

Probability and Computing – Classic Hash Tables

Stefan Walzer | WS 2025/2026



1. Conceptions: What is a Hash Function?

- Hashing in the Wild
- What should a Theorist do?

2. Use Case 1: Hash Table with Chaining

- Using SUHA
- Using Universal Hashing

3. Use Case 2: Linear Probing

- Using SUHA
- Using Universal Hashing

4. Recent Developments

5. Conclusion

Hash Table with Chaining

e.g. `std::unordered_set`, `java.util.HashMap`

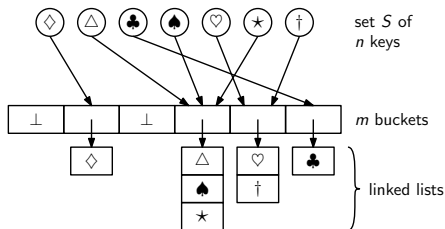
Terminology

D : Universe (or domain) of keys
(strings, integers, game states in chess)

$S \subseteq D$: set of n keys (possibly with associated data)

$h : D \rightarrow R$: hash function, range usually $R = [m]$

$\alpha = \frac{n}{m}$: load factor, $\alpha = \mathcal{O}(1)$



Goal

Operations in time t with $\mathbb{E}[t] = \mathcal{O}(1)$.
Randomness comes from the hash function.

Ideal Hash Functions

Every function from D to R is equally likely to be h .

Ideal Hash Functions are Impractical

Naive Idea

- Let R^D denote all functions from D to R . We pick $h \sim \mathcal{U}(R^D)$.
- There are $|R|$ options for the hash of each $x \in D$
- Hence: $|R^D| = |R|^{|D|}$

$x \in D$	x_1	x_2	x_3	$\dots$	$x_{ D }$
$h(x) \in R$	?	?	?	$\dots$	?

Why $h \sim \mathcal{U}(R^D)$ is desirable

- $h \sim \mathcal{U}(R^D) \Leftrightarrow \forall x_1, \dots, x_n \in D : h(x_1), h(x_2), \dots, h(x_n)$ are *independent* and uniformly random in R .
 $\hookrightarrow$ independence is very useful in an analysis
- In particular: $\forall x_1, \dots, x_n \in D, \forall i_1, \dots, i_n : \Pr_{h \sim \mathcal{U}(R^D)}[h(x_1) = i_1 \wedge \dots \wedge h(x_n) = i_n] = |R|^{-n}$.

Why $h \sim \mathcal{U}(R^D)$ is unwieldy

$\log_2(|R|^{|D|}) = |D| \cdot \log_2(|R|)$ bits to store $h \sim \mathcal{U}(R^D)$ $\rightsquigarrow$ for $D = \{0, 1\}^{64}$: more than 2^{64} bits.

1. Conceptions: What is a Hash Function?

- Hashing in the Wild
- What should a Theorist do?

2. Use Case 1: Hash Table with Chaining

- Using SUHA
- Using Universal Hashing

3. Use Case 2: Linear Probing

- Using SUHA
- Using Universal Hashing

4. Recent Developments

5. Conclusion

Conceptions: What is a Hash Function?

●○○○○○

Use Case 1: Hash Table with Chaining

○○○○○○○○

Use Case 2: Linear Probing

○○○○○○○○○○○○○○

Recent Developments

○○○○○

Conclusion

○○○

References

What is a Hash Function?

(it depends on who you ask)

Cryptographic Hash Function

A **collision resistant** function such as $h = \text{sha256sum}$

```
$ sha256sum myfile.txt  
018a7eae8a...3e79043e21ab4  myfile.txt
```

Range $R = \{0, 1\}^{256}$. It is hard to find x, y with $h(x) = h(y)$.

↪ Files with equal hashes are likely the same.

Cryptographic Pseudorandom Function

A function $f : \text{Seeds} \times D \rightarrow R$ where $\log_2 |\text{Seeds}|$ is small and no efficient algorithm can distinguish

- $f(s, \cdot)$ for $s \sim \mathcal{U}(\text{Seeds})$ and
- $h(\cdot)$ for $h \sim \mathcal{U}(R^D)$,

except with negligible probability.

Conceptions: What is a Hash Function?
●●○○○○

Use Case 1: Hash Table with Chaining
○○○○○○○○

Use Case 2: Linear Probing
○○○○○○○○○○○○

Recent Developments
○○○○

Conclusion
○○○

References

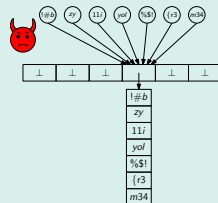
Hash Function in Algorithm Engineering

- typically small range $|R| = \mathcal{O}(n)$
↪ cannot be collision resistant
- should **behave like** $h \sim \mathcal{U}(R^D)$ in my application
- should be **fast** to evaluate
- adversarial settings rarely considered, although:



HashDoS is a thing.

However: Hash function and hash values need not be public.



High-Speed Hashing in Practical Data Structures

Black Magic, do not touch!

MurmurHash

Bitshifts, Magic Constants, ...

```
uint32_t murmur3_32(const uint8_t* key,
                    size_t len, uint32_t seed) {
    uint32_t h = seed;
    uint32_t k;
    for (size_t i = len >> 2; i--;) {
        memcpy(&k, key, sizeof(uint32_t));
        key += sizeof(uint32_t);
        h ^= murmur_32_scramble(k);
        h = (h << 13) | (h >> 19);
        h = h * 5 + 0xe6546b64;
    }
    [...]
    return h;
}

static inline uint32_t murmur_32_scramble(uint32_t k) {
    k *= 0xcc9e2d51;
    k = (k << 15) | (k >> 17);
    k *= 0x1b873593;
    return k;
}
```

Usage

For $R = [m]$, pick seed $\sim \mathcal{U}(\{0, 1\}^{32})$ and use

$$h(x) = \text{murmur3_32}(x, \text{seed}) \bmod m \quad // \text{mod slow in practice}$$

$$h'(x) = \lfloor \text{murmur3_32}(x, \text{seed}) \cdot m/2^{32} \rfloor \quad // \text{"fast range"}$$

(see <https://github.com/lemire/fastrange>)

Does h behave like a random function?

