Prüfungsanmeldung

- Am einfachsten: Hier angeben, wann ihr Zeit habt:
  https://www.terminplaner.dfn.de/W4m8QyA9vvp1K19m
- Alternativ: Email an Stefan und Max.
- Wir bieten euch dann einen Termin per Email an.
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. Conclusion
Hash Table with Chaining
e.g. std::unordered_set, java.util.HashMap

Terminology

\[ D: \text{ Universe (or domain) of keys} \]
(strings, integers, game states in chess)

\[ S \subseteq D: \text{ set of } n \text{ keys (possibly with associated data)} \]

\[ h : D \rightarrow R: \text{ hash function, range usually } R = [m] \]

\[ \alpha = \frac{n}{m}: \text{ load factor, } \alpha \leq \alpha_{\text{max}} = O(1) \]
**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 ⊆ 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, $\alpha \leq \alpha_{\text{max}} = \mathcal{O}(1)$

## Goal

Operations in time $t$ with $\mathbb{E}[t] = \mathcal{O}(1)$.

Randomness comes from the hash function.

---

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
**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 ⊆ D**: set of \( n \) keys (possibly with associated data)
- **h : D → R**: hash function, range usually \( R = [m] \)
- **\( \alpha = \frac{n}{m} \)**: load factor, \( \alpha \leq \alpha_{\text{max}} = \mathcal{O}(1) \)

---

**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 \( h \).

---

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

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

**Use Case 2: Linear Probing**

**Conclusion**

**References**
Naive Idea

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

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

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

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

$\log_2(|R|^{|D|}) = |D| \cdot \log_2(|R|)$ bits to store $h \sim U(R^D)$ $\Rightarrow$ 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. Conclusion
What is a Hash Function?
(it depends on who you ask)
What is a Hash Function?
(it depends on who you ask)

Cryptographic Hash Function

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

\[
\text{sha256sum myfile.txt}
\]

018a7eaee8a...3e79043e21ab4 myfile.txt

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

\( \rightarrow \) Files with equal hashes are likely the same.
What is a Hash Function?
(it depends on who you ask)

Cryptographic Hash Function

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

```bash
$ \text{sha256sum myfile.txt}
018a7eaee8a...3e79043e21ab4  myfile.txt
```

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

$\rightarrow$ 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.
**What is a Hash Function?**
(it depends on who you ask)

---

**Cryptographic Hash Function**

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

```shell
$ \text{sha256sum} \text{ myfile.txt }
018a7eaee8a...3e79043e21ab4 \text{ myfile.txt }
```

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

$\rightarrow$ 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.

---

**Hash Function in Algorithm Engineering**

- typically small range $|R| = O(n)$
  $\leftrightarrow$ cannot be collision resistant
- should **behave like** $h \sim \mathcal{U}(R^D)$ in my application
- should be **fast** to evaluate

---

Conceptions: What is a Hash Function?  Use Case 1: Hash Table with Chaining  Use Case 2: Linear Probing  Conclusion  References

7/36  WS 2023/2024  Stefan Walzer, Maximilian Katzmann: Classic Hash Tables

ITI, Algorithm Engineering & Scalable Algorithms
## What is a Hash Function?

(it depends on who you ask)

### Cryptographic Hash Function

A collision resistant function such as \( h = \text{sha256sum} \)

\[
\text{sha256sum} \text{ myfile.txt }
\]

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

\( \rightarrow \) 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.

### Hash Function in Algorithm Engineering

- typically small range \( |R| = \mathcal{O}(n) \)
  \( \leftrightarrow \) 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

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

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

#### Use Case 2: Linear Probing

#### Conclusion

#### References
What is a Hash Function?
(it depends on who you ask)

Cryptographic Hash Function

A collision resistant function such as \( h = \text{sha256sum} \)

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

\( \implies \) 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.

Hash Function in Algorithm Engineering

- typically small range \( |R| = \mathcal{O}(n) \)
  \( \implies \) 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.

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
What is a Hash Function?
(it depends on who you ask)

Cryptographic Hash Function

A collision resistant function such as \( h = \text{sha256sum} \)

\[
\text{sha256sum myfile.txt} \nonumber \\
018a7eaee8a...3e79043e21ab4 \text{ myfile.txt} \nonumber 
\]

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

\( \iff \) 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.

Hash Function in Algorithm Engineering

- typically small range \( |R| = \mathcal{O}(n) \)
  \( \iff \) 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.

Conceptions: What is a Hash Function? 

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
MurmurHash

Bitshifts, Magic Constants, ... 

```c
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; 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 $\sim U(\{0, 1\}^32)$ and use $h(x) = \text{murmur3}_32(x, \text{seed}) \mod m$.

(should avoid modulo in practice, see https://github.com/lemire/fastrange)

Does $h$ behave like a random function? 

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

**NO**, HashDoS attacks are known. 

**MAYBE**, for your favourite application.

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
MurmurHash

Usage

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

\[
h(x) = \text{murmur3}_32(x, \text{seed}) \mod m.
\]
Usage

For $R = \lfloor m \rfloor$, pick seed $\sim U\{0, 1\}^{32}$ and use

$$h(x) = \text{murmur3	extunderscore 32}(x, \text{seed}) \mod m.$$  

(should avoid modulo in practice, see https://github.com/lemire/fastrange)

Conceptions: What is a Hash Function?

Usage Case 1: Hash Table with Chaining

Usage Case 2: Linear Probing

Conclusion

References

8/36

WS 2023/2024

Stefan Walzer, Maximilian Katzmann: Classic Hash Tables

ITI, Algorithm Engineering & Scalable Algorithms
MurmurHash

Bitshifts, Magic Constants, . . .

```c
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; 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

\[
h(x) = \text{murmur3}_32(x, \text{seed}) \mod m.
\]

(should avoid modulo in practice, see https://github.com/lemire/fastrange)

Does \( h \) behave like a random function?

YES, with respect to many statistical tests.

