

Comprehensive Study Guide: Hashing & Bloom Filters

edX Course - Stanford University

1 The Motivation for Hashing

To maintain a set S of keys drawn from a large universe U , we compare three fundamental data structures. The goal of hashing is to combine the speed of arrays with the space efficiency of linked lists.

Approach	Space	Time (Lookup)	Trade-off
Direct Addressing (Array)	$\Theta(U)$	$O(1)$ Worst Case	Fast but Impossible Space. If U is large (e.g., IPv6), memory explodes.
Linked List	$\Theta(S)$	$\Theta(S)$ Worst Case	Space Efficient but Slow. Must scan the entire list to find items.
Hash Table	$\Theta(S)$	$O(1)$ Expected	The Sweet Spot. Maps huge U to small table size n .

Table 1: Comparison of Data Structures

2 Birthday Paradox

Consider n people with random birthdays (assuming 365 days in a year, equally likely, and independent). We want to find the smallest integer n such that the probability of at least two people sharing a birthday is at least 50%:

$$P(\text{at least one collision}) \geq 0.5 \tag{1}$$

2.1 The Complement Strategy

Calculating collisions directly is complex due to the number of possible pairs. It is easier to calculate the complement event: **all birthdays are unique**.

$$\begin{aligned} P(\text{shared}) &= 1 - P(\text{all unique}) \\ 1 - P(\text{unique}) &\geq 0.5 \\ P(\text{unique}) &\leq 0.5 \end{aligned}$$

2.2 Exact Probability Formula

We assign birthdays to n people one by one, ensuring no collisions:

- Person 1: 365/365 (Any day is fine)
- Person 2: 364/365 (Must avoid Person 1)
- Person k : $(365 - (k - 1))/365$ (Must avoid previous $k - 1$ people)

The total probability is the product of these independent choices:

$$P(\text{unique}) = 1 \times \frac{364}{365} \times \frac{363}{365} \times \cdots \times \frac{365 - (n - 1)}{365} \quad (2)$$

This can be written compactly using product notation:

$$P(\text{unique}) = \prod_{k=0}^{n-1} \left(\frac{365 - k}{365} \right) = \prod_{k=0}^{n-1} \left(1 - \frac{k}{365} \right) \quad (3)$$

2.3 Approximation using Taylor Series

To solve for n , we convert the product into a sum by taking the natural logarithm (\ln) of both sides:

$$\ln(P(\text{unique})) = \sum_{k=0}^{n-1} \ln \left(1 - \frac{k}{365} \right) \quad (4)$$

We use the Taylor Series expansion for $\ln(1 - x)$ near $x = 0$:

$$\ln(1 - x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \cdots$$

For small x (where $x = k/365$), we approximate $\ln(1 - x) \approx -x$.

Substituting this into our sum:

$$\begin{aligned} \ln(P(\text{unique})) &\approx \sum_{k=0}^{n-1} \left(-\frac{k}{365} \right) \\ &= -\frac{1}{365} \sum_{k=0}^{n-1} k \end{aligned}$$

Using the arithmetic series sum formula $\sum_{k=0}^{n-1} k = \frac{n(n-1)}{2}$, we get:

$$\ln(P(\text{unique})) \approx -\frac{1}{365} \cdot \frac{n(n-1)}{2} = -\frac{n(n-1)}{730} \quad (5)$$

Exponentiating both sides gives the approximate probability formula:

$$P(\text{unique}) \approx e^{-\frac{n(n-1)}{730}} \quad (6)$$

2.4 Solving for n

We require $P(\text{unique}) \leq 0.5$.

$$\begin{aligned} e^{-\frac{n(n-1)}{730}} &\leq 0.5 \\ -\frac{n(n-1)}{730} &\leq \ln(0.5) \\ -\frac{n(n-1)}{730} &\leq -0.693 \quad (\text{since } \ln(2) \approx 0.693) \\ n(n-1) &\geq 0.693 \times 730 \\ n(n-1) &\geq 505.89 \end{aligned}$$

Approximating $n(n-1) \approx n^2$:

$$\begin{aligned}n^2 &\approx 506 \\n &\approx \sqrt{506} \approx 22.49\end{aligned}$$

Since n must be an integer, we round up:

$$\boxed{n = 23} \tag{7}$$

3 Modular Arithmetic & Inverses

Understanding modular inverses is a prerequisite for the Universal Hashing proof.

3.1 Modular Inverses

An integer u is said to be **invertible modulo n** if there exists an integer v (often written as u^{-1}) such that:

$$u \cdot v \equiv 1 \pmod{n} \tag{8}$$

- **Existence Condition:** An inverse exists **if and only if** $\gcd(u, n) = 1$.
- **Prime Modulus Property:** If $n = p$ (a prime number), then every non-zero integer $u \in \{1, \dots, p-1\}$ is invertible. This is because a prime number shares no factors with numbers smaller than itself.

3.2 Computing Inverses: Extended Euclidean Algorithm (EEA)

To find the inverse of $u \pmod{n}$, we solve Bézout's Identity: $s \cdot u + t \cdot n = 1$. Reducing this modulo n gives $s \cdot u \equiv 1 \pmod{n}$.