- **YES**, with respect to many statistical tests.

see <https://github.com/aappleby/smhasher>

- **NO**, HashDoS attacks are known.

see <https://en.wikipedia.org/wiki/MurmurHash#Vulnerabilities>

- **MAYBE**, for your favourite application.

Conceptions: What is a Hash Function?

○○●○○○

Use Case 1: Hash Table with Chaining

○○○○○○○○

Use Case 2: Linear Probing

○○○○○○○○○○○○

Recent Developments

○○○○○

Conclusion

○○○

References

1. Conceptions: What is a Hash Function?

- Hashing in the Wild
- What should a Theorist do?

2. Use Case 1: Hash Table with Chaining

- Using SUHA
- Using Universal Hashing

3. Use Case 2: Linear Probing

- Using SUHA
- Using Universal Hashing

4. Recent Developments

5. Conclusion

What should a Theorist do?

Approach 1: Ignore the Problem

Simple Uniform Hashing Assumption (SUHA)

- We have access to $h \sim \mathcal{U}(R^D)$ for any R and D .
- h takes $\mathcal{O}(1)$ time to evaluate.
- h takes no space to store.

How to Analyse your Algorithm

- 1 *Assume* SUHA holds.
- 2 *Analyse* algorithm under SUHA.
- 3 *Hope* that algorithm still works with real hash functions.

SUHA is “wrong” but adequate

- *Modelling* assumption.
↔ like e.g. ideal gas law in physics
- Excellent track record in non-adversarial settings.

What should a Theorist do?

Approach 2: Bring your own Hash Functions

Analyse Algorithm using Universal Hashing

- 1 Define family $\mathcal{H} \subseteq R^D$ of hash functions with $\log(|\mathcal{H}|)$ not too large.
 $\hookrightarrow$ sampling and storing $h \in \mathcal{H}$ is cheap
- 2 Proof that algorithm with $h \sim \mathcal{U}(\mathcal{H})$ has good expected behaviour.

Remarks

- Mathematical structure of $\mathcal{H}$ must be amenable to analysis.
- *Rigorously* covers non-adversarial settings.
- Proofs often difficult.
 $\hookrightarrow$ Wider theory practice gap than with SUHA.

What should a Theorist do?

Approach 3: Let the Cryptographers do the Work

How to Analyse your Algorithm using Cryptographic Assumptions

- 1 Analyse algorithm under *SUHA*.
- 2 Actually use *cryptographic pseudorandom function* f .
 - **Case 1:** Everything still works. Great! :-)
 - **Case 2:** Something fails.
 - ⇒ Your use case can tell the difference between f and true randomness.
 - ↪ The cryptographers said this is impossible. $\nexists$

Should we use cryptographic pseudorandom functions?

- **YES.** Algorithms become robust even in some adversarial settings.
 - ↪ e.g. Python, Haskell, Ruby, Rust use **SipHash** by default
 - <https://en.wikipedia.org/wiki/SipHash>
- **NO.** Too slow in high-performance settings.

Hash Function	MiB / sec
SipHash	944
Murmur3F	7623
xxHash64	12109

(source: <https://github.com/rurban/smhasher>)

1. Conceptions: What is a Hash Function?

- Hashing in the Wild
- What should a Theorist do?

2. Use Case 1: Hash Table with Chaining

- Using SUHA
- Using Universal Hashing

3. Use Case 2: Linear Probing

- Using SUHA
- Using Universal Hashing

4. Recent Developments

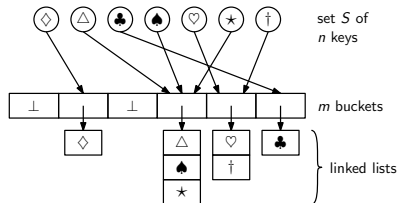
5. Conclusion

Search Time under Chaining

For $n, m \in \mathbb{N}$ and a family $\mathcal{H} \subseteq [m]^D$ of hash functions the *maximum expected search time* is at most

$$T_{\text{chaining}}(n, m, \mathcal{H}) = \max_{\substack{S \subseteq D \\ |S|=n}} \max_{x \in D} \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[1 + |\{y \in S \mid h(y) = h(x)\}| \right]$$

⚠ Key set is *worst case*. Only $h \in \mathcal{H}$ is random. Key set is fixed *before* h is chosen.



Theorem: Hash Table with Chaining under SUHA

If $\mathcal{H} = [m]^D$ then $T_{\text{chaining}}(n, m, \mathcal{H}) \leq 2 + \alpha = \mathcal{O}(1)$ if $\alpha \in \mathcal{O}(1)$.

Theorem: Hash Table with Chaining under SUHA

Let $\mathcal{H} = [m]^D$, $S \subseteq D$ with $|S| = n$ and $x \in D$ then

$$\mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[1 + |\{y \in S \mid h(y) = h(x)\}| \right] \leq 2 + \alpha$$

Proof.

$$\begin{aligned} & \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[1 + |\{y \in S \mid h(y) = h(x)\}| \right] \\ &= \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[1 + \sum_{y \in S} [h(y) = h(x)] \right] \\ &= 1 + \sum_{y \in S} \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} [h(y) = h(x)] \end{aligned}$$

$$\begin{aligned} &= 1 + \sum_{y \in S} \Pr_{h \sim \mathcal{U}(\mathcal{H})} [h(y) = h(x)] \\ &\leq 1 + 1 + \sum_{y \in S \setminus \{x\}} \Pr_{h \sim \mathcal{U}(\mathcal{H})} [h(y) = h(x)] \\ &= 2 + \sum_{y \in S \setminus \{x\}} \frac{1}{m} \leq 2 + \frac{n}{m} = 2 + \alpha. \quad \square \end{aligned}$$

1. Conceptions: What is a Hash Function?

- Hashing in the Wild
- What should a Theorist do?

