Dietzfelbinger, Martin:

On randomness in Hash functions

Original published in:	29th International Symposium on Theoretical Aspects of Computer Science / STACS 29, 2012 Paris, France. - Saarbrücken/Wadern : Schloss Dagstuhl - Leibniz-Center for Informatics, 2012. - (2012), p. 25-28. ISBN 978-3-939897-35-4 (Leibniz International Proceedings in Informatics (LIPIcs) ; 14)
Original published:	February 2012
ISSN:	1868-8969
DOI:	10.4230/LIPIcs.STACS.2012.25
URN:	urn:nbn:de:0030-drops-33884
[*Visited:*	2020-02-27]

On Randomness in Hash Functions*

Martin Dietzfelbinger[1]

1 **Technische Universität Ilmenau**
Ilmenau, Germany
martin.dietzfelbinger@tu-ilmenau.de

Abstract

In the talk, we shall discuss quality measures for hash functions used in data structures and algorithms, and survey positive and negative results. (This talk is *not* about cryptographic hash functions.) For the analysis of algorithms involving hash functions, it is often convenient to assume the hash functions used behave fully randomly; in some cases there is no analysis known that avoids this assumption. In practice, one needs to get by with weaker hash functions that can be generated by randomized algorithms. A well-studied range of applications concern realizations of dynamic dictionaries (linear probing [37], chained hashing, dynamic perfect hashing [21], cuckoo hashing and its generalizations [30, 42]) or Bloom filters [8, 11] and their variants.

A particularly successful and useful means of classification are Carter and Wegman's *universal* or k-wise independent classes, introduced in 1977 [13, 53]. A natural and widely used approach to analyzing an algorithm involving hash functions is to show that it works if a sufficiently strong universal class of hash functions is used [10, 18, 21, 41], and to substitute one of the known constructions of such classes [2, 3, 7, 13, 16, 20, 45, 50, 51, 53]. This invites research into the question of just how much independence in the hash functions is necessary for an algorithm to work. Some recent analyses that gave impossibility results constructed rather artificial classes that would not work [15, 43]; other results pointed out natural, widely used hash classes that would not work in a particular application [1, 25, 26, 41, 43]. Only recently it was shown that under certain assumptions on some entropy present in the set of keys even 2-wise independent hash classes will lead to strong randomness properties in the hash values [14, 39]. The negative results in [25] show that these results may not be taken as justification for using weak hash classes indiscriminately, in particular for key sets with structure.

When stronger independence properties are needed for a theoretical analysis, one may resort to classic constructions [46, 47, 22]. Only in 2003 it was found out how full randomness can be *simulated* using only linear space overhead (which is optimal) [28, 40]. The "split-and-share" approach [17, 24, 30] can be used to justify the full randomness assumption in some situations in which full randomness is needed for the analysis to go through, like in many applications involving multiple hash functions (e. g., generalized versions of cuckoo hashing with multiple hash functions [19, 30, 31, 32, 33, 34, 38] or larger bucket sizes [27, 29, 12], load balancing [5], Bloom filters and variants [8, 23], or minimal perfect hash function constructions [6, 9, 17]).

For practice, efficiency considerations beyond constant factors are important. It is not hard to construct very efficient 2-wise independent classes [16, 48]. Using k-wise independent classes for constant k bigger than 3 has become feasible in practice only by new constructions [36, 50, 51] involving tabulation. This goes together well with the quite new result that linear probing works with 5-independent hash functions [41].

Recent developments suggest that the classification of hash function constructions by their degree of independence alone may not be adequate in some cases. Thus, one may want to analyze the behaviour of specific hash classes in specific applications, circumventing the concept of k-wise independence. Several such results were recently achieved concerning hash functions that utilize tabulation [44, 49]. In particular if the analysis of the application involves using

* This work was partially supported by DFG grant DI 412/10-2.

29th Symposium on Theoretical Aspects of Computer Science (STACS'12).
Editors: Christoph Dürr, Thomas Wilke; pp. 25–28
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

randomness properties in graphs and hypergraphs (generalized cuckoo hashing [30], also in the version with a "stash" [35], or load balancing [5, 52]), a hash class combining k-wise independence with tabulation has turned out to be very powerful [4, 28, 54].