Example: Find $11^{-1} \pmod{26}$

Step 1: Forward Pass (Euclidean Division)

$$\begin{aligned}26 &= 2 \cdot 11 + 4 \\11 &= 2 \cdot 4 + 3 \\4 &= 1 \cdot 3 + 1 \quad \leftarrow \text{GCD is 1, so inverse exists.}\end{aligned}$$

Step 2: Backward Pass (Substitution) Express 1 as a linear combination of 11 and 26.

$$\begin{aligned}1 &= 4 - 1 \cdot 3 \\&= 4 - 1 \cdot (11 - 2 \cdot 4) \quad (\text{Substitute } 3 = 11 - 2 \cdot 4) \\&= 3 \cdot 4 - 1 \cdot 11 \\&= 3 \cdot (26 - 2 \cdot 11) - 1 \cdot 11 \quad (\text{Substitute } 4 = 26 - 2 \cdot 11) \\&= 3 \cdot 26 - 6 \cdot 11 - 1 \cdot 11 \\&= 3 \cdot 26 - 7 \cdot 11\end{aligned}$$

Step 3: Result Reducing modulo 26, the term $3 \cdot 26$ becomes 0:

$$\begin{aligned}-7 \cdot 11 &\equiv 1 \pmod{26} \\-7 &\equiv 19 \pmod{26}\end{aligned}$$

Conclusion: $11^{-1} \equiv 19 \pmod{26}$.

4 Universal Hashing

4.1 Definition

A family of hash functions \mathcal{H} mapping $U \rightarrow \{0, \dots, n-1\}$ is **universal** if for any two distinct keys $x \neq y$:

$$\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{n} \quad (9)$$

Note: Per-key uniformity (probability $1/n$ of hitting a specific bucket) is necessary but **not sufficient**. We specifically require the **pairwise collision probability** to be low.

4.2 Proof: Hashing IP Addresses

The Setup:

- **Keys:** IP addresses decomposed into 4 parts: $x = (x_1, x_2, x_3, x_4)$ where $x_i \in \{0, \dots, n-1\}$.
- **Buckets:** A prime number n .
- **Hash Family:** Defined by a random vector $a = (a_1, a_2, a_3, a_4)$ where each a_i is uniform in $\{0, \dots, n-1\}$.
- **Function:** $h_a(x) = \sum_{i=1}^4 a_i x_i \pmod{n}$.

The Proof: We wish to calculate the probability that distinct keys x and y collide.

1. The Collision Equation

$$\begin{aligned} h_a(x) &= h_a(y) \\ \sum_{i=1}^4 a_i x_i &\equiv \sum_{i=1}^4 a_i y_i \pmod{n} \\ \sum_{i=1}^4 a_i (x_i - y_i) &\equiv 0 \pmod{n} \end{aligned}$$

2. Isolate a Non-Zero Difference Since $x \neq y$, they must differ in at least one position. Without loss of generality, assume $x_4 \neq y_4$. We isolate the term involving a_4 :

$$\begin{aligned} a_4(x_4 - y_4) + \sum_{i=1}^3 a_i(x_i - y_i) &\equiv 0 \pmod{n} \\ a_4(x_4 - y_4) &\equiv -\sum_{i=1}^3 a_i(x_i - y_i) \pmod{n} \end{aligned}$$

Let the right-hand side be $C = \sum_{i=1}^3 a_i(y_i - x_i)$. The equation becomes:

$$a_4(x_4 - y_4) \equiv C \pmod{n} \quad (10)$$

3. Principle of Deferred Decisions Imagine we pick the random coefficients a_1, a_2, a_3 **first**. This fixes the value of C . Now, we pick a_4 .

Since n is prime and $(x_4 - y_4) \not\equiv 0 \pmod{n}$, the term $(x_4 - y_4)$ has a unique modular inverse. We multiply both sides by this inverse:

$$a_4 \equiv C \cdot (x_4 - y_4)^{-1} \pmod{n} \quad (11)$$

4. Conclusion The equation above yields exactly **one** valid solution for a_4 . Since a_4 is chosen uniformly from n possibilities $\{0, \dots, n-1\}$, the probability of picking exactly this one solution is:

$$\Pr[\text{Collision}] = \frac{1}{n} \quad (12)$$

Thus, the family is Universal.

5 Bloom Filters

A Bloom Filter is a space-efficient probabilistic data structure for set membership. It yields no false negatives, but possible false positives ($h_i(x)$'s already set to 1 by other insertions.).

5.1 Mathematical Derivation of False Positive Rate

Parameters:

- n : Number of items inserted.
- m : Size of the bit array.
- k : Number of hash functions.

Step 1: Probability a bit is 0 When inserting 1 element with 1 hash function, the probability a specific bit is *not* set (remains 0) is:

$$1 - \frac{1}{m}$$

After inserting n elements using k hash functions, we perform kn total writes. The probability the bit is still 0 is:

$$\left(1 - \frac{1}{m}\right)^{kn}$$

Step 2: Taylor Series Approximation Recall the Taylor series expansion for e^{-x} for small x : $e^{-x} \approx 1 - x$. Conversely, for large m :

$$1 - \frac{1}{m} \approx e^{-1/m}$$

Substituting this into our equation:

$$\Pr[\text{bit is 0}] \approx \left(e^{-1/m}\right)^{kn} = e^{-kn/m}$$

Step 3: Probability a bit is 1

$$\Pr[\text{bit is 1}] = 1 - \Pr[\text{bit is 0}] = 1 - e^{-kn/m}$$

Step 4: False Positive Probability (ϵ) A false positive occurs when we query an item that is **not** in the set, but all k hash functions map to bits that happen to be 1. Assuming independence:

$$\epsilon = \left(1 - e^{-kn/m}\right)^k \tag{13}$$

6 Summary Checklist

- **Why Prime Buckets?** Prime n ensures that every non-zero difference $(x_i - y_i)$ has a modular inverse. This guarantees the collision equation has a unique solution.
- **Invertible Slopes:** In affine hashing $h(x) = (ax + b) \pmod{m}$, if a is not coprime to m ($\gcd(a, m) > 1$), multiple keys will collapse to the same bucket systematically. We need $\gcd(a, m) = 1$.
- **Universal Bound:** The gold standard is $\Pr[\text{Collision}] \leq 1/n$.

7 Question 1