2. Use Case 1: Hash Table with Chaining

- Using SUHA
- Using Universal Hashing

3. Use Case 2: Linear Probing

- Using SUHA
- Using Universal Hashing

4. Recent Developments

5. Conclusion

A Universal Hash Family

Definition: c -universal hash family

A class $\mathcal{H} \subseteq [m]^D$ is called c -universal if: $\forall x \neq y \in D: \Pr_{h \sim \mathcal{U}(\mathcal{H})} [h(x) = h(y)] \leq \frac{c}{m}.$

Note: $\mathcal{H} = [m]^D$ is 1-universal.

Reminder (?): Finite Fields

Let $\mathbb{F}_p = \{0, \dots, p-1\}$ for a prime number p . Then $(\mathbb{F}_p, \times, \oplus)$ is a field where

$$a \times b := (a \cdot b) \bmod p \quad \text{and} \quad a \oplus b := (a + b) \bmod p.$$

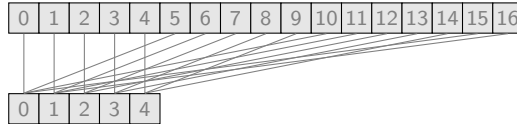
In particular $(\mathbb{F}_p^* := \mathbb{F}_p \setminus \{0\}, \times)$ is a group.

The class of Linear Hash Functions

Assume $D \subseteq \mathbb{F}_p$ for prime p . Then the following class is 1-universal:

$$\mathcal{H}_{p,m}^{\text{lin}} := \{x \mapsto ((a \times x) \oplus b) \bmod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}.$$

Can't we use a simpler class?



$\text{mod } m$

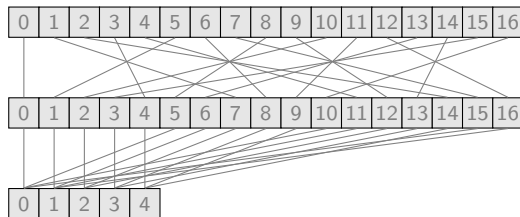
$$p = 17$$

$$m = 5$$

What about just $x \mapsto x \bmod m$?

Nothing is random. We have $h(0) = h(5)$ but this should hold only with probability $1/5$.

Can't we use a simpler class?


 $\times a$
 $\text{mod } m$

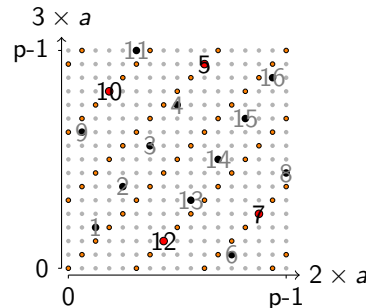
$$p = 17$$

$$m = 5$$

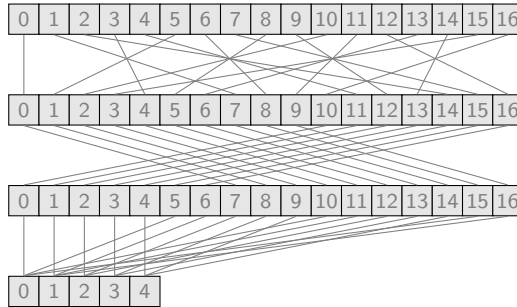
$$a = 7 \text{ // } \sim \mathcal{U}(\mathbb{F}_p^*)$$

What about just $\{x \mapsto (a \times x) \bmod m \mid a \in \mathbb{F}_p^*\}$?

- Example: Do 2 and 3 collide? Picture $\{(a \times 2, a \times 3) \mid a \in \mathbb{F}_p^*\}$.
 $\Pr_{a \sim \mathcal{U}(\mathbb{F}_p^*)}[h(2) = h(3)] = \Pr[a \in \{5, 7, 10, 12\}] = \frac{4}{16} > \frac{3}{15} = \frac{1}{5}$.
 $\Rightarrow$ not 1-universal (but reportedly 2-universal)
- Also note: $h(0) = 0$ is not random.



Can't we use a simpler class?


 $\times a$
 $\oplus b$
 $\text{mod } m$

$$p = 17$$

$$m = 5$$

$$a = 7 \quad // \quad \sim \mathcal{U}(\mathbb{F}_p^*)$$

$$b = 7 \quad // \quad \sim \mathcal{U}(\mathbb{F}_p)$$

Back to $\{x \mapsto ((a \times x) \oplus b) \bmod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}$

Mathematically “cleaner”. Proof of 1-universality on next slide.

Remark: “... mod m ” can be replaced with “ $\lfloor \dots \cdot \frac{m}{p} \rfloor$ ” with no downsides.

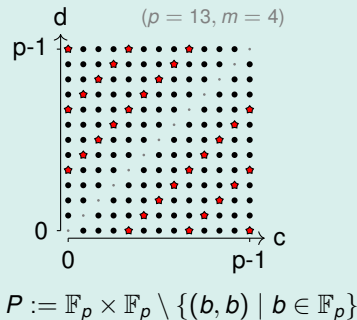
Proof that $\mathcal{H}_{p,m}^{\text{lin}} := \{x \mapsto ((a \times x) \oplus b) \bmod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}$ is 1-universal.

Let $x \neq y \in \mathbb{F}_p$. (To show: $\Pr_{h \sim \mathcal{H}_{p,m}^{\text{lin}}} [h(x) = h(y)] \leq 1/m$.)

