Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Information Processing Letters 86 (2003) 215–219

Information Processing Letters

www.elsevier.com/locate/ipl

Cuckoo hashing: Further analysis [☆]

Luc Devroye [*], Pat Morin

School of Computer Science, McGill University, 3480 University Street, Montreal, Canada H3A 2K6

Received 26 November 2001

Communicated by F. Dehne

Abstract

We consider cuckoo hashing as proposed by Pagh and Rodler in 2001. We show that the expected construction time of the hash table is $O(n)$ as long as the two open addressing tables are each of size at least $(1+\varepsilon)n$, where $\varepsilon > 0$ and n is the number of data points. Slightly improved bounds are obtained for various probabilities and constraints. The analysis rests on simple properties of branching processes.
© 2003 Elsevier Science B.V. All rights reserved.

MSC: 60D05; 68U05

Keywords: Open addressing; Hashing; Cuckoo hashing; Worst-case search time; Collision resolution; Probabilistic analysis of algorithms; Algorithms

1. Introduction

Cuckoo hashing is a new hashing method with very interesting worst-case properties. Introduced by Pagh and Rodler [20], it hashes n data points into two tables of size m in expected time $O(n)$ as long as $m/n > 1+\varepsilon_1$ for some $\varepsilon_1 > 0$. Once the table is constructed, each search takes at most two probes.

In hashing with chaining with a table of size $m = \lfloor \alpha n \rfloor$, where $\alpha > 0$ is a constant, the worst-case search time is equal to the length of the longest chain. If the hash values are independent and uniformly distributed over the table, then the maximum chain length is asymptotic to $\log n / \log\log n$ in probability [13,8], for any fixed value of α.

Consider now open addressing with a table of size m again, with $\alpha > 1$ fixed. If the elements have a choice of two randomly picked positions, and are placed into the slot with the least number of elements (at the time of insertion), then the maximum slot occupancy is in probability asymptotic to $\log_2 \log_2 n$ [1–4,7,18,19].

There has been interest in obtaining $O(1)$ expected worst-case performance, or even $O(1)$ deterministic worst-case performance for search in hash tables. For static hash tables, Fredman, Komlós and Szemerédi [12] proposed a solution. Czumaj and Stemann [7] showed that if each element has two randomly chosen hash positions, then with high probability, a static (off-line) chaining hash table can be constructed that has worst chain length 2, provided that the table size

[☆] The authors' research was sponsored by NSERC Grant A3456 and FCAR Grant 90-ER-0291.

[*] Corresponding author.
E-mail address: luc@cs.mcgill.ca (L. Devroye).

is at least αn for some threshold constant α. For dynamic hash tables, the early research was in the direction of dynamic perfect hash functions [9–11, 5]. Cuckoo hashing [20] is also an attempt in this direction. It stands out though through its simplicity and the promising experimental results reported by Pagh and Rodler.

The present paper only attempts to clarify the probability theoretical properties of cuckoo hashing for an abstract setting, in which all hash values are independent and uniformly distributed. For surveys on hashing, see [17] or [21].

2. Raw cuckoo hashing

In a raw cuckoo hash, each data point gets two destinations, one in each table. Let X_i and Y_i denote the target destinations for the ith data point, obtained by hashing. We will assume throughout that $X_1, Y_1, \ldots, X_n, Y_n$ are independent uniform random integers drawn from $\{1, \ldots, m\}$. Data points are inserted sequentially, and are placed in one slot in one table by the following mechanism. Data point i is placed in position X_i in table one. If this slot was empty, the insertion is complete. If the slot was previously occupied by data point k, then k is "kicked out" (hence the name "cuckoo hash") and is placed in slot Y_k in table two. If that is empty, we are done. Otherwise, if that slot was occupied by data point ℓ, then ℓ in turn is kicked out and is placed in slot X_ℓ in table one, and so forth. Clearly, there are situations in which this procedure loops forever. Pagh and Rodler set a limit $C \log n$ on the number of allowed iterations, that is, on the maximal consecutive number of element removals. If at any point during the insertion of n elements, $C \log n$ is attained, we declare the raw cuckoo hash a failure.

