CUCKOO HASHING: FURTHER ANALYSIS

Luc Devroye and Pat Morin
School of Computer Science
McGill University
Montreal, Canada H3A 2K6

November 9, 2001

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 $(1+\epsilon)n$, where $\epsilon > 0$ 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.

KEYWORDS AND PHRASES. Open addressing, hashing, cuckoo hashing, worst-case search time, collision resolution, probabilistic analysis of algorithms.

CR CATEGORIES: 3.74, 5.25, 5.5.

1991 MATHEMATICS SUBJECT CLASSIFICATIONS: 60D05, 68U05.

Authors' address: School of Computer Science, McGill University, 3480 University Street, Montreal, Canada H3A 2K6. The authors' research was sponsored by NSERC Grant A3456 and FCAR Grant 90-ER-0291. FAX number: 1–514–398-3883. Email of first author: `luc@cs.mcgill.ca`.

Introduction

Cuckoo hashing is a new hashing method with very interesting worst-case properties. Introduced by Pagh and Rodler (2001), it hashes n data points into two tables of size m in expected time $O(n)$ as long as $m/n > 1 + \epsilon_1$ for some $\epsilon_1 > 0$. 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 (Gonnet, 1981; Devroye, 1985), 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 $\log_2 \log_2 n$ (Azar, Broder, Karlin and Upfal, 1994, 1999; Broder and Karlin, 1990; Czumaj and Stemann, 1997; Mitzenmacher, 1997).

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 (1984) proposed a solution. Czumaj and Stemann (1997) 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 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 (Dietzfelbinger and Meyer auf der Heide (1990), Dietzfelbinger, Gil, Matias and Pippenger (1992), Dietzfelbinger, Karlin, Mehlhorn, Meyer auf der Heide, Rohnert and Tarjan (1994), Brodnik and Munro (1999)). Cuckoo hashing (Pagh and Rodler, 2001) 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 Knuth (1973) or Vitter and Flajolet (1990).

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 i-th 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 $m = (1+\epsilon)n$, for $\epsilon \in [\epsilon_1, M]$ for fixed $\epsilon_1 > 0$ and $M < \infty$ (with ϵ possibly depending upon n and m), and if $C > 2/R(\epsilon_1)$ where $R(t) \overset{\text{def}}{=} \log(1+t) - t/(1+t)$, then the probability that a raw cuckoo hash fails is less than $p = O((\log n)^4/n)$. Furthermore, the expected time taken by a raw cuckoo hash is $O(n)$.*

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,

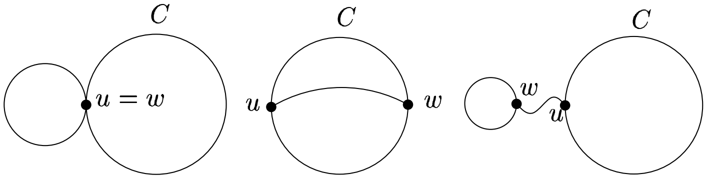
$$\mathbf{E}\left\{\sum_{i=1}^{N} T_i\right\} = \mathbf{E}\{N\}\mathbf{E}\{T_1\} = \frac{1}{1-p} \times O(n) \ .$$

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 (2001), as we appeal more directly to a graph-theoretic interpretation, and use standard arguments from branching processes.

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 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 figure). 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.



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

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

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

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

$$\mathbf{P}\left\{\max_u S(u) \ge c \log n\right\} \le 2me^{c \log n((m-n)/m - \log(m/n))} \ .$$

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

$$\mathbf{P}\left\{\max_u S(u) \ge c \log n\right\} \le 2(1+\epsilon)n^{1-cR(\epsilon)} \ .$$

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 Grimmett and Stirzaker (1992). As we have $n/m \le 1/(1+\epsilon_1) < 1$, this process is subcritical and thus extinct. Also,

$$\mathbf{E}\{S(u)\} \le 1 + (n/m) + (n/m)^2 + \cdots = \frac{1}{1 - n/m} = \frac{m}{m-n} \ .$$

Thus,

$$\mathbf{E}\left\{\sum_u S(u)\right\} \le \frac{2m^2}{m-n} \ .$$

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

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

Then

$$S(u) \le \max\{k > 0 : N_k > 0\} = \max\left\{k > 0 : 1 + \sum_{j=1}^{k}(B_j - 1) > 0\right\}$$

In particular,

$$\begin{aligned} \mathbf{P}\{S(u) \ge k\} &\le \mathbf{P}\left\{1 + \sum_{j=1}^{k}(B_j - 1) \ge 1\right\} \\ &= \mathbf{P}\left\{\sum_{j=1}^{k} B_j \ge k\right\} \\ &= \mathbf{P}\left\{\text{binomial}(nk, 1/m) \ge k\right\} \\ &\le \mathbf{E}\left\{e^{\lambda \text{binomial}(nk,1/m) - \lambda k}\right\} \\ &\qquad \text{(by Chernoff's bound (Chernoff, 1952), where } \lambda > 0) \end{aligned}$$

$$
\begin{aligned}
&= \left(1 - 1/m + (1/m)e^{\lambda}\right)^{nk} e^{-\lambda k} \\
&\leq e^{\left(e^{\lambda}-1\right)nk/m-\lambda k} \\
&\leq e^{(m-n)k/m-k\log(m/n)} \\
&\qquad \text{(upon taking } \lambda = \log(m/n)) \ .
\end{aligned}
$$