- Define
$$\begin{aligned} c &= (a \times x) \oplus b \\ d &= (a \times y) \oplus b \end{aligned} \Leftrightarrow \begin{pmatrix} c \\ d \end{pmatrix} = \underbrace{\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix}}_{\text{regular!}} \begin{pmatrix} a \\ b \end{pmatrix}.$$

- The mapping $(a, b) \mapsto (c, d)$ is a bijection (for every $x \neq y$) from $\mathbb{F}_p^* \times \mathbb{F}_p \rightarrow P$. $\parallel (a, b) \sim \mathcal{U}(\mathbb{F}_p^* \times \mathbb{F}_p) \Rightarrow (c, d) \sim \mathcal{U}(P)$

- Define **bad set** $B := \{(c, d) \in P \mid c \bmod m = d \bmod m\}$.
 $\hookrightarrow$ from picture: $\frac{|B|}{|P|} \leq \frac{1}{m}$.



$$\begin{aligned} \Pr_{(a,b) \sim \mathcal{U}(\mathbb{F}_p^* \times \mathbb{F}_p)} [h(x) = h(y)] &= \Pr_{(a,b)} [((a \times x) \oplus b) \bmod m = ((a \times y) \oplus b) \bmod m] \\ &= \Pr_{(a,b)} [c \bmod m = d \bmod m] = \Pr_{(a,b)} [(c, d) \in B] = \Pr_{(c,d) \sim \mathcal{U}(P)} [(c, d) \in B] = \frac{|B|}{|P|} \leq \frac{1}{m}. \quad \square \end{aligned}$$

Analysis of Hash Table with Chaining

... using a Universal Hash Family

Theorem

If $\mathcal{H} \subseteq [m]^D$ is a c -universal hash family then $T_{\text{chaining}}(n, m, \mathcal{H}) \leq 2 + c\alpha = \mathcal{O}(1)$ if $\alpha \in \mathcal{O}(1)$ and $c \in \mathcal{O}(1)$.

Proof: Mostly the same.

$$\begin{aligned} \forall S \subseteq [D], \forall x \in D : \quad & \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[1 + |\{y \in S \mid h(y) = h(x)\}| \right] \\ & \leq \dots \leq 2 + \sum_{y \in S \setminus \{x\}} \Pr_{h \sim \mathcal{U}(\mathcal{H})} [h(y) = h(x)] \\ & = 2 + \sum_{y \in S \setminus \{x\}} \frac{c}{m} \leq 2 + \frac{cn}{m} = 2 + c\alpha. \quad \square \end{aligned}$$

Examples for Universal Hash Families

- “ $((ax + b) \bmod p) \bmod m$ ” is 1-universal

as discussed: $D = \mathbb{F}_p$, $R = [m]$,

$$\mathcal{H}_{p,m}^{\text{lin}} := \{x \mapsto ((a \times b) \oplus b) \bmod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}$$

- “ $(ax \bmod p) \bmod m$ ” is only 2-universal^a:

$$D = \mathbb{F}_p, \quad R = [m],$$

$$\mathcal{H} = \{x \mapsto (a \times x) \bmod m \mid a \in \mathbb{F}_p^*\}$$

- **Multiply-Shift** is 1-universal:

$$D = \{0, \dots, 2^w - 1\}, \quad R = \{0, \dots, 2^\ell - 1\}$$

$$\mathcal{H} = \{x \mapsto \lfloor ((a \cdot x + b) \bmod 2^{2w}) / 2^{2w-\ell} \rfloor \mid$$

$$a, b \in \{0, \dots, 2^{2w} - 1\}\}.$$

Selling point of multiply shift:

- “ $x \bmod 2^{2w}$ ” drops some higher order bits
- “ $\lfloor x / 2^{2w-\ell} \rfloor$ ” drops some lower order bits
- No division or modulo operation needed!

For $w = 32$ (taken from Thorup 2015):

```
uint32_t hash(uint32_t x, uint32_t l,  
              uint64_t a, uint64_t b) {  
    return (a * x + b) >> (64-l);  
}
```

^aI failed to find an exact proof for this, though...

1. Conceptions: What is a Hash Function?

- Hashing in the Wild
- What should a Theorist do?

2. Use Case 1: Hash Table with Chaining

- Using SUHA
- Using Universal Hashing

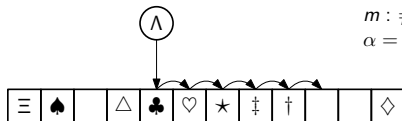
3. Use Case 2: Linear Probing

- Using SUHA
- Using Universal Hashing

4. Recent Developments

5. Conclusion

Hash Table with Linear Probing



S : set of n keys
 m : # of buckets
 $\alpha = n/m$

Operations

For key x probe buckets $h(x), h(x) + 1, h(x) + 2, \dots \pmod{m}$.

Insert. Put x into first empty bucket.

Lookup. Look for x , abort when encountering empty bucket.

Delete. Lookup and remove x and  check if a key to the right wants to move into the hole.^a

↪ For details see https://en.wikipedia.org/wiki/Linear_probing

^aAlternative implementations leaves special *tombstones* markers.

We Focus on Insertion Times

- Lookup($x \in S$): At most x 's insertion time.
- Lookup($x \notin S$): At most the time it *would take* to insert x now.
- Delete($x \in S$): At most the time it *would take* to insert $y \notin S$ with $h(y) = h(x)$.

Theorem: Linear Probing under SUHA

Let $T_{n,m}$ be the random insertion time into a linear probing hash table. If $\frac{1}{2} \leq \alpha < 1$ then under SUHA we have

$$\mathbb{E}[T_{n,m}] = \mathcal{O}\left(\frac{1}{(1-\alpha)^2}\right) = \mathcal{O}(1). \quad (\text{not here})$$

1. Conceptions: What is a Hash Function?

- Hashing in the Wild
- What should a Theorist do?