NO, HashDoS attacks are known. see https://en.wikipedia.org/wiki/MurmurHash#Vulnerabilities

MAYBE, for your favourite application.
MurmurHash

Bitshifts, Magic Constants, ...

```c
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; 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;
}
```

```c
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 \( \sim U(\{0, 1\}^{32}) \) and use

\[
h(x) = \text{murmur3}_32(x, \text{seed}) \mod m.
\]

(should avoid modulo in practice, see https://github.com/lemire/fastrange)

Does \( h \) behave like a random function?

- **YES**, with respect to many statistical tests.
- **NO**, HashDoS attacks are known.
- **MAYBE**, for your favourite application.

---

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

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

**Use Case 2: Linear Probing**

**Conclusion**

**References**
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; 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 $R = \{0, 1\}^32$, pick seed $\sim \mathcal{U}(\{0, 1\}^32)$ and use

$$h(x) = \text{murmur3}_32(x, \text{seed}) \mod m.$$  

(should avoid modulo in practice, 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

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
Usage

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

$$h(x) = \text{murmur3}_32(x, \text{seed}) \mod m.$$  

(should avoid modulo in practice, 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.

---

Hashing in Practice

Black Magic, do not touch!
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. 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.
Simple Uniform Hashing Assumption (SUHA)

- We have access to \( h \sim \mathcal{U}(R^D) \) for any \( R \) and \( D \).
- \( h \) takes \( O(1) \) 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.

What should a Theorist do?

Approach 1: Ignore the Problem
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 $O(1)$ 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

Conceptions: What is a Hash Function?
Use Case 1: Hash Table with Chaining
Use Case 2: Linear Probing
Conclusion
References
Simple Uniform Hashing Assumption (SUHA)

- We have access to $h \sim U(R^D)$ for any $R$ and $D$.
- $h$ takes $O(1)$ 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.
  - $\leftrightarrow$ like e.g. ideal gas law in physics
- Excellent track record in non-adversarial settings.

What should a Theorist do?

Approach 1: Ignore the Problem
### Analyse Algorithm using Universal Hashing

1. Define family $\mathcal{H} \subseteq \mathbb{R}^D$ of hash functions with $\log(|\mathcal{H}|)$ not too large.  
   → sampling and storing $h \in \mathcal{H}$ is cheap

2. Proof that algorithm with $h \sim \mathcal{U}(\mathcal{H})$ has good expected behaviour.
What should a Theorist do?
Approach 2: Bring your own Hash Functions

Analyse Algorithm using Universal Hashing

1. Define family $\mathcal{H} \subseteq \mathbb{R}^D$ of hash functions with $\log(|\mathcal{H}|)$ not too large.
   $\leftrightarrow$ 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

---

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
Analyse Algorithm using Universal Hashing

1. Define family $\mathcal{H} \subseteq \mathbb{R}^D$ of hash functions with $\log(|\mathcal{H}|)$ not too large.
   $\iff$ 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.
Analyse Algorithm using Universal Hashing

1. Define family $\mathcal{H} \subseteq R^D$ of hash functions with $\log(|\mathcal{H}|)$ not too large. 
   → 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.
### Analyse Algorithm using Universal Hashing

1. Define family $\mathcal{H} \subseteq R^D$ of hash functions with $\log(|\mathcal{H}|)$ not too large.  
   - $\implies$ 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.  
  - $\implies$ 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.
     - \( \Rightarrow \) Your use case can tell the difference between \( f \) and true randomness.
     - \( \leftarrow \) The cryptographers said this is impossible. 😕
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.

Should we use cryptographic pseudorandom functions?

---

Conceptions: What is a Hash Function?
Use Case 1: Hash Table with Chaining
Use Case 2: Linear Probing
Conclusion
References

WS 2023/2024 Stefan Walzer, Maximilian Katzmann: Classic Hash Tables
ITI, Algorithm Engineering & Scalable Algorithms
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.

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
  

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
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. 😡

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

  ![Hash Function MiB / sec](source: https://github.com/rurban/smhasher)

- **NO.** Too slow in high-performance settings.

Conceptions: What is a Hash Function?

<table>
<thead>
<tr>
<th>Use Case 1: Hash Table with Chaining</th>
<th>Use Case 2: Linear Probing</th>
<th>Conclusion</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**What should a Theorist do?**

**Approach 3: Let the Cryptographers do the Work**
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. Conclusion
Search Time under Chaining

\[
\max_{S \subseteq D} \max_{x \in D} \left| S \right| = n \\
1 + \left| \{ y \in S \mid h(y) = h(x) \} \right|
\]

Key set is worst case. Only \( h \in H \) is random. Key set is fixed before \( h \) is chosen.

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
Search Time under Chaining

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

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

Key set is worst case. Only \( h \in H \) is random. Key set is fixed before \( h \) is chosen.
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

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

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

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

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

⚠️ Key set is \textit{worst case}. Only \( h \in \mathcal{H} \) is random. Key set is fixed \textit{before} \( h \) is chosen.

Theorem: Hash Table with Chaining under SUHA

If \( \mathcal{H} = [m]^D \) then

\[
T_{\text{chaining}}(n, m, \mathcal{H}) \leq 2 + \alpha = O(1) \text{ if } \alpha \in O(1).
\]
Theorem: Hash Table with Chaining under SUHA

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

\[
\mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})}\left[ 1 + \left| \{y \in S \mid h(y) = h(x)\} \right| \right] \leq 2 + \alpha
\]
Analysis of Hash Table with Chaining under SUHA

**Theorem: Hash Table with Chaining under SUHA**

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

