedure **flip** decides first which color to use, and then ξ is used to select a slot for a green point, or g is used to select a slot for a red point.) It is not difficult to see that the distribution

of table configurations (and slot colors) is unaffected if we replace the block with the following:

UsualCaseVariant: Choose the next table location x to be filled according to uniform hashing. Choose whether x is filled with a red point or a green point by letting

$$\Pr\{x \text{ is green}\} = \frac{\xi(x)}{(1+\delta)/(m-f)}.$$

We can now begin to make comparisons between the distributions of the number of probes for *UsuallyDoubleHash* (UDH), double hashing (DH), and uniform hashing (UH).

Lemma 5. *If αm is an integer, we have*

$$P^{\mathrm{UDH}}_{\alpha,m}(k) = P^{\mathrm{UH}}_{(1+2\delta)\alpha,m}(k).$$

Proof. By the UsusalCaseVariant described above, if we ignore the colors of inserted points, UDH is simply performing uniform hashing, and inserts precisely $\lfloor(1+2\delta)n\rfloor$ points. (Note that regardless of why we exit the first **while**-loop, the second loop pads the table to contain $\lfloor(1+2\delta)n\rfloor$ points.) Thus the length of a probe sequence for a new hash pair chosen independently is just the same as that for uniform hashing in a table filled to this load factor. ∎

In order to complete the proof we compare the distributions of the number of probes for DH and UDH. Usually we have $Y^{\mathrm{DH}}_{\alpha,m} \le Y^{\mathrm{UDH}}_{\alpha,m}$, though this can fail if we did not insert all n of the original keys. Let E_{fail} be this event, i.e., that $k \neq n$ at the end of UDH.

Lemma 6. *If $\omega \in \mathcal{S} - E_{\mathrm{fail}}$, then $Y^{\mathrm{DH}}_{\alpha,m}(\omega) \le Y^{\mathrm{UDH}}_{\alpha,m}(\omega)$.*

Proof. If all n keys were placed by UDH, then DH and UDH insert the same keys (with the same hash sequences), except that UDH occasionally adds an extra point. Then by the monotonicity of double hashing, the configuration produced by UDH dominates that which would have been produced by DH. ∎

Fortunately, event E_{fail} is quite unlikely.

Lemma 7. $\Pr\{E_{\mathrm{fail}}\}$ *is c-polysmall.*

Proof. First consider the case that the **exit** statement labelled "VeryUnlikely:" was executed. By Lemma 3, (2.1), and (3.1), for each insertion the probability that the next point will be placed by the block labelled "VeryUnlikely:" is c-polysmall. Since the number of insertions is $O(m)$, the probability that this block is ever executed is c-polysmall.

Next, suppose that failure occurs but VeryUnlikely is never executed. Then it must be that there were $(1+2\delta)n$ calls to **flip** but fewer than n **true**'s among the results. Hence we can bound the probability of this case by the probability that there are fewer than n successes in $(1+2\delta)n$ Bernoulli trials with success probability $1/(1+\delta)$. By Lemma 1 and (3.1) this is c-polysmall in n, and hence also in m, for any fixed $0<\alpha<1$. ∎

Proof of Theorem 1, concluded. Combining Lemmas 5, 6, and 7, we immediately obtain the upper bound given in (3.2).

We now give a brief sketch of the proof of the lower bound for $P_{\alpha,m}^{\mathrm{DH}}(k)$. A lower bound on $\hat{\xi}(x)$ analogous to the upper bound of Lemma 3 can be obtained by similar methods. We then use the procedure UsuallyDoubleHash′ shown below. This procedure does not insert additional points, but rather makes the distribution probability for filled slots uniform by occasionally rejecting a point; this is quite similar to the sort of resampling used in another context in [9]. The result is a table with configurations which are distributed as if filled by uniform hashing and which are dominated by those that would have been obtained if we had used double hashing; moreover, they are very likely to contain at least $(1-3\delta)n$ points. We omit the details. ∎

```
procedure UsuallyDoubleHash′(n)
begin
  k:=0; /* Number of original keys processed so far */
  f:=0; /* Number of table positions filled so far */
  while k<n do
    begin
      if there exists an empty slot x for which |(m−f)ξ(x)−1|>δ
        then VeryUnlikely: report failure and return
        else UsualCase:
          begin
            for each empty slot x do
                RejectProbability(x):=1−((1−δ)/(m−f))/ξ(x);
            insert K_k into the table according to double hashing, and let x denote
            the slot which is filled;
            if flip(RejectProbability(x))
              then make slot x empty again (discarding key K_k)
              else f:=f+1;
            k:=k+1;
          end /* of UsualCase */
    end /* of while-loop */
end; /* of procedure */
```

Fig. 3. Procedure UsuallyDoubleHash′

4. Conclusions and Related Work

In [2] the probability distribution of the insertion costs for a type of hashing algorithm known as *random probing with k-ary clustering* is analyzed. In this type of hashing algorithm, we assume that the key distributions and hash functions are such that the first k positions in the probe sequences for successive keys are independent. The remaining positions in probe sequences are a function of the first k; this function is chosen randomly but fixed at the beginning of the run of the algorithm. For $k = 1$, [2] give the closed form of the generating function for the asymptotic probability distribution, and for $k \geq 2$, they show that the probability distributions are asymptotically identical to that of uniform hashing, enabling them to conclude that all moments are asymptotically equal to those of uniform hashing.