2. Use Case 1: Hash Table with Chaining

- Using SUHA
- Using Universal Hashing

3. Use Case 2: Linear Probing

- Using SUHA
- Using Universal Hashing

4. Recent Developments

5. Conclusion

Conceptions: What is a Hash Function?
○○○○○○○

Use Case 1: Hash Table with Chaining
○○○○○○○○○

Use Case 2: Linear Probing
○○●○○○○○○○○○

Recent Developments
○○○○○

Conclusion
○○○

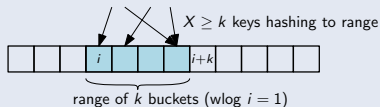
References

Preparation: A concentration bound

Chernoff

For $X \sim \text{Bin}(n, p)$ and $\delta \in [0, 1]$ we have $\Pr[X \geq (1 + \delta)\mathbb{E}[X]] \leq \exp(-\delta^2\mathbb{E}[X]/3)$.

Lemma: $\Pr[\geq k \text{ hits in segment of length } k]$



Let $k \in \mathbb{N}$ and $X = |\{y \in S \mid h(y) \in \{1, \dots, k\}\}|$.

Then $\Pr_{h \sim \mathcal{U}(R^D)}[X \geq k] \leq \exp(-(1 - \alpha)^2 k/3)$.

Proof

Let $S = \{x_1, \dots, x_n\}$ and $X_i = [h(x_i) \in \{1, \dots, k\}] \sim \text{Ber}(\frac{k}{m})$. Then $X = \sum_{i \in [n]} X_i \sim \text{Bin}(n, \frac{k}{m})$ with $\mathbb{E}[X] = \frac{kn}{m} = \alpha k$.

$$\begin{aligned} \Pr[X \geq k] &= \Pr[X \geq \frac{1}{\alpha}\mathbb{E}[X]] \\ &= \Pr[X \geq (1 + \frac{1-\alpha}{\alpha})\mathbb{E}[X]] \\ &\leq \exp(-(\frac{1-\alpha}{\alpha})^2 \alpha k/3) \\ &\leq \exp(-(1 - \alpha)^2 k/3). \end{aligned}$$

Proof: Expected LP-Insertion Time under SUHA is $\mathcal{O}(1)$

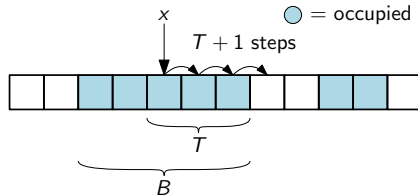
$$\mathbb{E}[T] \leq \mathbb{E}[B] = \sum_{k \geq 1} k \cdot \Pr[B = k] = \sum_{k \geq 1} k \cdot \Pr \left[\bigcup_{d=0}^{k-1} A_{h(x)-d, h(x)-d+k-1} \right]$$

$$\stackrel{(1)}{\leq} \sum_{k \geq 1} k \cdot \sum_{d=0}^{k-1} \Pr \left[A_{h(x)-d, h(x)-d+k-1} \right] \stackrel{(2)}{=} \sum_{k \geq 1} k \cdot k \cdot \Pr[A_{1,k}]$$

$$\stackrel{(3)}{\leq} \sum_{k \geq 1} k^2 \cdot \Pr[|\{y \in S \mid h(y) \in \{1, \dots, k\}\}| \geq k]$$

$$\stackrel{(4)}{\leq} \sum_{k \geq 1} k^2 \cdot \exp(-(1 - \alpha)^2 k/3) = \mathcal{O}(1).$$

Wolfram Alpha gives: $\int_0^{\infty} k^2 \exp(-(1 - \alpha)^2 k/3) = \frac{54}{(1 - \alpha)^6}.$



$A_{u,v} : \{u, v\}$ is maximal occupied block:



Reasoning:

- (1) Union Bound. (actually "=")
- (2) $h(x)$ is independent of keys in the table and hash distribution is invariant under cyclic shifts.
- (3) Note: Keys stored in block cannot come in from the left.
- (4) Lemma from previous slide.

1. Conceptions: What is a Hash Function?

- Hashing in the Wild
- What should a Theorist do?

2. Use Case 1: Hash Table with Chaining

- Using SUHA
- Using Universal Hashing

3. Use Case 2: Linear Probing

- Using SUHA
- Using Universal Hashing

4. Recent Developments

5. Conclusion

(Mutual / Collective) Independence

A family $\mathcal{E}$ of **events** is **independent** if $\forall k \in \mathbb{N}$ and distinct $E_1, \dots, E_k \in \mathcal{E}$ we have

$$\Pr \left[\bigcap_{i=1}^k E_i \right] = \prod_{i=1}^k \Pr[E_i].$$

A family $\mathcal{X}$ of discrete **random variables** is **independent** if $\forall k \in \mathbb{N}$, distinct $X_1, \dots, X_k \in \mathcal{X}$ and all $x_1, \dots, x_k$ we have

$$\Pr \left[\bigwedge_{i=1}^k X_i = x_i \right] = \prod_{i=1}^k \Pr[X_i = x_i].$$

Pairwise Independence

A family of **events** is **pairwise independent** if any subfamily of size 2 is independent.

A family of **random variables** is **pairwise independent** if any subfamily of size 2 is independent.

d -wise Independence

A family of **events** is **d -wise independent** if any subfamily of size at most d is independent.