If a raw cuckoo hash succeeds, then searching is very efficient. Indeed, element i is either at location X_i in Table 1 or at location Y_i in Table 2, so that the search time is uniformly bounded. We will show the following:

Theorem 1. *If $m = (1+\varepsilon)n$, for $\varepsilon \in [\varepsilon_1, M]$ for fixed $\varepsilon_1 > 0$ and $M < \infty$ (with ε possibly depending upon n and m), and if $C > 2/R(\varepsilon_1)$ where $R(t) \overset{\text{def}}{=} \log(1+t) - t/(1+t)$, then the probability that a raw cuckoo hash fails is less than $p = O(1/n)$. Furthermore, the expected time taken by a raw cuckoo hash is $O(n)$.*

3. Full cuckoo hashing

In first instance, cuckoo hashing can be used to create a good static hash table at little cost. Indeed, if a raw cuckoo hash fails, a new raw cuckoo hash is attempted, with a new collection of $2n$ hash values, picked independently from the previous bunch, and so on until a successful raw cuckoo hash is observed. One may argue that the requirement that for each element i, an infinite sequence of independent hash values be available, is unreasonable. There are ways of creating, for each i, sequences that are nearly independent, following the example of double hashing, and thus, having noted this caveat, we will just maintain our assumption. If $T_1, T_2, \ldots$ denote the computation times for the various hashing and rehashing steps, and if N denotes the total number of steps, we have by Wald's lemma,

$$\mathrm{E}\left\{\sum_{i=1}^{N} T_i\right\} = \mathrm{E}\{N\}\mathrm{E}\{T_1\} = \frac{1}{1-p} \times O(n).$$

Therefore, in linear expected time, we can find and complete a successful raw cuckoo hash. This set-up time is very reasonable, and makes the method interesting.

The purpose of the remainder of this short note is to prove Theorem 1. Our approach is different from that of Pagh and Rodler [20], as we appeal more directly to a graph-theoretic interpretation, and use standard arguments from branching processes.

4. The cuckoo graph

Consider the bipartite graph on $\{1, \ldots, m\} \times \{1, \ldots, m\}$ with n edges, where each edge is chosen uniformly at random (and repetitions are thus possible). The graph thus obtained represents the hash value pairs for a raw cuckoo hash. Consider the graph's connected components. If one connected component is not a tree or unicyclic [note: a cycle is a path that begins and ends at one node, such that no edge is traversed twice; a graph is unicyclic if it has exactly one cycle], then it is impossible that the raw cuckoo hash is successful, as that connected component has k nodes and $k+1$ or more edges for some integer k. Vice versa, if a component is a tree or is unicyclic, then an infinite

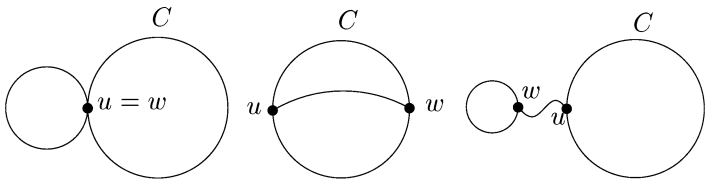

Fig. 1. The possible values of w: (a) $w = u$, (b) $w \neq u$ on C, (c) w not on C.

loop in the cuckoo heuristic is impossible—in fact, if the component has k nodes, then the insertion of one element can always be carried out in at most $2k$ iterations. To prove this, consider the walk W in the cuckoo graph that corresponds to the table cells visited during an insertion. If W has no repeated vertices then it has length at most k and the proof is complete. So, let u be the first vertex of W that is repeated. Then the cuckoo graph contains a cycle C involving u. Furthermore, the edge used by W immediately following the second occurrence of u is not an edge of C. Now consider the walk W' that begins just after the first occurrence of u in W and then follows W. Suppose there is a vertex that is repeated in W' and let w be the first such vertex. Then, either w is a vertex of C or not (see Fig. 1). In either case, the cuckoo graph contains a cycle that uses an edge not in C and is not unicyclic. Therefore, the walk W' contains no repeated vertices. By construction, $W \setminus W'$ also contains no repeated vertices. We conclude that W contains at most $2k$ vertices, as required.