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

**Proof.**

$$\mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[ 1 + \left| \{ y \in S \mid h(y) = h(x) \} \right| \right]$$
Analysis of Hash Table with Chaining under SUHA

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.

$$
\mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[ 1 + |\{y \in S \mid h(y) = h(x)\}| \right]
= \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[ 1 + \sum_{y \in S} 1_{\{h(y) = h(x)\}} \right]
$$
Analysis of Hash Table with Chaining under SUHA

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.

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

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
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.

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

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
Theorem: Hash Table with Chaining under SUHA

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

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

Proof.

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

$$= 1 + 1 + \sum_{y \in S \setminus \{x\}} \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(y) = h(x)]$$

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
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.

\[
\begin{align*}
\mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[ 1 + |\{ y \in S \mid h(y) = h(x) \}| \right] &= 1 + \sum_{y \in S} \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[ 1 \{ h(y) = h(x) \} \right] \\
&= 1 + \sum_{y \in S} \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(y) = h(x)] \\
&= 1 + 1 + \sum_{y \in S \setminus \{x\}} \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(y) = h(x)] \\
&= 2 + \sum_{y \in S \setminus \{x\}} \frac{1}{m} \leq 2 + \frac{n}{m} = 2 + \alpha. \quad \Box
\end{align*}
\]
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. Conclusion
Definition: c-universal hash family

A class \( \mathcal{H} \subseteq [m]^D \) is called \( c \)-universal if:

\[
\forall x \neq y \in D : \quad \Pr_{h \sim \mathcal{U}(\mathcal{H})} [h(x) = h(y)] \leq \frac{c}{m}.
\]
A Universal Hash Family

**Definition: c-universal hash family**

A class $\mathcal{H} \subseteq [m]^D$ is called *c-universal* if:

$$\forall x \neq y \in D : \Pr_{h \sim \mathcal{H}}[h(x) = h(y)] \leq \frac{c}{m}.$$ 

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

A class $\mathcal{H} \subseteq [m]^D$ is called $c$-universal if:

$$\forall x \neq y \in D : \Pr_{h \sim U(\mathcal{H})}[h(x) = h(y)] \leq \frac{c}{m}.$$
**Definition: c-universal hash family**

A class $\mathcal{H} \subseteq [m]^D$ is called $c$-universal if:

$$\forall x \neq y \in D : \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x) = h(y)] \leq \frac{c}{m}.$$  

**Reminder (?.): Finite Fields**

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

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

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

Definition: $c$-universal hash family

A class $\mathcal{H} \subseteq [m]^D$ is called $c$-universal if:

$$\forall x \neq y \in D : \Pr_{h \sim \mathcal{U}(\mathcal{H})} [h(x) = h(y)] \leq \frac{c}{m}.$$

Reminder (?): Finite Fields

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

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

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:

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

Let $x \neq y \in \mathbb{F}_p$. (To show: $\Pr_{h \sim \mathcal{H}_{p,m}^{\text{lin}}} [h(x) = h(y)] \leq 1/m.$)
Proof that $\mathcal{H}_{p,m}^{\text{lin}} := \{ x \mapsto ((a \times x) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p \}$ 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
  
  \[ c = (a \times x) \oplus b \]
  \[ d = (a \times y) \oplus b \]
Proof that $\mathcal{H}_{p,m}^{\text{lin}} := \{ x \mapsto ((a \times x) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p \}$ 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 $c = (a \times x) \oplus b$ and $d = (a \times y) \oplus b \iff \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$.

regular!
Proof that \( H_{p,m}^{\text{lin}} := \{ x \mapsto ((a \times x) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p \} \) is 1-universal.

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

- Define \( c = (a \times x) \oplus b \) \( d = (a \times y) \oplus b \) \( \iff \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \). regular!

- The mapping \( (a, b) \mapsto (c, d) \) is a bijection (for every \( x \neq y \)) from \( \mathbb{F}_p \times \mathbb{F}_p \to \mathbb{F}_p \times \mathbb{F}_p \)
Proof that $\mathcal{H}_{p,m}^{\text{lin}} := \{ x \mapsto ((a \times x) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p \}$ 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 $c = (a \times x) \oplus b$
  $d = (a \times y) \oplus b \iff (c, d) = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$.

- The mapping $(a, b) \mapsto (c, d)$ is a bijection (for every $x \neq y$)
  from $\mathbb{F}_p^* \times \mathbb{F}_p \rightarrow \mathbb{F}_p \times \mathbb{F}_p \setminus \{(b, b) \mid b \in \mathbb{F}_p\}$

$P := \mathbb{F}_p \times \mathbb{F}_p \setminus \{(b, b) \mid b \in \mathbb{F}_p\}$
Proof that $H_{p,m}^{\text{lin}} := \{ x \mapsto ((a \times x) \oplus b) \mod m \mid a \in F_p^*, b \in F_p \}$ is 1-universal.

Let $x \neq y \in F_p$. (To show: $\Pr_{h \sim H_{p,m}^{\text{lin}}} [h(x) = h(y)] \leq 1/m$.)

- Define
  
  \[
  c = (a \times x) \oplus b \\
  d = (a \times y) \oplus b
  \]

  \[
  \iff \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}.
  \]

  - The mapping $(a, b) \mapsto (c, d)$ is a bijection (for every $x \neq y$) from $F_p^* \times F_p \to P$.

\[
P := F_p \times F_p \setminus \{(b, b) \mid b \in F_p\}
\]
Proof that \( \mathcal{H}_{p,m}^{\text{lin}} := \{ x \mapsto ((a \times x) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p \} \) 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 
  
  \[
  c = (a \times x) \oplus b \\
  d = (a \times y) \oplus b 
  \]
  
  \[
  \iff \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}.
  \]

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

- Define bad set \( B := \{(c, d) \in P \mid c \mod m = d \mod m \} \).

\[ \leftarrow \text{from picture: } \frac{|B|}{|P|} \leq \frac{1}{m}. \]

\[ d \] 
\[ \uparrow \]
\[ p-1 \] 
\[ \text{\( (p = 13, m = 4) \)} \]

\[ c \]
\[ \downarrow \]
\[ 0 \] 
\[ 0 \] 
\[ p-1 \]

\[
P := \mathbb{F}_p \times \mathbb{F}_p \setminus \{(b, b) \mid b \in \mathbb{F}_p \}
\]
Proof that $H_{p,m}^{\text{lin}} := \{ x \mapsto ((a \times x) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p \}$ is 1-universal.