A family of **random variables** is **d -wise independent** if any subfamily of size at most d is independent.

d -Independent Hash Family

Definition: d -Independent Hash Family

A family $\mathcal{H} \subseteq R^D$ of hash functions is d -independent if for distinct $x_1, \dots, x_d \in D$ and any $i_1, \dots, i_d \in R$: (grey is implied by black)

$$\Pr_{h \sim \mathcal{U}(\mathcal{H})} [h(x_1) = i_1 \wedge \dots \wedge h(x_d) = i_d] = \prod_{j=1}^d \Pr_{h \sim \mathcal{U}(\mathcal{H})} [h(x_j) = i_j] = |R|^{-d}.$$

Theorem

Let $D = R = \mathbb{F}$ be a finite field. Then

$$\mathcal{H} := \{x \mapsto \sum_{i=0}^{d-1} a_i x^i \mid a_0, \dots, a_{d-1} \in \mathbb{F}\}$$

is a d -independent family.

Note: $\mathcal{H} \subseteq \mathbb{F}^{\mathbb{F}} \rightsquigarrow$ not yet useful.

Alternative Definition

$\mathcal{H}$ is d -independent if for $h \sim \mathcal{U}(\mathcal{H})$

- the family $(h(x))_{x \in D}$ of random variables is d -independent and
- $h(x) \sim \mathcal{U}(R)$ for each $x \in D$.

Corollary: Smaller Ranges (proof omitted)

- If m divides $|\mathbb{F}|$, then adding “mod m ” gives a d -independent family $\mathcal{H}' \subseteq [m]^{\mathbb{F}}$.
- If m does not divide $|\mathbb{F}|$, then adding “mod m ” gives a family $\mathcal{H}' \subseteq [m]^{\mathbb{F}}$ such that for $h \sim \mathcal{U}(\mathcal{H}')$ the family $(h(x))_{x \in \mathbb{F}}$ is d -independent but only *approximately* uniformly distributed in $[m]$.

Proof: $\mathcal{H} := \{x \mapsto \sum_{i=0}^{d-1} a_i x^i \mid a_0, \dots, a_{d-1} \in \mathbb{F}\}$ is d -independent

Let $x_1, \dots, x_d \in \mathbb{F}$ be distinct keys and $i_1, \dots, i_d \in \mathbb{F}$ arbitrary.

$\hookrightarrow$ to show : $\Pr_{h \sim \mathcal{U}(\mathcal{H})}[\forall j \in [d] : h(x_j) = i_j] = |\mathbb{F}|^{-d}$.

For $h \in \mathcal{H}$ (given via $a_0, \dots, a_{d-1}$) the following is equivalent:

$$\begin{array}{ll} h(x_1) = i_1 & a_0 + a_1 x_1 + \dots + a_{d-1} x_1^{d-1} = i_1 \\ h(x_2) = i_2 & a_0 + a_1 x_2 + \dots + a_{d-1} x_2^{d-1} = i_2 \\ \vdots & \vdots \\ h(x_d) = i_d & a_0 + a_1 x_d + \dots + a_{d-1} x_d^{d-1} = i_d \end{array} \iff \underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{d-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{d-1} \\ \vdots & & & \ddots & \vdots \\ 1 & x_d & x_d^2 & \dots & x_d^{d-1} \end{pmatrix}}_{\text{Vandermonde matrix } M \Rightarrow \text{regular}} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{pmatrix} = \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_d \end{pmatrix}$$

Exactly one vector $\vec{a} = M^{-1} \cdot \vec{i}$ solves the equation.

$$\Rightarrow \Pr_{h \sim \mathcal{U}(\mathcal{H})}[\forall j : h(x_j) = i_j] = \Pr_{a_0, \dots, a_{d-1} \sim \mathcal{U}(\mathbb{F})}[\vec{a} = M^{-1} \cdot \vec{i}] = |\mathbb{F}|^{-d}. \quad \square$$

Concentration Bound for d -Independent Variables

(Tricky) Exercise

Let d be even and $X_1, \dots, X_n \sim \text{Ber}(p)$ a d -independent family of random variables with $p = \Omega(1/n)$. Let $X = \sum_{i=1}^n X_i$. Then for any $\delta > 0$ we have

$$\Pr[X - \mathbb{E}[X] \geq \delta \mathbb{E}[X]] = \mathcal{O}(\delta^{-d} \mathbb{E}[X]^{-d/2}).$$

Remark: Weaker than Chernoff, stronger than Chebyshev

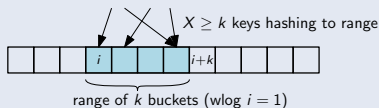
- Chebycheff gives $\Pr[X - \mathbb{E}[X] \geq \delta \mathbb{E}[X]] \leq \frac{1-p}{\delta^2 \mathbb{E}[X]}$. (requires $d = 2$)
 $\hookrightarrow$ uses that $\text{Var}(X_1 + \dots + X_n) = \text{Var}(X_1) + \dots + \text{Var}(X_n)$ for pairwise independent $X_1, \dots, X_n$.
- Chernoff gave $\Pr[X - \mathbb{E}[X] \geq \delta \mathbb{E}[X]] \leq \exp(-\delta^2 \mathbb{E}[X]/3)$. (requires $d = n$).

Preparation: A Concentration Bound again for d -independence

Lemma (last slide)

For d -independent $X_1, \dots, X_n \sim \text{Ber}(p)$ and $X = \sum_{i \in [n]} X_i$ we have $\Pr[X \geq (1 + \delta)\mathbb{E}[X]] = \mathcal{O}(\delta^{-d}\mathbb{E}[X]^{-d/2})$.

Lemma: $\geq k$ hits in segment of length k



Let $\mathcal{H}$ be a d -independent hash family and $h \sim \mathcal{U}(\mathcal{H})$.
Let $k \in \mathbb{N}$ and $X = |\{y \in S \mid h(y) \in \{1, \dots, k\}\}|$.

Then $\Pr[X \geq k] \leq \mathcal{O}((1 - \alpha)^{-d}k^{-d/2})$.

Proof

Let $S = \{x_1, \dots, x_n\}$ and

$X_i = [h(x_i) \in \{1, \dots, k\}] \sim \text{Ber}(\frac{k}{m})$.