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems–Computations on discrete structures–Sorting and searching, E.2 Data Storage Representations–Hash-table representations

Keywords and phrases Algorithms, hash functions, randomized algorithms, data structures, graphs, hypergraphs

Digital Object Identifier 10.4230/LIPIcs.STACS.2012.25

References

1 N. Alon, M. Dietzfelbinger, P. B. Miltersen, E. Petrank, and G. Tardos. Linear hash functions. *J. ACM*, 46(5):667–683, 1999.
2 N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992.
3 N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Addendum to "Simple construction of almost k-wise independent random variables". *Random Struct. Algorithms*, 4(1):119–120, 1993.
4 M. Aumüller. An alternative analysis of cuckoo hashing with a stash and realistic hash functions. Diplomarbeit, Technische Universität Ilmenau, March 2010.
5 Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999.
6 D. Belazzougui, F. C. Botelho, and M. Dietzfelbinger. Hash, displace, and compress. In *Proc. 17th ESA*, LNCS 5757, pages 682–693. Springer, 2009.
7 C. Bertram-Kretzberg and H. Lefmann. mod_p-Tests, almost independence and small probability spaces. *Random Struct. Algorithms*, 16(4):293–313, 2000.
8 B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.
9 F. C. Botelho, R. Pagh, and N. Ziviani. Simple and space-efficient minimal perfect hash functions. In *Proc. 10th WADS*, LNCS 4619, pages 139–150. Springer, 2007.
10 A. Z. Broder and A. R. Karlin. Multilevel adaptive hashing. In *Proc. 1st ACM-SIAM SODA*, pages 43–53, 1990.
11 A. Z. Broder and M. Mitzenmacher. Survey: Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4), 2003.
12 J. A. Cain, P. Sanders, and N. C. Wormald. The random graph threshold for k-orientiability and a fast algorithm for optimal multiple-choice allocation. In *Proc. 18th ACM-SIAM SODA*, pages 469–476. SIAM, 2007.
13 L. Carter and M. N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.
14 K.-M. Chung and S. P. Vadhan. Tight bounds for hashing block sources. In *Proc. 11th APPROX/12th RANDOM*, LNCS 5171, pages 357–370. Springer, 2008.
15 J. Cohen and D. M. Kane. Bounds on the independence required for cuckoo hashing. Manuscript, 2009. `http://math.stanford.edu/~dankane/cuchkoohashing.pdf` (last download: 2012-01-02).
16 M. Dietzfelbinger. Universal hashing and k-wise independent random variables via integer arithmetic without primes. In *Proc. 13th STACS*, LNCS 1046, pages 569–580. Springer, 1996.

17 M. Dietzfelbinger. Design strategies for minimal perfect hash functions. In *Proc. 4th SAGA*, LNCS 4665, pages 2–17. Springer, 2007.

18 M. Dietzfelbinger, J. Gil, Y. Matias, and N. Pippenger. Polynomial hash functions are reliable (extended abstract). In *Proc. 19th ICALP*, LNCS 443, pages 235–246. Springer, 1992.

19 M. Dietzfelbinger, A. Goerdt, M. Mitzenmacher, A. Montanari, R. Pagh, and M. Rink. Tight thresholds for cuckoo hashing via XORSAT. In *Proc. 37th ICALP, Part I*, LNCS 6198, pages 213–225, 2010.

20 M. Dietzfelbinger, T. Hagerup, J. Katajainen, and M. Penttonen. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms*, 25(1):19–51, 1997.

21 M. Dietzfelbinger, A. R. Karlin, K. Mehlhorn, F. Meyer auf der Heide, H. Rohnert, and R. E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. *SIAM J. Comput.*, 23(4):738–761, 1994.

22 M. Dietzfelbinger and F. Meyer auf der Heide. A new universal class of hash functions and dynamic hashing in real time. In *Proc. 17th ICALP*, LNCS 443, pages 6–19. Springer, 1990.

23 M. Dietzfelbinger and R. Pagh. Succinct data structures for retrieval and approximate membership (extended abstract). In *Proc. 35th ICALP, Part I*, LNCS 5125, pages 385–396. Springer, 2008.

24 M. Dietzfelbinger and M. Rink. Applications of a splitting trick. In *Proc. 36th ICALP, Part I*, LNCS 5555, pages 354–365. Springer, 2009.

25 M. Dietzfelbinger and U. Schellbach. On risks of using cuckoo hashing with simple universal hash classes. In *Proc. 20th ACM-SIAM SODA*, pages 795–804, 2009.

26 M. Dietzfelbinger and U. Schellbach. Weaknesses of cuckoo hashing with a simple universal hash class: The case of large universes. In *Proc. 35th SOFSEM*, LNCS 5404, pages 217–228. Springer, 2009.