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

- Define $c = (a \times x) \oplus b$ \quad $d = (a \times y) \oplus b$ \quad $\iff \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$.

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

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

$\Pr_{a,b \sim \mathcal{U}(\mathbb{F}_p^* \times \mathbb{F}_p)}[h(x) = h(y)]$
Proof that $\mathcal{H}_{p,m}^{\text{lin}} := \{ x \mapsto ((a \times x) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p \}$ 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 $c = (a \times x) \oplus b$, $d = (a \times y) \oplus b$ $\iff$ $\left( \begin{array}{c} c \\ d \end{array} \right) = \left( \begin{array}{cc} x & 1 \\ y & 1 \end{array} \right) \left( \begin{array}{c} a \\ b \end{array} \right)$.

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

- Define bad set $B := \{ (c, d) \in P \mid c \mod m = d \mod m \}$.

  $\iff$ from picture: $\frac{|B|}{|P|} \leq \frac{1}{m}$.

$Pr_{a,b \sim U(\mathbb{F}_p^* \times \mathbb{F}_p)} [h(x) = h(y)] = Pr_{a,b}[((a \times x) \oplus b) \mod m = ((a \times y) \oplus b) \mod m]$

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
Proof that \( \mathcal{H}_{p,m}^{\text{lin}} := \{ x \mapsto ((a \times x) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p \} \) 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 \( c = (a \times x) \oplus b \) and \( d = (a \times y) \oplus b \). The mapping \((a, b) \mapsto (c, d)\) is a bijection (for every \( x \neq y \)) from \( \mathbb{F}_p^* \times \mathbb{F}_p \rightarrow P \).

- Define the bad set \( B := \{(c, d) \in P \mid c \mod m = d \mod m \} \).

\[ \Pr_{a, b \sim \mathcal{U}(\mathbb{F}_p^* \times \mathbb{F}_p)}[h(x) = h(y)] = \Pr_{a, b}[((a \times x) \oplus b) \mod m = ((a \times y) \oplus b) \mod m] \]

\[ = \Pr_{a, b}[c \mod m = d \mod m] = \Pr_{a, b}[(c, d) \in B] \]

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

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

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

- Define $c = (a \times x) \oplus b$ and $d = (a \times y) \oplus b$.
  
- The mapping $(a, b) \mapsto (c, d)$ is a bijection (for every $x \neq y$) from $\mathbb{F}_p^* \times \mathbb{F}_p \to P$.

- Define bad set $B := \{(c, d) \in P \mid c \mod m = d \mod m\}$.

\[
\Pr_{a,b \sim \mathcal{U}((\mathbb{F}_p^* \times \mathbb{F}_p))}[h(x) = h(y)] = \Pr_{a,b}[(a \times x) \oplus b) \mod m = ((a \times y) \oplus b) \mod m] = \Pr_{a,b}[c \mod m = d \mod m] = \Pr_{a,b}[(c, d) \in B] = \Pr_{c,d \sim \mathcal{U}(P)}[(c, d) \in B]
\]
Proof that $\mathcal{H}_{p,m}^{\text{lin}} := \{x \mapsto ((a \times x) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}$ 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 $c = (a \times x) \oplus b$, $d = (a \times y) \oplus b$ $\iff$ $(c, d) = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$.

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

- Define bad set $B := \{(c, d) \in P \mid c \mod m = d \mod m\}$.

\[ \Pr_{a, b \sim \mathcal{U}(\mathbb{F}_p^* \times \mathbb{F}_p)}[h(x) = h(y)] = \Pr_{a, b}[((a \times x) \oplus b) \mod m = ((a \times y) \oplus b) \mod m] \]

\[ = \Pr_{a, b}[c \mod m = d \mod m] = \Pr_{a, b}[(c, d) \in B] = \frac{|B|}{|P|} \]
Proof that $\mathcal{H}_{p,m}^{\text{lin}} := \{ x \mapsto ((a \times x) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p \}$ 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 $c = (a \times x) \oplus b$, $d = (a \times y) \oplus b \iff \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$.

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

- Define bad set $B := \{(c, d) \in P \mid c \mod m = d \mod m\}$.

  $\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) \mod m = ((a \times y) \oplus b) \mod m]

  = \Pr_{a, b}[c \mod m = d \mod 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}$. □
**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.**

\[ \forall S \subseteq [D], \forall x \in D : \quad \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[ 1 + |\{y \in S \mid h(y) = h(x)\}| \right] = \ldots = 2 + \sum_{y \in S \{x\}} c \alpha \]

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

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

**Use Case 2:** Linear Probing  

**Conclusion**  

**References**
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.

$\forall S \subseteq [D], \forall x \in D : \quad \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[ 1 + \left| \{ y \in S \mid h(y) = h(x) \} \right| \right]$

$= \ldots = 2 + \sum_{y \in S \setminus \{x\}} \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(y) = h(x)]$
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 $T_{\text{chaining}}(n, m, \mathcal{H}) \leq 2 + c\alpha = O(1)$ if $\alpha \in O(1)$ and $c \in O(1)$.

**Proof: Mostly the same.**

\[
\forall S \subseteq [D], \forall x \in D : \quad \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \left[ 1 + |\{y \in S \mid h(y) = h(x)\}| \right] \\
\quad = \ldots = 2 + \sum_{y \in S \setminus \{x\}} \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(y) = h(x)] \\
\quad = 2 + \sum_{y \in S \setminus \{x\}} \frac{c}{m} \leq 2 + \frac{cn}{m} = 2 + c\alpha. \quad \Box
\]
Examples for Universal Hash Families

- "\((ax + b) \mod p\) \mod m" is 1-universal

  as discussed: \(D = \mathbb{F}_p\), \(R = [m]\),
  \(H_{p,m}^{\text{lin}} := \{x \mapsto ((a \times b) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}\)
Examples for Universal Hash Families

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

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

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

- “\((ax \mod p) \mod m\)” is only 2-universal:

  \(D = \mathbb{F}_p\), \(R = [m]\),

  \(\mathcal{H} := \{x \mapsto (a \times b) \mod m \mid a \in \mathbb{F}_p^*\}\)
Examples for Universal Hash Families

- \(((ax + b) \mod p) \mod m\) is 1-universal
  
  as discussed: \(D = \mathbb{F}_p\), \(R = [m]\),
  
  \(H^\text{lin}_{p,m} := \{x \mapsto ((a \times b) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}\)

- \((ax \mod p) \mod m\) is only 2-universal:

  \(D = \mathbb{F}_p\), \(R = [m]\),
  
  \(H = \{x \mapsto (a \times b) \mod m \mid a \in \mathbb{F}_p^*\}\)

- **Multiply-Shift** is 2-universal:

  \(D = \{0, \ldots, 2^w - 1\}\), \(R = \{0, \ldots, 2^\ell - 1\}\)

  \(H = \{x \mapsto \left\lfloor \left((a \cdot x + b) \mod 2^w\right)/2^{w-\ell} \right\rfloor \mid \text{odd } a \in \{1, \ldots, 2^w - 1\}, b \in \{0, \ldots, 2^w - 1\}\} \)
More Universal Families

Examples for Universal Hash Families

- "((ax + b) \mod p) \mod m" is 1-universal