Then $X = \sum_{i \in [n]} X_i$ fits the Lemma with $\mathbb{E}[X] = \frac{kn}{m} = \alpha k$.

$$\begin{aligned} \Pr[X \geq k] &= \Pr[X \geq \frac{1}{\alpha}\mathbb{E}[X]] \\ &= \Pr[X \geq (1 + \frac{1-\alpha}{\alpha})\mathbb{E}[X]] \\ &= \mathcal{O}\left(\left(\frac{1-\alpha}{\alpha}\right)^{-d}(\alpha k)^{-d/2}\right) \\ &\leq \mathcal{O}((1 - \alpha)^{-d}k^{-d/2}). \quad (\text{using } \alpha \leq 1) \end{aligned}$$

Theorem: Linear Probing with d -independence

Under the same conditions as before, except with 9-independent hash functions, the insertion time $T_{n,m}$ for linear probing satisfies:

$$\mathbb{E}[T_{n,m}] = \mathcal{O}(1)$$

Proof Sketch

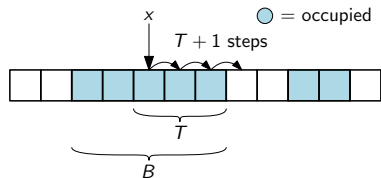
$$\mathbb{E}[T] \leq \mathbb{E}[B] \leq \dots$$

$$\stackrel{(1)}{\leq} \sum_{k \geq 1} k^2 \cdot \Pr[|\{y \in S \mid h(y) \in \{1, \dots, k\}\}| \geq k]$$

$$\stackrel{(2)}{\leq} \sum_{k \geq 1} k^2 \cdot \mathcal{O}((1 - \alpha)^{-8} k^{-8/2})$$

$$\leq \sum_{k \geq 1} k^{-2} \cdot \mathcal{O}((1 - \alpha)^{-8})$$

$$\stackrel{(3)}{=} \frac{\pi^2}{6} \mathcal{O}((1 - \alpha)^{-8}) = \mathcal{O}(1). \quad \square$$



$A_{u,v} : \{u, v\}$ is a maximal occupied block:



Reasoning:

- (1) Same as before, except we have to condition on $h(x)$ and may only use 8-independence in the following. (this is the hand wavy part!)
- (2) Concentration bound from previous slide for $d = 8$.
- (3) If interested, see 3Blue1Brown video: <https://www.youtube.com/watch?v=d-o3eB9sfls>

Much more is known about insertion times of linear probing:

- Any 5-independent family gives $\mathcal{O}\left(\frac{1}{(1-\alpha)^2}\right)$.
↪ A. Pagh, R. Pagh, and Ruzic 2011
- An (artificially bad) 4-independent family gives $\Omega(\log n)$.
↪ Pătraşcu and Thorup 2016
- A (well-designed) 3-independent family gives $\mathcal{O}\left(\frac{1}{(1-\alpha)^2}\right)$.
↪ Aamand et al. 2020

1. Conceptions: What is a Hash Function?

- Hashing in the Wild
- What should a Theorist do?

2. Use Case 1: Hash Table with Chaining

- Using SUHA
- Using Universal Hashing

3. Use Case 2: Linear Probing

- Using SUHA
- Using Universal Hashing

4. Recent Developments

5. Conclusion

Conceptions: What is a Hash Function?
○○○○○○

Use Case 1: Hash Table with Chaining
○○○○○○○○

Use Case 2: Linear Probing
○○○○○○○○○○○○○○

Recent Developments
●○○○○

Conclusion
○○○

References

Theoretically Good¹ Hash Tables

Remark: Storing $x \in D$ using $\log_2(|D|)$ Bits Wastes $\Theta(\log_2 n)$ bits per key (Cleary 1984)

Example: If $D = [2n]$ and $|S| = n$, then bitvector of $2n$ bits suffices.

↪ Much smaller than Array with $\mathcal{O}(n)$ entries of $\log_2 |D|$ bits!

In General: Can aim for $\text{OPT} = \log_2 \binom{|D|}{n} = n \log_2(|D|) - \Theta(n \log_2(n))$ bits rather than $n \cdot \log_2 |D|$.

↪ better than load factor 1

Theoreticians can “have it all” (Arbitman, Naor, and Segev 2010)

- worst case $\mathcal{O}(1)$ operations with high probability (e.g. probability $1 - n^{-100}$)
- *succinctness*, i.e. memory usage $(1 + o(1))\text{OPT}$
- explicit families of hash functions

¹and practically bad

Conceptions: What is a Hash Function?
○○○○○○○

Use Case 1: Hash Table with Chaining
○○○○○○○○○

Use Case 2: Linear Probing
○○○○○○○○○○○○○

Recent Developments
○●○○○

Conclusion
○○○

References

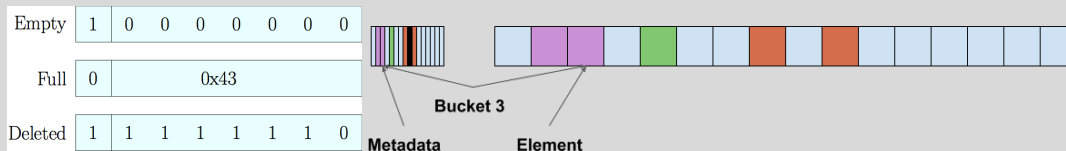
Fast hashing with strong concentration bounds (Aamand et al. 2020)

Variants of “tabulation hashing”:

- are only 3-independent
- + offer Chernoff-style concentration bounds
- + are much faster than highly independent hash functions
- are still slower than what algorithm engineers would use

Linear Probing holds up well

Used at Google: “SwissTable” / `absl::flat_hash_set` = Linear Probing + SIMD



<https://abseil.io/about/design/swisstable>

Linear Probing: Is “Primary Clustering” a Myth?

