Hashing: Chaining, Collision Analysis, and Universal Hashing

REVANT • 3 Jan 2026

Abstract

This article presents a theoretical analysis of hashing using chaining and universal hashing. All results are derived using probability theory and elementary number theory. The focus is on understanding both idealized probabilistic models and their implications for practical hash function design.

Prerequisites

This article assumes that the reader has basic familiarity with the concept of hashing as used in data structures and algorithms.

In particular, the reader is expected to know:

- What a hash table is and how it is used to implement dictionary operations such as insertion, search, and deletion.
- The role of a hash function in mapping keys to table indices.
- The concept of collisions, where multiple keys map to the same table slot.
- Common collision resolution strategies, including chaining and open addressing.

No prior knowledge of probabilistic analysis or advanced hashing theory is assumed; all such concepts are developed within the article.

Model, notation, and assumptions

We begin by formally specifying the hashing model and the notation used throughout the analysis.

Let:

- U be the universe of keys.
- $K = \{k_1, k_2, \ldots, k_n\} \subseteq U$ be the set of stored keys.

- m be the number of hash table slots.
- $h : U \to \{0, 1, \ldots, m-1\}$ be a hash function.

Load factor

The *load factor* of the hash table is defined as

$$\alpha = \frac{n}{m}.$$

Intuitively, the load factor measures the average occupancy of the table and plays a central role in the performance analysis of hashing schemes.

It represents the expected number of keys per slot.

Simple Uniform Hashing (SUH) assumption

To enable tractable probabilistic analysis, we adopt the Simple Uniform Hashing (SUH) assumption.

For every key $k \in U$,

$$\Pr(h(k) = j) = \frac{1}{m} \quad \forall j \in \{0, 1, \ldots, m-1\},$$

and hash values of distinct keys are mutually independent.

This is an *assumption*, not a property of real hash functions. All expectations derived below are conditional on this assumption and represent idealized average-case behavior.

Expected number of keys in a slot (chaining)

We first analyze the expected distribution of keys across slots when chaining is used to resolve collisions.

Fix a slot j.

Define indicator random variables:

$$X_i = \begin{cases} 1 & \text{if } h(k_i) = j, \\ 0 & \text{otherwise.} \end{cases}$$

The number of keys stored in slot j is

$$n_j = \sum_{i=1}^{n} X_i. \tag{1}$$

Taking expectation and using linearity of expectation,

$$E[n_j] = E\left[\sum_{i=1}^{n} X_i\right] \tag{2}$$

$$= \sum_{i=1}^{n} E[X_i]. \tag{3}$$

Since X_i is an indicator variable,

$$E[X_i] = \Pr(h(k_i) = j) = \frac{1}{m}.$$

Substituting,

$$E[n_j] = \frac{n}{m} = \alpha.$$

Thus, under SUH, the expected chain length in every slot equals the load factor.

Collision probability between two distinct keys

We now quantify the likelihood of collisions between pairs of distinct keys, which will be used in the analysis of search cost.

Let $k_i \neq k_j$ be two distinct keys.

Define

$$X_{i,j} = \begin{cases} 1 & \text{if } h(k_i) = h(k_j), \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$E[X_{i,j}] = \Pr(h(k_i) = h(k_j)).$$

Using the law of total probability,

$$\Pr(h(k_i) = h(k_j)) = \sum_{r=0}^{m-1} \Pr(h(k_i) = r \wedge h(k_j) = r). \tag{4}$$

By independence under SUH,

$$\Pr(h(k_i) = r \wedge h(k_j) = r) = \frac{1}{m^2}.$$

Therefore,

$$\Pr(h(k_i) = h(k_j)) = \frac{1}{m}.$$

Expected cost of successful search (chaining)

Using the collision analysis above, we now derive the expected cost of a successful search in a hash table with chaining.

Assume that each key is inserted at the *head* of its chain.

Keys are inserted in the order

$$k_1, k_2, \ldots, k_n.$$

When searching for k_i, only keys inserted *after* k_i may appear before it in the chain.

Thus, the cost of searching for k_i is

$$C_i = 1 + \sum_{j=i+1}^{n} X_{i,j}.$$

Taking expectation,

$$E[C_i] = 1 + \frac{n-i}{m}.$$

Averaging over all keys,

$$E[C] = 1 + \frac{n-1}{2m}.$$

Using $\alpha = n/m$,

$$E[C] \in \Theta(1+\alpha).$$

Expected cost of unsuccessful search (chaining)

A complete analysis of chaining must also consider the cost of an *unsuccessful* search.

