On Randomness in Hash Functions
Martin Dietzfelbinger

To cite this version:

HAL Id: hal-00678157
https://hal.archives-ouvertes.fr/hal-00678157
Submitted on 3 Feb 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
On Randomness in Hash Functions

Martin Dietzfelbinger

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
On Randomness in Hash Functions

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


28 On Randomness in Hash Functions