The results discussed here were originally presented [11], where we only presented bounds on the expectation of $Y_{\alpha,m}^{\mathrm{DH}}$, rather than on its distribution. In Theorem 1 we now give bounds on the distribution of $Y_{\alpha,m}^{\mathrm{DH}}$, as [2] did for algorithms with k-ary clustering. In particular, in view of the bound in Theorem 1, we have the following, which answers a question mentioned in [2]. (Let $Y_{\alpha,m}^{A}$ be the random variable which specifies the probe length, when using algorithm A, during an unsuccessful search of a table which has been filled to a load factor α.)

Corollary 2. *For any fixed α and $q \geq 0$, we have (as $m \to \infty$ over the primes)*

$$\mathsf{E}\left[(Y_{\alpha,m}^{\mathrm{DH}})^q\right] \sim \mathsf{E}\left[(Y_{\alpha,m}^{\mathrm{UH}})^q\right]. \qquad \blacksquare$$

There are several ways in which these results might be strengthened. One of the most interesting, recently addressed in [13], is the question of how one might analyze open addressing schemes if we weaken the assumption that all of the keys to be inserted are independently distributed. For example, by using universal hashing schemes, one might be able to eliminate any assumptions about the distribution of keys and still enforce the condition that any set of $\log n$ hash pairs are independent. This complicates the problem considerably, but [13] were able to obtain results under such assumptions. They also strengthened the bound of Theorem 1.

Acknowledgements. We are very grateful to Nicholas Pippenger for a number of very helpful suggestions, especially for suggesting the proof approach used in Section 2 and generously allowing us to incorporate it into this paper. We also thank David Eppstein for enabling us to simplify the proof of that section by pointing out the relevance of graph coloring to the problem. We thank [13] for pointing out that bounds on the distribution of $Y_{\alpha,m}^{\mathrm{DH}}$ were implicit in [11]. We thank an anonymous referee for helpful comments, such as pointing out the alternative proof given for Lemma 3 and calling reference [5] to our attention.

References

[1] MIKLÓS AJTAI, JÁNOS KOMLÓS, and ENDRE SZEMERÉDI: There Is No Fast Single Hashing Algorithm, *Information Processing Letters* **7** (1978), 270–273.

[2] BÉLA BOLLOBÁS, ANDREI Z. BRODER and ISTVAN SIMON: The Cost Distribution of Clustering in Random Probing, *J. ACM* **37** (1990), 224-237.
[3] G. H. GONNET and R. BAEZA-YATES: *Handbook of Algorithms and Data Structures: In Pascal and C*, Second Edition, Addison–Wesley, Wokingham, England, 1991.
[4] WASSILY HOEFFDING: Probability Inequalities for Sums of Bounded Random Variables, *J. American Statistical Association* **58** (1963), 13–30.
[5] KUMAR JOG-DEV and S. M. SAMUELS: Monotone Convergence of Binomial Probabilities and a Generalization of Ramanujan's Equation, *The Annals of Mathematical Statistics* **39** (1968), 1191–1195.
[6] JÁNOS KOMLÓS: Private communication, 1986.
[7] LEO J. GUIBAS: Private communication, Fall 1987.
[8] LEO J. GUIBAS and ENDRE SZEMERÉDI: The Analysis of Double Hashing, *Journal of Computer and System Sciences* **16** (1978), 226–274.
[9] NARENDRA KARMARKAR and RICHARD M. KARP: The Differencing Method of Set Partititoning, Report No. UCB/CSD 82/113, Computer Science Division (EECS), University of California, Berkeley, December 1982.
[10] D. KNUTH: *The Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley, Reading, Mass., 1973.
[11] GEORGE S. LUEKER and MARIKO MOLODOWITCH: More Analysis of Double Hashing, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, Chicago, IL, May 1988, 354–359.
[12] NICHOLAS PIPPENGER: Private communication, January 1988.
[13] JEANETTE P. SCHMIDT and ALAN SIEGEL: On Aspects of the Universality and Performance for Closed Hashing, *Proc. 21st Annual ACM Symposium on Theory of Computing*, Seattle, WA, May 1989, 355–366.
[14] JEFFREY D. ULLMAN: A Note on the Efficiency of Hash Functions, *Journal of the ACM* **19** (1972), 569–575.
[15] ANDREW C. YAO: Uniform Hashing Is Optimal, *Journal of the ACM* **32** 3 (1985), 687–693.

George S. Lueker

Department of Information
and Computer Science
University of California, Irvine
Irvine, CA 92717
U.S.A.
lueker@ics.uci.edu

Mariko Molodowitch

California State University, Fullerton
Fullerton, CA 92634
U.S.A.
mariko@fullerton.edu