If the $2n$ nodes in the graph are generically denoted by u, and if $S(u)$ denotes the size of the connected component to which u belongs, the time taken by a raw cuckoo hash (regardless of success or failure) is bounded from above by

$$\sum_u \min\bigl(2S(u), C\log n\bigr).$$

Lemma 1 (A bound for the largest connected component). *Let $m > n$, $c > 0$.*

$$\mathrm{P}\Bigl\{\max_u S(u) \geqslant c\log n\Bigr\} \leqslant 2m\mathrm{e}^{c\log n((m-n)/m-\log(m/n))}.$$

In particular, with $m = (1+\varepsilon)n$ and $\varepsilon > 0$, we have

$$\mathrm{P}\Bigl\{\max_u S(u) \geqslant c\log n\Bigr\} \leqslant 2(1+\varepsilon)n^{1-cR(\varepsilon)}.$$

Proof. Consider each of the $2m$ nodes in our bipartite graph, and fix a node u. Let Z_1 be the neighbors of u in the graph, where neighbors may be repeated. Clearly, Z_1 has a binomial $(n, 1/m)$ distribution. These neighbors act independently and have in turn Z_2' neighbors not among nodes already counted. Stochastically, Z_2' is bounded from above by Z_2, a sum of Z_1 i.i.d. binomial $(n, 1/m)$ random variables. In other words, $S(u)$ is bounded from above by the size of a Galton–Watson branching process for the binomial $(n, 1/m)$ distribution. For definitions and basic results, see [14]. As we have $n/m \leqslant 1/(1+\varepsilon_1) < 1$, this process is subcritical and thus extinct. Also,

$$\mathrm{E}\bigl\{S(u)\bigr\} \leqslant 1 + (n/m) + (n/m)^2 + \cdots = \frac{1}{1-n/m} = \frac{m}{m-n}.$$

Thus,

$$\mathrm{E}\Bigl\{\sum_u S(u)\Bigr\} \leqslant \frac{2m^2}{m-n}.$$

We are interested in tail bounds on $S(u)$. The easiest way to proceed from here is to consider the random walk presentation of such branching processes, which is equivalent to generating and visiting the Galton–Watson tree in preorder. As a node is "considered" (expanded), it receives a binomial $(n, 1/m)$ number of children, which are placed in a queue of nodes to be expanded. This process continues until no further nodes can be expanded (the queue is empty). Thus, set $N_0 = 1$, $B_1, B_2, \ldots$ i.i.d. binomial $(n, 1/m)$, and

$$N_{k+1} = \begin{cases} 0 & \text{if } N_k = 0, \\ \max(N_k + B_{k+1} - 1, 0) & \text{otherwise.} \end{cases}$$

Then

$$S(u) \leqslant \max\{k > 0\colon N_k > 0\} = \max\Biggl\{k > 0\colon 1 + \sum_{j=1}^{k}(B_j - 1) > 0\Biggr\}.$$

In particular,

$$
\begin{aligned}
&\mathrm{P}\bigl\{S(u)\geqslant k\bigr\}\\
&\quad\leqslant \mathrm{P}\left\{1+\sum_{j=1}^{k}(B_j-1)\geqslant 1\right\}\\
&\quad= \mathrm{P}\left\{\sum_{j=1}^{k}B_j\geqslant k\right\}\\
&\quad= \mathrm{P}\bigl\{\mathrm{binomial}(nk,1/m)\geqslant k\bigr\}\\
&\quad\leqslant \mathrm{E}\bigl\{\mathrm{e}^{\lambda\,\mathrm{binomial}(nk,1/m)-\lambda k}\bigr\}\\
&\qquad\qquad\text{(by Chernoff's bound [6], where } \lambda>0,\\
&\qquad\qquad\quad\text{see also [15,16])}\\
&\quad= \bigl(1-1/m+(1/m)\mathrm{e}^{\lambda}\bigr)^{nk}\mathrm{e}^{-\lambda k}\\
&\quad\leqslant \mathrm{e}^{(\mathrm{e}^{\lambda}-1)nk/m-\lambda k}\\
&\quad\leqslant \mathrm{e}^{(m-n)k/m-k\log(m/n)}\\
&\qquad\qquad\text{(upon taking } \lambda=\log(m/n)).
\end{aligned}
$$

We note that

$$
\mathrm{P}\Bigl\{\max_u S(u)\geqslant c\log n\Bigr\}
\leqslant 2m\mathrm{e}^{c\log n((m-n)/m-\log(m/n))}.
$$

If $m=n(1+\varepsilon)$, we obtain

$$
\mathrm{P}\Bigl\{\max_u S(u)\geqslant c\log n\Bigr\}\leqslant 2(1+\varepsilon)n^{1-cR(\varepsilon)}. \qquad \square
$$

Remark 1. Observe that

$$
R(\varepsilon)\geqslant\frac{\varepsilon^2(1-\varepsilon)}{2(1+\varepsilon)}>0 \quad \text{for } \varepsilon\in(0,1),
$$

and that $R(\varepsilon)$ is increasing.