An unsuccessful search for a key $k \notin K$ examines all keys stored in the slot $h(k)$. Under the Simple Uniform Hashing assumption, the expected number of keys in any slot equals the load factor α.

Therefore, the expected cost of an unsuccessful search is

$$E[C_{\text{unsuccessful}}] = \alpha.$$

Including the constant-time hash computation, unsuccessful searches take expected time

$$\Theta(1 + \alpha),$$

which serves as a baseline comparison for the successful search bound.

Division method and structural failure

The preceding analysis relies critically on the Simple Uniform Hashing assumption, which models hash functions as ideal random mappings. In practice, however, hash functions are deterministic and may violate this assumption.

We now examine the division method as a concrete example illustrating how poor design choices can lead to systematic clustering.

Consider the division method

$$h(k) = k \bmod m.$$

If $m = 2^p$, then

$$k \bmod 2^p$$

depends only on the lowest p bits of k.

Hence, any regularity in lower-order bits of keys produces systematic clustering.

More generally, for keys of the form

$$k = a + td,$$

the sequence

$$(a + td) \bmod m$$

covers only

$$\frac{m}{\gcd(d, m)}$$

distinct slots.

Choosing m prime ensures $\gcd(d, m) = 1$ for all $1 \leq d < m$, preventing such collapse.

Universal hashing

To overcome the limitations of deterministic hashing schemes, we introduce the concept of universal hashing.

Definition

A family H of hash functions is *universal* if

$$\forall k \neq \ell : \quad \Pr_{h \in H}[h(k) = h(\ell)] \leq \frac{1}{m}.$$

Universal hash family construction

Let p be a prime such that $p > \max U$.

Choose:

- $a \in \{1, 2, \ldots, p-1\}$,
- $b \in \{0, 1, \ldots, p-1\}$.

Define

$$h_{a,b}(k) = ((ak + b) \bmod p) \bmod m.$$

Proof of universality

Fix $k \neq \ell$.

Let

$$x = (ak + b) \bmod p, \quad y = (a\ell + b) \bmod p.$$

For fixed a, as b ranges uniformly, the pair (x, y) is uniformly distributed over $\{0, 1, \ldots, p-1\}^2$.

A collision occurs when

$$x \equiv y \pmod{m}.$$

For a fixed x, the number of y satisfying this congruence is at most $\lceil p/m \rceil$.

Thus,

$$\Pr[h(k) = h(\ell)] \leq \frac{\lceil p/m \rceil}{p} \tag{5}$$

$$\leq \frac{1}{m} + \frac{1}{p}. \tag{6}$$

Since $p > m$,

$$\Pr[h(k) = h(\ell)] \leq \frac{1}{m}.$$

Strictly speaking, the above bound yields

$$\Pr[h(k) = h(\ell)] \leq \frac{1}{m} + \frac{1}{p},$$

which is sometimes referred to as *almost universal hashing*. In standard treatments, p is chosen sufficiently large compared to m, making the additional $\frac{1}{p}$ term negligible. Under this convention, the family satisfies the universal hashing requirement for all practical and theoretical purposes.

Final conclusions

We summarize the main theoretical insights obtained in this analysis.

- The expected chain length equals the load factor.
- Successful search time under chaining is $\Theta(1 + \alpha)$.
- Unsuccessful search time under chaining is also $\Theta(1 + \alpha)$.
- Poor modulus choice causes deterministic clustering.
- Universal hashing guarantees bounded collision probability independent of input distribution.

Continuation: Open Addressing and Advanced Topics

This article has focused on hashing with chaining under idealized probabilistic assumptions and on the design of universal hash families. In **Part II** of this article, we will study *open-address hashing*, where all keys are stored directly within the hash table itself and collisions are resolved by systematic probing sequences.

The next part will develop both algorithmic techniques and their rigorous analysis, with particular emphasis on probabilistic bounds and worst-case guarantees.

Topics to be covered in Part II

Part II will cover the following topics in detail:

- Open addressing and probe sequences
- Linear probing
- Quadratic probing
- Double hashing
- Expected-time analysis of open-address hashing

- Longest-probe bound for hashing
- Slot-size bounds for chaining
- Perfect hashing
- Hashing and authentication

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, 3rd Edition, 2009.

[2] E. Demaine, J. Ku, and J. Solomon, *6.006: Introduction to Algorithms*, Massachusetts Institute of Technology, Spring 2020. Available via MIT OpenCourseWare.

[3] J. Kleinberg and É. Tardos, *Algorithm Design*, Pearson, 2006.