  as discussed: \(D = \mathbb{F}_p\), \(R = [m]\),
  
  \(H_{p,m}^{\text{lin}} := \{x \mapsto ((a \times b) \oplus b) \mod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}\)

- "(ax \mod p) \mod m" is only 2-universal:

  \(D = \mathbb{F}_p\), \(R = [m]\),
  
  \(H = \{x \mapsto (a \times b) \mod m \mid a \in \mathbb{F}_p^*\}\)

- **Multiply-Shift** is 2-universal:

  \(D = \{0, \ldots, 2^w - 1\}\), \(R = \{0, \ldots, 2^\ell - 1\}\)
  
  \(H = \{x \mapsto \lfloor((a \cdot x + b) \mod 2^w)/2^{w-\ell}\rfloor \mid \)
  
  \(\text{odd } a \in \{1, \ldots, 2^w - 1\}, b \in \{0, \ldots, 2^w - 1\}\}\)

  Selling point of multiply shift:

  - "\(x \mod 2^w\)" drops some higher order bits
  - "\(\lfloor x/2^w-\ell \rfloor\) drops some lower order bits
  - No division or modulo operation needed!

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

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

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. Conclusion

<table>
<thead>
<tr>
<th>Conceptions: What is a Hash Function?</th>
<th>Use Case 1: Hash Table with Chaining</th>
<th>Use Case 2: Linear Probing</th>
<th>Conclusion</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WS 2023/2024  Stefan Walzer, Maximilian Katzmann: Classic Hash Tables

ITI, Algorithm Engineering & Scalable Algorithms
Hash Table with Linear Probing

Let $S : \text{set of } n \text{ keys}$

$m : \text{# of buckets}$

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

**Operations**

For key $x$ probe buckets $h(x), h(x) + 1, h(x) + 2, \ldots \mod m$.

**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.

**Running Times**

- $\text{Lookup}(x \in S)$: At most $x$’s insertion time.
- $\text{Lookup}(x \not\in S)$: At most the time it would take to insert $x$ now.
- $\text{Delete}(x \in S)$: At most the time it would take to insert $y \not\in S$ with $h(y) = h(x)$.

It suffices to understand insertion times!

**Theorem:** Linear Probing under SUHA

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

$\mathbb{E}[T_{n,m}] = O\left(\frac{1}{\left(1 - \alpha_{\text{max}}\right)^2}\right) = O\left(\frac{1}{1}\right)$.
Hash Table with Linear Probing

$S$ : set of $n$ keys
$m$ : # of buckets
$\alpha = \frac{n}{m}$
Hash Table with Linear Probing

$S$ : set of $n$ keys
$m$ : # of buckets
$\alpha = \frac{n}{m}$

Operations

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

Insert. Put $x$ into first empty bucket.
Hash Table with Linear Probing

\[ S : \text{set of } n \text{ keys} \]
\[ m : \# \text{ of buckets} \]
\[ \alpha = \frac{n}{m} \]

Operations

For key \( x \) probe buckets \( h(x), h(x) + 1, h(x) + 2, \ldots \) (mod \( m \)).

Insert. Put \( x \) into first empty bucket.
**Hash Table with Linear Probing**

- $S$: set of $n$ keys
- $m$: # of buckets
- $\alpha = \frac{n}{m}$

**Operations**

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

Insert. Put $x$ into first empty bucket.
Hash Table with Linear Probing

\[ S : \text{set of } n \text{ keys} \]
\[ m : \# \text{ of buckets} \]
\[ \alpha = n/m \]

**Operations**

For key \( x \) probe buckets \( h(x), h(x) + 1, h(x) + 2, \ldots \) (mod \( m \)).

Insert. Put \( x \) into first empty bucket.
Hash Table with Linear Probing

\[ S : \text{set of } n \text{ keys} \]
\[ m : \# \text{ of buckets} \]
\[ \alpha = \frac{n}{m} \]

**Operations**

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

Insert. Put \( x \) into first empty bucket.
Hash Table with Linear Probing

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

**Operations**

For key $x$ probe buckets $h(x), h(x) + 1, h(x) + 2, \ldots$ (mod $m$).

- **Insert**: Put $x$ into first empty bucket.
- **Lookup**: Look for $x$, abort when encountering empty bucket.

**Running 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 $T_{\alpha}$ be the random insertion time into a linear probing hash table. If $\frac{1}{2} \leq \alpha = n/m < \alpha_{\text{max}}$ for some $\alpha_{\text{max}} < 1$, then under SUHA we have $E[T_{\alpha}] = O\left(\frac{1}{(1 - \alpha_{\text{max}})^2}\right) = O\left(\frac{1}{(1 - \alpha)^2}\right)$.
Hash Table with Linear Probing

\[ S : \text{set of } n \text{ keys} \]
\[ m : \# \text{ of buckets} \]
\[ \alpha = \frac{n}{m} \]

**Operations**

*For key \( x \) probe buckets \( h(x), h(x) + 1, h(x) + 2, \ldots \mod m \).*

- **Insert.** Put \( x \) into first empty bucket.
- **Lookup.** Look for \( x \), abort when encountering empty bucket.
- **Delete.** Lookup and remove \( x \) and \( \triangle \) check if a key to the right wants to move into the hole.

\[ \rightarrow \text{For details see https://en.wikipedia.org/wiki/Linear_probing} \]
Hash Table with Linear Probing

Operations

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

- **Insert.** Put $x$ into first empty bucket.
- **Lookup.** Look for $x$, abort when encountering empty bucket.
- **Delete.** Lookup and remove $x$ and $\triangle$ check if a key to the right wants to move into the hole.

$\Rightarrow$ For details see [https://en.wikipedia.org/wiki/Linear_probing](https://en.wikipedia.org/wiki/Linear_probing)

Running Times

- **Lookup** ($x \in S$): At most $x$’s insertion time.

Conceptions: What is a Hash Function?

- Use Case 1: Hash Table with Chaining
- Use Case 2: Linear Probing

Conclusion

References

22/36  WS 2023/2024  Stefan Walzer, Maximilian Katzmann: Classic Hash Tables
ITI, Algorithm Engineering & Scalable Algorithms
Hash Table with Linear Probing

Operations

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

- 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.

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

Running 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.

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
Hash Table with Linear Probing

Operations

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

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.

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

Running 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 \( h(y) = h(x) \).

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

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References

ITI, Algorithm Engineering & Scalable Algorithms
Hash Table with Linear Probing

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

Operations

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

- Insert. Put $x$ into first empty bucket.
- Lookup. Look for $x$, abort when encountering empty bucket.
- Delete. Lookup and remove $x$ and $\triangle$ check if a key to the right wants to move into the hole.

$\triangleright$ For details see https://en.wikipedia.org/wiki/Linear_probing