Next we offer the following result regarding the structure of the graph.

Lemma 2. *Let m be as in Theorem* 1. *Let r denote the probability that a given cuckoo graph has at least one connected component that is not a tree or unicyclic. Then $r=\mathrm{O}(1/n)$.*

Proof. The probability that a given selection of $k+1$ edges cover at most ℓ and $k-\ell$ slots in tables one and two respectively, is

$$
\left(\frac{\ell(k-\ell)}{m^2}\right)^{k+1}.
$$

Thus, the probability that there exists a subgraph of size k that receives at least $k+1$ edges does not exceed

$$
\begin{aligned}
&\sum_{\ell=1}^{k-1}\binom{m}{\ell}\binom{m}{k-\ell}\binom{n}{k+1}\left(\frac{\ell(k-\ell)}{m^2}\right)^{k+1}\\
&\quad\leqslant\sum_{\ell=1}^{k-1}\frac{m^k n^{k+1}}{\ell!(k-\ell)!(k+1)!}\left(\frac{\ell(k-\ell)}{m^2}\right)^{k+1}\\
&\quad\leqslant\sum_{\ell=1}^{k-1}\frac{n^{k+1}\ell^{k+1}(k-\ell)^{k+1}\mathrm{e}^{2k+1}}{m^{k+2}\ell^{\ell}(k-\ell)^{k-\ell}(k+1)^{k+1}\sqrt{(2\pi)^3\ell(k-\ell)(k+1)}}\\
&\qquad\qquad\text{(where we used Stirling's inequality } n!\geqslant(n/\mathrm{e})^n\sqrt{2\pi n})\\
&\quad=\sum_{\ell=1}^{k-1}\frac{n^{k+1}(\frac{\ell}{k})^{k-\ell+1}(\frac{k-\ell}{k})^{\ell+1}\mathrm{e}^{2k+1}}{m^{k+2}k^{k+2}(k+1)^{k+1}\sqrt{(2\pi)^3(\frac{\ell}{k})(1-\frac{\ell}{k})k^2(k+1)}}\\
&\quad\leqslant\frac{1}{n}\sum_{\ell=1}^{k-1}p(\ell,k),
\end{aligned}
$$

where

$$
\begin{aligned}
p(\ell,k)&=\frac{(\frac{\ell}{k})^{k-\ell+1/2}(\frac{k-\ell}{k})^{\ell+1/2}\mathrm{e}^{2k}}{k^{2k+9/2}}\\
&\leqslant(\mathrm{e}/k)^{2k}/k^{9/2}.
\end{aligned}
$$

Note that

$$
\sum_{\ell=1}^{k-1}p(\ell,k)\leqslant(\mathrm{e}/k)^{2k}/k^{7/2}.
$$

Thus,

$$
r\leqslant(1/n)\sum_{k\geqslant 2}\bigl((\mathrm{e}/k)^{2k}/k^{7/2}\bigr)=\mathrm{O}(1/n). \qquad \square
$$

Completion of the proof of Theorem 1. The first part follows immediately from Lemmas 1 and 2. If all components are trees or unicyclic, and if all components are smaller than or equal to $(C/2)\log n$, then the time of a raw cuckoo hash is bounded by

$$
2\sum_u S(u).
$$

Otherwise, the time is bounded by

$$
Cn\log n.
$$

Let us use q for the upper bound of Lemma 1, with $c=C/2$. The expected time is thus bounded by

$$
(r+q)Cn\log n+2\mathrm{E}\Bigl\{\sum_u S(u)\Bigr\}.
$$

The first term on the right-hand side is $\mathrm{O}(\log n)$ as $r = \mathrm{O}(1/n)$, and q decreases exponentially quickly in n provided that $C/2 > 1/R(\varepsilon)$. Using an estimate from the proof of Lemma 1, the second term is bounded by