We note that

$$\mathbb{P}\{\max_u S(u) \geq c\log n\} \leq 2me^{c\log n((m-n)/m-\log(m/n))} \ .$$

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

$$\mathbb{P}\{\max_u S(u) \geq c\log n\} \leq 2(1+\epsilon)n^{1-cR(\epsilon)} \ . \square$$

Remark 1. Observe that $R(\epsilon) \geq \frac{\epsilon^2(1-\epsilon)}{2(1+\epsilon)} > 0$ for $\epsilon \in (0,1)$, and that $R(\epsilon)$ 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 $r = O((\log n)^4/n)$.*

Proof. We argue as in Lemma 1, considering the component obtained starting from a node u. We grow this component from u by adding edges of the cuckoo graph as they are discovered. Assume that u is in the first table, and that k vertices of the second table have already been discovered, and that ℓ edges have already been processed. Then the number of already discovered vertices in the second table that are hit by a new edge emanating from u is distributed as a binomial $(n-\ell, k/m^2)$ random variable, because each new random edge must independently select u in the first table and one of the k discovered vertices in the second table. This is stochastically less than a binomial $(n, K/m^2)$ random variable if $k \leq K$. Let $D(u)$ be the number of edges that find already discovered vertices when we run this process to its completion, i.e., when the entire component of u has been visited. Note that $D(u) \leq 1$ if and only if the component of u is acyclic or unicyclic. Given that $S(u) \leq K$, $D(u)$ is thus stochastically bounded by a sum of K independent binomial $(n, K/m^2)$ random variables, that is, by a binomial $(Kn, K/m^2)$ random variable, which we call B. Thus,

$$\mathbb{P}\{D(u) \geq 2 | S(u) \leq K\} \leq \mathbb{P}\{B \geq 2\} \leq (Kn)^2(K/m^2)^2 = \frac{K^4n^2}{m^4}.$$

Therefore, if u_0 is the first node of the first table, and if $c > 0$ is a given constant,

$$\begin{aligned}\mathbf{P}\{\max_u D(u) \geq 2\} &\leq \mathbf{P}\{\max_u D(u) \geq 2, \max_u S(u) < c\log n\} + \mathbf{P}\{\max_u S(u) \geq c \log n\} \\ &\leq m\mathbf{P}\{D(u_0) \geq 2 | \max_u S(u) < c\log n\} + 2(1+\epsilon)n^{1-cR(\epsilon)} \\ &\leq \frac{(c\log n)^4 n^2}{m^3} + 2(1+\epsilon)n^{1-cR(\epsilon)}.\end{aligned}$$

For $\epsilon > 0$, we find c so large that $cR(\epsilon) \geq 2$. Then the upper bound is $O((\log n)^4/n)$. □

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

$$(r+q)Cn\log n + 2\mathbf{E}\left\{\sum_u S(u)\right\}\ .$$

We have $rn\log n = O(\log^5 n)$ by Lemma 2, and $qn\log n = o(n)$ by Lemma 1 provided that $C/2 > 1/R(\epsilon)$. Using an estimate from the proof of Lemma 1, the second term is bounded by

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

References

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

Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, "Balanced allocations," *SIAM Journal on Computing*, vol. 29, pp. 180–200, 1999.

A. Z. Broder and A. R. Karlin, "Multilevel adaptive hashing," in: *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 43–53, SIAM, Philadelphia, 1990.

A. Z. Broder and M. Mitzenmacher, "Using multiple hash functions to improve IP lookups," in: *INFOCOM 2001*, pp. 0–0, 2001.

A. Brodnik and I. Munro, "Membership in constant time and almost-minimum space," *SIAM Journal on Computing*, vol. 28, pp. 1627–1640, 1999.

H. Chernoff, "A measure of asymptotic efficiency of tests of a hypothesis based on the sum of observations ," *Annals of Mathematical Statistics*, vol. 23, pp. 493–507, 1952.

A. Czumaj and 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*, pp. 194–203, 1997.

L. Devroye, “The expected length of the longest probe sequence when the distribution is not uniform,” *Journal of Algorithms*, vol. 6, pp. 1–9, 1985.

M. Dietzfelbinger and 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)*, vol. 443, pp. 6–19, Lecture Notes in Computer Science, 1990.

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

M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F. Meyer auf de Heide, H. Rohnert, and R. E. Tarjan, “Dynamic perfect hashing: upper and lower bounds,” *SIAM Journal on Computing*, vol. 23, pp. 738–761, 1994.

M. L. Fredman, J. Komlós, and E. Szemerédi, “Storing a sparse table with O(1) worst case access time,” *Journal of the ACM*, vol. 31, pp. 538–544, 1984.

G. H. Gonnet, “Expected length of the longest probe sequence in hash code searching,” *Journal of the ACM*, vol. 28, pp. 289–304, 1981.

G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes*, Oxford University Press, 1992.

W. Hoeffding, “Probability inequalities for sums of bounded random variables,” *Journal of the American Statistical Association*, vol. 58, pp. 13–30, 1963.

S. Janson, *Poisson Approximation*, Oxford University Press, 1992.

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

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

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

R. Pagh and F. F. Rodler, “Cuckoo hashing,” BRICS Report Series RS-01-32, Department of Computer Science, University of Aarhus, 2001.

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