Running 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 $h(y) = h(x)$.

$\triangleright$ It suffices to understand insertion times!

Theorem: Linear Probing under SUHA

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

$$E[T_{n,m}] = O\left(\frac{1}{\left(1 - \alpha_{\text{max}}\right)^2}\right) = O\left(\frac{1}{\alpha}\right).$$

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References

ITI, Algorithm Engineering & Scalable Algorithms
Hash Table with Linear Probing

Operations

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

- **Insert.** Put $x$ into first empty bucket.
- **Lookup.** Look for $x$, abort when encountering empty bucket.
- **Delete.** Lookup and remove $x$ and ⌊ probe check if a key to the right wants to move into the hole.

$S : \text{set of } n \text{ keys}$
$m : \# \text{ of buckets}$
$\alpha = \frac{n}{m}$

Running 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 $h(y) = h(x)$.

$\Rightarrow$ It suffices to understand insertion times!

Theorem: Linear Probing under SUHA

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

$$\mathbb{E}[T_{n,m}] = O(1).$$
Hash Table with Linear Probing

Operations

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

- **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.

$S : \text{set of } n \text{ keys}$
$m : \# \text{ of buckets}$
$\alpha = \frac{n}{m}$

Running 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 $h(y) = h(x)$.

$\iff$ It suffices to understand insertion times!

Theorem: Linear Probing under SUHA

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

$$E[T_{n,m}] = \mathcal{O}\left(\frac{1}{(1-\alpha_{\text{max}})^2}\right) = \mathcal{O}(1).$$

(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. Conclusion
Chernoff

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

Lemma:

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

$\Pr[h \sim U(RD) [X \geq k] \leq \exp(-\frac{(1-\alpha)2k}{3}).$

Proof

Let $S = \{x_1, \ldots, x_n\}$ and $X_i = 1$ if $h(x_i) \in \{1, \ldots, k\}$ with $\mathbb{E}[X] = knm = \alpha k$. Then

$X = \sum_{i \in [n]} X_i \sim Bin(n, km)$ with $\mathbb{E}[X] = \alpha k$.

$\Pr[X \geq k] = \Pr[X \geq 1 \alpha \mathbb{E}[X]] \leq \exp(-\varepsilon^2\mathbb{E}[X]/3) \leq \exp(-\frac{(1-\alpha)2k}{3}).$ (using $\frac{1}{2} \leq \alpha \leq 1$)

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
**Preparation: A concentration bound**

**Chernoff**

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

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

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

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

---

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

**Use Case 2: Linear Probing**

**Conclusion**

**References**

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Stefan Walzer, Maximilian Katzmann: Classic Hash Tables

ITI, Algorithm Engineering & Scalable Algorithms
Preparation: A concentration bound

Chernoff

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

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

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

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

Proof

Let $S = \{x_1, \ldots, x_n\}$ and $X_i = \mathbb{I}_{\{h(x_i) \in \{1, \ldots, k\}\}} \sim \text{Ber}(\frac{k}{m})$.

Then $X = \sum_{i \in [n]} X_i \sim \text{Bin}(n, \frac{k}{m})$ with $E[X] = \frac{kn}{m} = \alpha k$.
**Preparation: A concentration bound**

**Chernoff**

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

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

Let $k \in \mathbb{N}$ and $X = |\{y \in S \mid h(y) \in \{1, \ldots, k\}\}|$. Then $\Pr_{h \sim \mathcal{U}(R^n)}[X \geq k] \leq \exp(-(1 - \alpha)^2k/3)$.  

**Proof**

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

\[
\Pr[X \geq k] = \Pr[X \geq \frac{1}{\alpha}\mathbb{E}[X]] \\
= \Pr[X \geq (1 + \frac{1-\alpha}{\alpha})\mathbb{E}[X]] \\
\leq \exp(-(\frac{1-\alpha}{\alpha})^2\alpha k/3) \\
\leq \exp(-(1 - \alpha)^2k/3). \quad \text{(using } \frac{1}{2} \leq \alpha \leq 1)\
\]
Proof: Expected LP-Insertion Time under SUHA is $O(1)$

$E[T]$

Reasoning:

(1) Union Bound.

(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) Chernoff argument from previous slide.

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References

WS 2023/2024

Stefan Walzer, Maximilian Katzmann: Classic Hash Tables

ITI, Algorithm Engineering & Scalable Algorithms
Proof: Expected LP-Insertion Time under SUHA is $\mathcal{O}(1)$

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

Wolfram Alpha gives:

$Z_{\infty} 0 k^2 \exp(- (1 - \alpha_{\text{max}})^2 k/3) = \frac{54}{1 - \alpha_{\text{max}}^6}$.

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References

ITI, Algorithm Engineering & Scalable Algorithms
Proof: Expected LP-Insertion Time under SUHA is $O(1)$

$$\mathbb{E}[T] \leq \mathbb{E}[B] = \sum_{k \geq 1} k \cdot \Pr[B = k]$$
Proof: Expected LP-Insertion Time under SUHA is $O(1)$

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

Reasoning:

1. Union Bound.
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. Chernoff argument from previous slide.

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
Proof: Expected LP-Insertion Time under SUHA is $O(1)$

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

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

Reasoning:

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

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

\[
\leq \sum_{k \geq 1} k \cdot \sum_{d=0}^{k-1} \Pr\left[ A_{h(x) - d, h(x) - d + k - 1} \right] \overset{(2)}{=} \sum_{k \geq 1} k \cdot k \cdot \Pr[A_1, k]
\]

Reasoning:

(1) Union Bound.
(2) \( h(x) \) is independent of keys in the table and hash distribution is invariant under cyclic shifts.

Conceptions: What is a Hash Function?
Use Case 1: Hash Table with Chaining
Use Case 2: Linear Probing
Conclusion
References

ITI, Algorithm Engineering & Scalable Algorithms
Proof: Expected LP-Insertion Time under SUHA is $O(1)$

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

(1) Union Bound.

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

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

Reasoning:

(1) Union Bound.

(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.
Proof: Expected LP-Insertion Time under SUHA is $O(1)$

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

(1) Union Bound. 

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

(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) Chernoff argument from previous slide.

Reasoning:

(1) Union Bound. 