$$\frac{4m^2}{m-n} = \mathrm{O}(n). \qquad \square$$

5. Dynamic hashing

The results above can also be used for a dynamic version of the algorithm. Consider the insertion of one point in a table. If after $C \log n$ moves, there is still no collision resolution, the insertion of the new point is deemed a failure, and the entire table is rehashed. The probability of a rehash is clearly bounded by the probability that the connected component containing the new node is not a tree, or not unicyclic, or of size more than $(C/2)\log n$. The probability of this happening is less than Q. Thus, the expected time of an insert is bounded by the expected size of the connected component (which is less than $m/(m-n)$, as we saw earlier) plus $Q \times \mathrm{O}(n)$. The latter contribution is clearly $\mathrm{O}(1)$, as we showed that $Q = \mathrm{O}(1/n)$. Thus, the overall expected time per insert is bounded by $m/(m-n) + \mathrm{O}(1) = \mathrm{O}(1)$.

References

[1] Y. Azar, A.Z. Broder, A.R. Karlin, E. Upfal, Balanced allocations (extended abstract), in: Proceedings of the 26th ACM Symposium on the Theory of Computing, 1994, pp. 593–602.

[2] Y. Azar, A.Z. Broder, A.R. Karlin, E. Upfal, Balanced allocations, SIAM J. Comput. 29 (1999) 180–200.

[3] A.Z. Broder, A.R. Karlin, Multilevel adaptive hashing, in: Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, PA, 1990, pp. 43–53.

[4] A.Z. Broder, M. Mitzenmacher, Using multiple hash functions to improve IP lookups, in: INFOCOM 2001, 2001.

[5] A. Brodnik, I. Munro, Membership in constant time and almost-minimum space, SIAM J. Comput. 28 (1999) 1627–1640.

[6] H. Chernoff, A measure of asymptotic efficiency of tests of a hypothesis based on the sum of observations, Ann. Math. Statist. 23 (1952) 493–507.

[7] A. Czumaj, V. Stemann, Randomized allocation processes, in: Proceedings of the 38th IEEE Symposium on Foundations of Computer Science (FOCS'97), October 19–22, 1997, Miami Beach, FL, 1997, pp. 194–203.

[8] L. Devroye, The expected length of the longest probe sequence when the distribution is not uniform, J. Algorithms 6 (1985) 1–9.

[9] M. Dietzfelbinger, F. Meyer auf de Heide, A new universal class of hash functions and dynamic hashing in real time, in: Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP'90), in: Lecture Notes in Computer Science, Vol. 443, Springer, Berlin, 1990, pp. 6–19.

[10] M. Dietzfelbinger, J. Gil, Y. Matias, N. Pippenger, Polynomial hash functions are reliable (extended abstract), in: Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92), in: Lecture Notes in Computer Science, Vol. 623, 1992, pp. 235–246.

[11] M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F. Meyer auf de Heide, H. Rohnert, R.E. Tarjan, Dynamic perfect hashing: upper and lower bounds, SIAM J. Comput. 23 (1994) 738–761.

[12] M.L. Fredman, J. Komlós, E. Szemerédi, Storing a sparse table with O(1) worst case access time, J. ACM 31 (1984) 538–544.

[13] G.H. Gonnet, Expected length of the longest probe sequence in hash code searching, J. ACM 28 (1981) 289–304.

[14] G.R. Grimmett, D.R. Stirzaker, Probability and Random Processes, Oxford University Press, 1992.

[15] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Amer. Statist. Assoc. 58 (1963) 13–30.

[16] S. Janson, Poisson Approximation, Oxford University Press, 1992.

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

[18] M. Mitzenmacher, Studying balanced allocations with differential equations, Technical Note 1997024, Digital Equipment Corporation Systems Research Center, Palo Alto, CA, 1997.

[19] M. Mitzenmacher, A.W. Richa, R. Sitaraman, The power of two random choices: a survey of techniques and results, Technical Report, 2000.

[20] R. Pagh, F.F. Rodler, Cuckoo hashing, BRICS Report Series RS-01-32, Department of Computer Science, University of Aarhus, 2001.

[21] J.S. Vitter, P. Flajolet, Average-case analysis of algorithms and data structures, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity, MIT Press, Amsterdam, 1990, pp. 431–524.