Perfect hash function

A **perfect hash function** for a set S is a hash function that maps distinct elements in S to a set of integers, with no collisions. A perfect hash function has many of the same applications as other hash functions, but with the advantage that no collision resolution has to be implemented. In mathematical terms, it is a total injective function.

Properties and uses

A perfect hash function for a specific set S that can be evaluated in constant time, and with values in a small range, can be found by a randomized algorithm in a number of operations that is proportional to the size of S[1]. The minimal size of the description of a perfect hash function depends on the range of its function values: The smaller the range, the more space is required. Any perfect hash functions suitable for use with a hash table require at least a number of bits that is proportional to the size of S.

A perfect hash function with values in a limited range can be used for efficient lookup operations, by placing keys from S (or other associated values) in a table indexed by the output of the function. Using a perfect hash function is best in situations where there is a frequently queried large set, S, which is seldom updated. Efficient solutions to performing updates are known as dynamic perfect hashing, but these methods are relatively complicated to implement. A simple alternative to perfect hashing, which also allows dynamic updates, is cuckoo hashing.

Minimal perfect hash function

A **minimal perfect hash function** is a perfect hash function that maps n keys to n consecutive integers—usually [0..n−1] or [1..n]. A more formal way of expressing this is: Let j and k be elements of some finite set **K**. F is a minimal perfect hash function iff $F(j) = F(k)$ implies $j=k$ (injectivity) and there exists an integer a such that the range of F is $a..a+|\mathbf{K}|-1$. It has been proved that a general purpose minimal perfect hash scheme requires at least 1.44 bits/key.[2] However the smallest currently use around 2.5 bits/key.

A minimal perfect hash function F is **order preserving** [3] if keys are given in some order $a_1, a_2, ...$, and for any keys a_j and a_k, $j<k$ implies $F(a_j)<F(a_k)$. Order-preserving minimal perfect hash functions require necessarily $\Omega(n \log n)$ bits to be represented.

A minimal perfect hash function F is **monotone** if it preserves the lexicographical order of the keys. Monotone minimal perfect hash functions can be represented in very little space.

References

[1] Fredman, M. L., Komlós, J., and Szemerédi, E. 1984. Storing a Sparse Table with 0(1) Worst Case Access Time. J. ACM 31, 3 (Jun. 1984), 538-544 http://portal.acm.org/citation.cfm?id=1884#

[2] Djamal Belazzougui, Fabiano C. Botelho, Martin Dietzfelbinger (2009) (PDF). *Hash, displace, and compress* (http://cmph.sourceforge.net/papers/esa09.pdf). Springer Berlin / Heidelberg. . Retrieved 2011-08-11.

[3] Fox A., Chen Q., Amjad M Daoud, Heath L. Order preserving minimal perfect hash functions and information retrieval (http://dl.acm.org/citation.cfm?id=96749.98233) SIGIR '90 Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval, Pages 279 - 311,1990

Further reading

- Fox A., Heath L., Chen Q., Amjad M Daoud. "Practical Minimal Perfect Hash Functions for Large Databases" (http://dl.acm.org/citation.cfm?id=129623), Communications of the ACM (CACM), 35(1), Jan. 1992.
- Fox A., Chen Q., Amjad M Daoud, Heath L. "Order-preserving minimal perfect hash functions and information retrieval" (http://dl.acm.org/citation.cfm?id=98233), ACM Transactions on Information Systems (TOIS), 9(1), Jan 1991.
- Amjad M Daoud. *Perfect Hash Functions for Large Web Repositories*, The Seventh International Conference on Information Integration and Web Based Applications & Services (iiWAS2005) , Kuala Lumpur, Malaysia, Sept. 2005.
- Amjad M Daoud. *Augmented Order Preserving Minimal Perfect Hash Functions for Very Large Digital Libraries*, Proceedings of the 15th WSEAS International Conference on Communications, pp. 90-95, ISBN: 978-1-61804-018, Corfu Island, Greece, Jul. 14-17, 2011.
- Amjad M Daoud. "Efficient Data Structures for Information Retrieval" (http://dl.acm.org/citation.cfm?id=194112), PhD. Thesis, Virginia Polytechnic Institute & State University, Blacksburg, Virginia, 1993; scholar.lib.vt.edu: OCLC Number: 29179633. Published by: University Microfilms Int./UMI, 1994.
- Richard J. Cichelli. *Minimal Perfect Hash Functions Made Simple*, Communications of the ACM, Vol. 23, Number 1, January 1980.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 11.5: Perfect hashing, pp. 245–249.
- Fabiano C. Botelho, Rasmus Pagh and Nivio Ziviani. "Perfect Hashing for Data Management Applications" (http://arxiv.org/pdf/cs/0702159).
- Fabiano C. Botelho and Nivio Ziviani. "External perfect hashing for very large key sets" (http://homepages.dcc.ufmg.br/~nivio/papers/cikm07.pdf). 16th ACM Conference on Information and Knowledge Management (CIKM07), Lisbon, Portugal, November 2007.
- Djamal Belazzougui, Paolo Boldi, Rasmus Pagh, and Sebastiano Vigna. "Monotone minimal perfect hashing: Searching a sorted table with O(1) accesses" (http://vigna.dsi.unimi.it/ftp/papers/MonotoneMinimalPerfectHashing.pdf). In Proceedings of the 20th Annual ACM-SIAM Symposium On Discrete Mathematics (SODA), New York, 2009. ACM Press.
- Djamal Belazzougui, Paolo Boldi, Rasmus Pagh, and Sebastiano Vigna. "Theory and practise of monotone minimal perfect hashing" (http://www.siam.org/proceedings/alenex/2009/alx09_013_belazzouguid.pdf). In Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX). SIAM, 2009.
- Douglas C. Schmidt, GPERF: A Perfect Hash Function Generator (http://www.cs.wustl.edu/~schmidt/PDF/gperf.pdf), C++ Report, SIGS, Vol. 10, No. 10, November/December, 1998.

External links

- Minimal Perfect Hashing Resources (http://iswsa.acm.org/mphf/tutorial.html) by Amjad M Daoud
- Minimal Perfect Hashing (http://burtleburtle.net/bob/hash/perfect.html) by Bob Jenkins
- gperf (http://www.gnu.org/software/gperf/) is a Open Source C and C++ perfect hash generator
- cmph (http://cmph.sourceforge.net/index.html) is Open Source implementing many perfect hashing methods
- Sux4J (http://sux4j.dsi.unimi.it/) is Open Source implementing perfect hashing, including monotone minimal perfect hashing in Java
- MPHSharp (http://www.dupuis.me/node/9) is Open Source implementing many perfect hashing methods in C#

Article Sources and Contributors

Perfect hash function *Source*: http://en.wikipedia.org/w/index.php?oldid=542205838 *Contributors*: 4hodmt, Arka sett, Bbb23, Burschik, CesarB's unpriviledged account, Cimon Avaro, Cobi, Daoudamjad, Dcoetzee, Drkarger, Dtrebbien, Długosz, E David Moyer, Fredrik, G121, Gajeam, Giftlite, Glrx, Headbomb, JMCorey, Johndburger, LOL, Maysak, Mcichelli, MegaHasher, Mudd1, Neilc, Otus, Pagh, Ruud Koot, Salrizvy, Spl, Srchvrs, SteveT84, Voomoo, Wikilolo, 40 anonymous edits

License