(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) Chernoff argument from previous slide.
Proof: Expected LP-Insertion Time under SUHA is $O(1)$

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

\begin{align*}
&\leq \sum_{k \geq 1} k \cdot \sum_{d=0}^{k-1} \Pr[A_{h(x)-d,h(x)-d+k-1}] \\
&\stackrel{(1)}{=} \sum_{k \geq 1} k \cdot k \cdot \Pr[A_{1,k}] \\
&\leq \sum_{k \geq 1} k^2 \cdot \Pr[\{y \in S \mid h(y) \in \{1, \ldots, k\}\} \geq k] \\
&\stackrel{(3)}{=} \sum_{k \geq 1} k^2 \cdot \Pr[|\{y \in S \mid h(y) \in \{1, \ldots, k\}\}| \geq k] \\
&\leq \sum_{k \geq 1} k^2 \cdot \exp(- (1 - \alpha)^2 k / 3) \\
&\leq \sum_{k \geq 1} k^2 \cdot \exp(- (1 - \alpha_{\text{max}})^2 k / 3) \\
\end{align*}

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References

Use Case 2: Linear Probing

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
Proof: Expected LP-Insertion Time under SUHA is \( O(1) \)

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

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

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

\[
\leq \sum_{k \geq 1} k^2 \cdot \exp(- (1 - \alpha)^2 k / 3)
\]

\[
\leq \sum_{k \geq 1} k^2 \cdot \exp(- (1 - \alpha_{\text{max}})^2 k / 3) = O(1).
\]

Wolfram Alpha gives: \[
\int_0^\infty k^2 \exp(- (1 - \alpha_{\text{max}})^2 k / 3) = \frac{54}{(1 - \alpha_{\text{max}})^6}.
\]

Reasoning:

(1) Union Bound.

(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) Chernoff argument from previous slide.

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References

WS 2023/2024 Stefan Walzer, Maximilian Katzmann: Classic Hash Tables

ITI, Algorithm Engineering & Scalable Algorithms
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. Conclusion
A family $\mathcal{E}$ of events is independent if $\forall k \in \mathbb{N}$ and distinct $E_1, \ldots, E_k \in \mathcal{E}$ 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 $X_1, \ldots, X_k \in \mathcal{X}$ and all $x_1, \ldots, x_k \in \mathbb{R}$ 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 $\mathcal{E}$ of events is pairwise independent if for distinct $E_1, E_2 \in \mathcal{E}$ we have

$$\Pr[E_1 \cap E_2] = \Pr[E_1] \cdot \Pr[E_2].$$

A family $\mathcal{X}$ of discrete random variables is pairwise independent if for all distinct $X_1, X_2 \in \mathcal{X}$ and all $x_1, x_2 \in \mathbb{R}$ we have

$$\Pr[X_1 = x_1 \land X_2 = x_2] = \Pr[X_1 = x_1] \cdot \Pr[X_2 = x_2].$$
### $d$-wise Independence

A family $\mathcal{E}$ of events is **$d$-wise independent** if $\forall k \in \{2, \ldots, d\}$ and distinct $E_1, \ldots, E_k \in \mathcal{E}$ 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 **$d$-wise independent** if $\forall k \in \{2, \ldots, d\}$, distinct $X_1, \ldots, X_k \in \mathcal{X}$ and all $x_1, \ldots, x_k \in \mathbb{R}$ we have

$$\Pr \left[ \bigwedge_{i=1}^{k} X_i = x_i \right] = \prod_{i=1}^{k} \Pr[X_i = x_i].$$
**d-Independent Hash Family**

**Definition:** A family \( \mathcal{H} \subseteq [R]^D \) of hash functions is \( d \)-independent if for distinct \( x_1, \ldots, x_d \in D \) and any \( i_1, \ldots, i_d \in R \):

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

---

Conceptions: What is a Hash Function?  Use Case 1: Hash Table with Chaining  Use Case 2: Linear Probing  Conclusion  References

---

30/36  WS 2023/2024  Stefan Walzer, Maximilian Katzmann: Classic Hash Tables  ITI, Algorithm Engineering & Scalable Algorithms
**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 $x_1, \ldots, x_d \in D$ and any $i_1, \ldots, i_d \in R$:

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

**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 $x \in D$. 

---

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

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

**Use Case 2: Linear Probing**

**Conclusion**

**References**
**Definition: d-Independent Hash Family**

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

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

**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 $x \in D$.

**Theorem**

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

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

is a d-independent family.

Note: $\mathcal{H} \subseteq \mathbb{F}^D \rightsquigarrow$ not yet useful.
**Definition: $d$-Independent Hash Family**

A family $\mathcal{H} \subseteq [R]^D$ of hash functions is $d$-independent if for distinct $x_1, \ldots, x_d \in D$ and any $i_1, \ldots, i_d \in R$:

$$\Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x_1) = i_1 \land \ldots \land h(x_d) = i_d] = \prod_{j=1}^{d} \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x_j) = i_j] = |R|^{-d}.$$ (grey is implied by black)

**Theorem**

Let $D = R = F$ be a finite field. Then

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

is a $d$-independent family.

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

**Corollary: Smaller Ranges (proof omitted)**

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

**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 $x \in D$.

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

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

Let $x_1, \ldots, x_d \in \mathbb{F}$ be distinct keys and $i_1, \ldots, i_d \in \mathbb{F}$ arbitrary. To show:

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

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

$$h(x_1) = i_1 \quad h(x_2) = i_2 \quad \ldots \quad h(x_d) = i_d \iff a_0 + a_1 x_1 + \cdots + a_{d-1} x_{d-1} = i_1 \quad a_0 + a_1 x_2 + \cdots + a_{d-1} x_{d-1} = i_2 \quad \ldots \quad a_0 + a_1 x_d + \cdots + a_{d-1} x_{d-1} = i_d.$$

Given the Vandermonde matrix $M$, exactly one vector $\vec{a} = M^{-1} \cdot \vec{i}$ solves the equation. Therefore:

$$\Pr_{h \sim U(\mathcal{H})} \left[ \forall j \in [d] : h(x_j) = i_j \right] = \Pr_{a_0, \ldots, a_{d-1} \sim U(\mathbb{F})} \left[ \vec{a} = M^{-1} \cdot \vec{i} \right] = |\mathbb{F}| - d.$$
Proof: \( \mathcal{H} := \{ x \mapsto \sum_{i=0}^{d-1} a_i x^i \mid a_0, \ldots, a_{d-1} \in \mathbb{F} \} \) is \( d \)-independent

Let \( x_1, \ldots, x_d \in \mathbb{F} \) be distinct keys and \( i_1, \ldots, i_d \in \mathbb{F} \) arbitrary.

\[ \leadsto \text{to show: } \Pr_{h \sim \mathcal{U}(\mathcal{H})} [ \forall j \in [d] : h(x_j) = i_j ] = |\mathbb{F}|^{-d}. \]
Proof: $\mathcal{H} := \{ x \mapsto \sum_{i=0}^{d-1} a_i x^i \mid a_0, \ldots, a_{d-1} \in \mathbb{F} \}$ is $d$-independent

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

$\leftarrow$ 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 $a_0, \ldots, a_{d-1}$) the following is equivalent:

$$
\begin{align*}
