Weekplan: Hashing

Philip Bille

References and Reading

[1] Notes on universal hashing, Peter Bro Miltersen.

[2] Notes on string matching, Jeff Erickson.

[3] Universal Classes of Hash Functions, J. Carter and M. Wegman, J. Comp. Sys. Sci., 1977.

[4] Storing a Sparse Table with O(1) Worst Case Access Time, M. Fredman, J. Komlos and E. Szemeredi, J. ACM., 1984.

[5] Efficient randomized pattern-matching algorithms, R. Karp, M. O. Rabin, IBM J. Res. Dev, 1987.

[6] Notes on Discrete Probability, Jeff Erickson.

We recommend reading [1] and [2] in detail. The research papers [3], [4], and [5] provide background on universal hashing, perfect hashing, and string hashing. The notes in [6] provide a concise refresh of basic discrete probability.

Exercises

1 [w] **Streaming Statistics** An IT-security friend of yours wants a high-speed algorithm to count the number of *distinct* incoming IP-addresses in his router to help detect denial-of-service attacks. Can you help him?

2 [w] **Dense Set Hashing** A set $S \subseteq U = \{0, \dots, u-1\}$ is called *dense* if $|S| = \Theta(u)$. Suggest a simple and efficient dictionary data structure for dense sets.

3 [w] **Multi-Set Hashing** A multi-set is a set M, where each element may occur multiple times. Design an efficient data structure supporting the following operations:

- add(x): Add an(other) occurrence of x to M.
- remove(x): Remove an occurrence of x from M. If x does not occur in M do nothing.
- report(x): Return the number of occurrences of x.

4 **Properties of Universal Hashing** Let $h \in H$ be a hash function from a universal family mapping $U = \{0, \dots, u-1\}$ to $M = \{0, \dots, m-1\}$. Solve the following exercises.

4.1 If h has no collision on U, how large must m be?

4.2 Suppose $m \geq u$. Is the identity function $f(x) = x$ a universal hash function?

4.3 A family G of hash functions mapping U to M is *family of pair-wise independent hash function* if for any $g \in G$,

$$\Pr(g(x) = \alpha \wedge g(y) = \beta) = 1/m^2 \qquad \forall x \neq y \in U, \quad \forall \alpha, \beta \in M.$$

Show that any family of pairwise independent hash functions is a family of universal hash functions.

5 Linear Space Hashing The chained hashing solution for the dynamic dictionary problem presented assume that $m = \Theta(n)$. Solve the following exercises.

5.1 What is the space and time of chained hashing without this assumption? State your answer in terms of n and m.

5.2 Suppose that we do not know n in advance (as in the exercise streaming statistics where we do not know how many distinct IP-address we will see). Give a solution that achieves $O(n)$ space and constant time without assuming $m = \Theta(n)$. *Hint:* Think dynamic arrays.

6 Graph Adjacency Let G be a graph with n vertices and m edges. We want to represent G efficiently and support the following operation.

- adjacent(v, w): Return true if nodes v are w are adjacent and false otherwise.

Solve the following exercises:

6.1 Analyse the space and query time in terms of n and m for the classic adjacency matrix and adjacency list representation.

6.2 Design a data structure that improves both the adjacency matrix and adjacency list.

7 Perfect Hashing Analysis Consider the 2-level FKS perfect hashing scheme. A friend suggest the following two "optimizations" to the data structure. What happens to the performance of the data structure for each of these?

7.1 Modify level 1 of the data structure to map U to an array of size $n\sqrt{n}$ instead of n to further decrease the probability of collisions.

7.2 Replace the universal hash function with a faster *near-universal hash function* on both levels. Near-universal hashing is the same as universal hashing except that $\leq 1/m$ guarantee on the probability is changed to $\leq 2/m$.

8 String Hashing and String Matching

8.1 Show how to compute a fingerprint of a string of length s in time $O(s)$.

8.2 Show the rolling property.

8.3 Given a string S of length n and a set of k strings $\mathscr{P} = P_1, P_2, \ldots, P_k$ all of length m, the *multi-string matching problem* is to decide if any of the strings in $\mathscr{P}$ occurs in S. Give a fast algorithm for this problem.

8.4 Let S, T, R be three strings such that $S = T \odot R$, where $\odot$ denotes concatenation. Show that given the fingerprint of any two strings, we can efficiently compute the third's fingerprint.

9 Basic Probability Theory Refresh Bonus In case your knowledge of probability theory is rusty. Solve the following self-help exercises.

9.1 Prove linearity of expectation.

9.2 Prove that the expectation of the *indicator function* for $h(x) = h(y)$ (1 if $h(x) = h(y)$ and 0 otherwise) is equal to the probability that $h(x) = h(y)$.

9.3 Show that the expected number of trials to get a perfect hashing function using an array of size n^2 is ≤ 2.