27 M. Dietzfelbinger and C. Weidling. Balanced allocation and dictionaries with tightly packed constant size bins. *Theor. Comput. Sci.*, 380(1-2):47–68, 2007.

28 M. Dietzfelbinger and P. Woelfel. Almost random graphs with simple hash functions. In *Proc. 35th ACM STOC*, pages 629–638, 2003.

29 D. Fernholz and V. Ramachandran. The k-orientability thresholds for $g_{n,p}$. In *Proc. 18th ACM-SIAM SODA*, pages 459–468. SIAM, 2007.

30 D. Fotakis, R. Pagh, P. Sanders, and P. G. Spirakis. Space efficient hash tables with worst case constant access time. *Theory Comput. Syst.*, 38(2):229–248, 2005.

31 N. Fountoulakis, M. Khosla, and K. Panagioutou. The multiple-orientability thresholds for random hypergraphs. In *Proc. 22nd ACM-SIAM SODA*, pages 1222–1236. SIAM, 2011.

32 N. Fountoulakis and K. Panagiotou. Orientability of random hypergraphs and the power of multiple choices. In *Proc. 37th ICALP, Part I*, LNCS 6198, pages 348–359. Springer, 2010.

33 A. M. Frieze and P. Melsted. Maximum matchings in random bipartite graphs and the space utilization of cuckoo hashtables. CoRR, arXiv:0910.5535, 2009.

34 P. Gao and N. C. Wormald. Load balancing and orientability thresholds for random hypergraphs. In *Proc. 42nd ACM STOC*, pages 97–104. ACM, 2010.

35 A. Kirsch, M. Mitzenmacher, and U. Wieder. More robust hashing: Cuckoo hashing with a stash. In *Proc. 16th ESA*, LNCS 5193, pages 611–622. Springer, 2008.

36 T. Q. Klassen and P. Woelfel. Independence of tabulation-based hash classes. CoRR, arXiv:1112.3323, 2011.

37 D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching.* Addison-Wesley, 1973.

38 E. Lehman and R. Panigrahy. 3.5-way cuckoo hashing for the price of 2-and-a-bit. In *Proc. 17th ESA*, LNCS 5757, pages 671–681. Springer, Springer, 2009.
39 M. Mitzenmacher and S. P. Vadhan. Why simple hash functions work: Exploiting the entropy in a data stream. In *Proc. 19th ACM-SIAM SODA*, pages 746–755, 2008.
40 A. Pagh and R. Pagh. Uniform hashing in constant time and optimal space. *SIAM J. Comput.*, 38(1):85–96, 2008.
41 A. Pagh, R. Pagh, and M. Ružić. Linear probing with constant independence. *SIAM J. Comput.*, 39(3):1107–1120, 2009.
42 R. Pagh and F. F. Rodler. Cuckoo hashing. *J. Algorithms*, 51(2):122–144, 2004.
43 M. Pătraşcu and M. Thorup. On the k-independence required by linear probing and minwise independence. In *Proc. 37th ICALP, Part I*, LNCS 6198, pages 715–726, 2010.
44 M. Pătraşcu and M. Thorup. The power of simple tabulation hashing. In *Proc. 43rd ACM STOC*, pages 1–10, 2011.
45 E. Porat and B. Shalem. Another one flew over the cuckoo's nest. CoRR, arXiv:1104.5400, 2011.
46 A. Siegel. On universal classes of fast high performance hash functions, their time-space tradeoff, and their applications (extended abstract). In *30th IEEE FOCS*, pages 20–25, 1989.
47 A. Siegel. On universal classes of extremely random constant-time hash functions. *SIAM J. Comput.*, 33(3):505–543, 2004.
48 M. Thorup. Even strongly universal hashing is pretty fast. In *Proc. 11th ACM-SIAM SODA*, pages 496–497, 2000.
49 M. Thorup. String hashing for linear probing. In *Proc. 20th ACM-SIAM SODA*, pages 655–664, 2009.
50 M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proc. 15th ACM-SIAM SODA*, pages 615–624, 2004.
51 M. Thorup and Y. Zhang. Tabulation based 5-universal hashing and linear probing. In *Proc. 12th ALENEX 2010*, pages 62–76. SIAM, 2010.
52 B. Vöcking. How asymmetry helps load balancing. *J. ACM*, 50(4):568–589, 2003.
53 M. N. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.
54 P. Woelfel. Asymmetric balanced allocation with simple hash functions. In *Proc. 17th ACM-SIAM SODA*, pages 424–433, 2006.