Theorem: Primary Clustering

Assume we insert $n = (1 - \varepsilon)m$ elements into a linear probing hash table of size m . Then the last insertion takes $\mathcal{O}(1/\varepsilon^2)$ time in expectation.

Linear Probing Revisited²

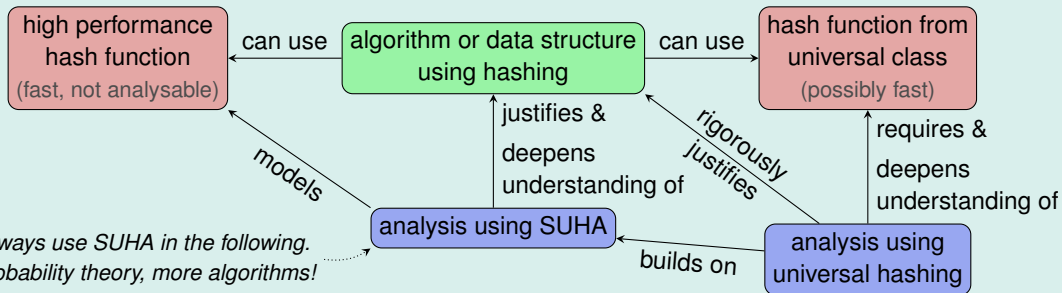
- Linear probing tables with tombstones give *amortised* times of $\tilde{\mathcal{O}}(1/\varepsilon^{3/2})$.
- Graveyard hashing improves this to $\mathcal{O}(1/\varepsilon)$.

²Linear Probing Revisited: Tombstones Mark the Demise of Primary Clustering Bender, B. C. Kuszmaul, and W. Kuszmaul 2022

Technical Takeaway: Performance of Hash Tables

For both an **ideal hash function** (SUHA) and a random hash function from a suitable **universal class**, a hash table using **linear probing** or **chaining** provably has an expected running time of $\mathcal{O}(1)$ per operation.

Non-Technical Takeaway: Approaches to analyse hashing based algorithms



- Was könnte eine Idealvorstellung einer Hashfunktion sein? Inwiefern wäre eine ideale Hashfunktion nützlich? Was ist das Problem an dieser Vorstellung?
- Was ist die Simple Uniform Hashing Assumption (SUHA)? Was spricht dafür diese Annahme zu treffen? Welche Alternativen gibt es?
- Inwiefern ist eine pseudozufällige Funktion mit kryptographischen Ununterscheidbarkeitsgarantien nützlich für uns? Wie ist der Zusammenhang zur SUHA?*
- Universelles Hashing:
 - Wie ist c -Universalität definiert?
 - Welche c -universellen Hashklasse haben wir kennengelernt? Wie haben wir die c -Universalität bewiesen?
 - Wie ist d -Unabhängigkeit für eine Hashklasse definiert?
 - Welche d -universelle Hashklasse haben wir kennengelernt?
 - Welcher Zusammenhang besteht zwischen d -Unabhängigkeit und c -Universalität? (Übungsaufgabe)
 - Chernoff Schranken sind für Summen unabhängiger Zufallsvariablen gedacht. Was kann man machen, wenn die Zufallsvariablen nur d -unabhängig sind?*

- Betrachten wir Hashing mit verketteten Listen:
 - Welche Schranke an die erwartete Einfügezeit haben wir bewiesen? Wie?
 - An welcher Stelle spielt die Verteilung der Hashfunktion eine Rolle?
 - Nenne eine hinreichende Eigenschaft, die eine universelle Hashklasse haben sollte, damit der Beweis funktioniert.
- Betrachten wir Hashing mit linearem Sondieren:
 - Welche Schranke an die erwartete Laufzeit haben wir bewiesen? Wie?
 - An welcher Stelle spielt die Verteilung der Hashfunktion eine Rolle?
 - Nenne eine hinreichende Eigenschaft, die eine universelle Hashklasse haben sollte, damit der Beweis funktioniert.
 - Wie wir diese Eigenschaft ausgenutzt?*

- [1] Anders Aamand et al. “Fast hashing with strong concentration bounds”. In: *STOC*. ACM, 2020, pp. 1265–1278. ISBN: 9781450369794. DOI: 10.1145/3357713.3384259.
- [2] Yuriy Arbitman, Moni Naor, and Gil Segev. “Backyard Cuckoo Hashing: Constant Worst-Case Operations with a Succinct Representation”. In: *Proc. 51th FOCS*. 2010, pp. 787–796. DOI: 10.1109/FOCS.2010.80.
- [3] Michael A. Bender, Bradley C. Kuszmaul, and William Kuszmaul. “Linear Probing Revisited: Tombstones Mark the Demise of Primary Clustering”. In: *FOCS*. 2022, pp. 1171–1182. DOI: 10.1109/FOCS52979.2021.00115.
- [4] John G. Cleary. “Compact Hash Tables Using Bidirectional Linear Probing”. In: *IEEE Trans. Computers* 33.9 (1984), pp. 828–834. DOI: 10.1109/TC.1984.1676499.
- [5] Anna Pagh, Rasmus Pagh, and Milan Ruzic. “Linear Probing with 5-wise Independence”. In: *SIAM Rev.* 53.3 (2011), pp. 547–558. DOI: 10.1137/110827831. URL: <https://doi.org/10.1137/110827831>.

- [6] Mihai Pătraşcu and Mikkel Thorup. “On the k -Independence Required by Linear Probing and Minwise Independence”. In: *ACM Trans. Algorithms* 12.1 (2016), 8:1–8:27. DOI: 10.1145/2716317. URL: <https://doi.org/10.1145/2716317>.
- [7] Mikkel Thorup. “High Speed Hashing for Integers and Strings”. In: *CoRR* abs/1504.06804 (2015). arXiv: 1504.06804. URL: <http://arxiv.org/abs/1504.06804>.