  h(x_1) &= i_1 & a_0 + a_1 x_1 + \cdots + a_{d-1} x_1^{d-1} &= i_1 \\
  h(x_2) &= i_2 & a_0 + a_1 x_2 + \cdots + a_{d-1} x_2^{d-1} &= i_2 \\
  &\vdots \quad \iff \quad & \vdots \\
  h(x_d) &= i_d & a_0 + a_1 x_d + \cdots + a_{d-1} x_d^{d-1} &= i_d
\end{align*}
$$

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

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

Let \( x_1, \ldots, x_d \in \mathbb{F} \) be distinct keys and \( i_1, \ldots, i_d \in \mathbb{F} \) arbitrary.

\( \iff \) 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 \( a_0, \ldots, a_{d-1} \)) the following is equivalent:

\[
\begin{align*}
    h(x_1) &= i_1, & a_0 + a_1 x_1 + \cdots + a_{d-1} x_1^{d-1} &= i_1 \\
    h(x_2) &= i_2, & a_0 + a_1 x_2 + \cdots + a_{d-1} x_2^{d-1} &= i_2 \\
    \vdots & \quad \iff \quad \vdots \\
    h(x_d) &= i_d, & a_0 + a_1 x_d + \cdots + a_{d-1} x_d^{d-1} &= i_d
\end{align*}
\]

\[
\begin{pmatrix}
    1 & x_1 & x_1^2 & \cdots & x_1^{d-1} \\
    1 & x_2 & x_2^2 & \cdots & x_2^{d-1} \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    1 & x_d & x_d^2 & \cdots & x_d^{d-1}
\end{pmatrix} \cdot \begin{pmatrix}
    a_0 \\
    a_1 \\
    \vdots \\
    a_{d-1}
\end{pmatrix} = \begin{pmatrix}
    i_1 \\
    i_2 \\
    \vdots \\
    i_d
\end{pmatrix}
\]

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

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

Let \( x_1, \ldots, x_d \in \mathbb{F} \) be distinct keys and \( i_1, \ldots, i_d \in \mathbb{F} \) arbitrary.

\( \iff \) 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 \( a_0, \ldots, a_{d-1} \)) the following is equivalent:

\[
\begin{align*}
  h(x_1) = i_1 & \quad \iff \quad a_0 + a_1 x_1 + \cdots + a_{d-1} x_1^{d-1} = i_1 \\
  h(x_2) = i_2 & \quad \iff \quad a_0 + a_1 x_2 + \cdots + a_{d-1} x_2^{d-1} = i_2 \\
  \vdots & \quad \vdots \\
  h(x_d) = i_d & \quad \iff \quad a_0 + a_1 x_d + \cdots + a_{d-1} x_d^{d-1} = i_d
\end{align*}
\]

\[
\begin{pmatrix}
  1 & x_1 & x_1^2 & \cdots & x_1^{d-1} \\
  1 & x_2 & x_2^2 & \cdots & x_2^{d-1} \\
  \vdots & \vdots & \ddots & \vdots & \vdots \\
  1 & x_d & x_d^2 & \cdots & x_d^{d-1}
\end{pmatrix}
\cdot
\begin{pmatrix}
  a_0 \\
  a_1 \\
  \vdots \\
  a_{d-1}
\end{pmatrix}
= \begin{pmatrix}
  i_1 \\
  i_2 \\
  \vdots \\
  i_d
\end{pmatrix}
\]

Vandermonde matrix \( M \Rightarrow \) regular

Conceptions: What is a Hash Function?
Use Case 1: Hash Table with Chaining
Use Case 2: Linear Probing
Conclusion
References

31/36  WS 2023/2024  Stefan Walzer, Maximilian Katzmann: Classic Hash Tables
ITI, Algorithm Engineering & Scalable Algorithms
Proof: $\mathcal{H} := \{ x \mapsto \sum_{i=0}^{d-1} a_i x^i \mid a_0, \ldots, a_{d-1} \in \mathbb{F} \}$ is $d$-independent

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

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

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

$$
\begin{align*}
   h(x_1) &= i_1 & a_0 + a_1 x_1 + \cdots + a_{d-1} x_1^{d-1} &= i_1 \\
   h(x_2) &= i_2 & a_0 + a_1 x_2 + \cdots + a_{d-1} x_2^{d-1} &= i_2 \\
   \vdots & & \vdots \\
   h(x_d) &= i_d & a_0 + a_1 x_d + \cdots + a_{d-1} x_d^{d-1} &= i_d
\end{align*}
$$

$\iff$

$$
\begin{bmatrix}
   1 & x_1 & x_1^2 & \cdots & x_1^{d-1} \\
   1 & x_2 & x_2^2 & \cdots & x_2^{d-1} \\
   \vdots & \vdots & \vdots & \ddots & \vdots \\
   1 & x_d & x_d^2 & \cdots & x_d^{d-1}
\end{bmatrix}
\begin{bmatrix}
   a_0 \\
   a_1 \\
   \vdots \\
   a_{d-1}
\end{bmatrix}
= 
\begin{bmatrix}
   i_1 \\
   i_2 \\
   \vdots \\
   i_d
\end{bmatrix}
$$

$Vandermonde matrix \quad M \Rightarrow \quad \text{regular}$

Exactly one vector $\vec{a} = M^{-1} \cdot \vec{i}$ solves the equation.
Proof: $\mathcal{H} := \{ x \mapsto \sum_{i=0}^{d-1} a_i x^i \mid a_0, \ldots, a_{d-1} \in \mathbb{F} \}$ is $d$-independent

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

$\iff$ 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 $a_0, \ldots, a_{d-1}$) the following is equivalent:

$h(x_1) = i_1$ \quad $a_0 + a_1 x_1 + \cdots + a_{d-1} x_1^{d-1} = i_1$
$h(x_2) = i_2$ \quad $a_0 + a_1 x_2 + \cdots + a_{d-1} x_2^{d-1} = i_2$
\quad \quad \quad \quad $\iff$
$h(x_d) = i_d$ \quad $a_0 + a_1 x_d + \cdots + a_{d-1} x_d^{d-1} = i_d$

\[
\begin{pmatrix}
1 & x_1 & x_1^2 & \cdots & x_1^{d-1} \\
1 & x_2 & x_2^2 & \cdots & x_2^{d-1} \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_d & x_d^2 & \cdots & x_d^{d-1}
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
\vdots \\
a_{d-1}
\end{pmatrix}
= 
\begin{pmatrix}
i_1 \\
i_2 \\
\vdots \\
i_d
\end{pmatrix}
\]

Vandermonde matrix $M \Rightarrow$ regular

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

$\Rightarrow \Pr_{h \sim \mathcal{U}(\mathcal{H})}[\forall j : h(x_j) = i_j] = \Pr_{a_0, \ldots, a_{d-1} \sim \mathcal{U}(\mathbb{F})}[\vec{a} = M^{-1} \cdot \vec{i}] = |\mathbb{F}|^{-d}$. \qed
Concentration Bound for $d$-Independent Variables

(Tricky) Exercise

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

$$\Pr[X - \mathbb{E}[X] \geq \varepsilon \mathbb{E}[X]] = O(\varepsilon^{-d} \mathbb{E}[X]^{-d/2}).$$
Concentration Bound for $d$-Independent Variables

(Tricky) Exercise

Let $d$ be even and $X_1, \ldots, X_n \sim \text{Ber}(p)$ a $d$-independent family of random variables with $p = \Omega(1/n)$.

Let $X = \sum_{i=1}^{n} X_i$. Then for any $\varepsilon > 0$ we have

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

Remark: Weaker than Chernoff, stronger than Chebyshev

Chebycheff gives $\Pr[X - \mathbb{E}[X] \geq \varepsilon \mathbb{E}[X]] \leq \frac{1-p}{\varepsilon^2 \mathbb{E}[X]}$. (requires $d = 2$)

Chernoff gave $\Pr[X - \mathbb{E}[X] \geq \varepsilon \mathbb{E}[X]] \leq \exp(-\varepsilon^2 \mathbb{E}[X]/3)$. (requires $d = n$).
Lemma (last slide)

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

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

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 $X = |\{y \in S \mid h(y) \in \{1, \ldots, k\}\}|$. Then $\Pr[X \geq k] \leq O((1 - \alpha)^{-d}k^{-d/2})$. 

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
Lemma (last slide)

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

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 \( X = |\{y \in S \mid h(y) \in \{1, \ldots, k\}\}|. \)

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

Proof

Let \( S = \{x_1, \ldots, x_n\} \) and \( X_i = 1_{\{h(x_i) \in \{1, \ldots, k\}\}} \sim Ber(\frac{k}{m}). \)

Then \( X = \sum_{i \in [n]} X_i \) fits the Lemma with \( \mathbb{E}[X] = \frac{kn}{m} = \alpha k. \)
Preparation: A Concentration Bound again for \(d\)-independence

**Lemma (last slide)**

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

**Lemma: \(\geq k\) hits in segment of length \(k\)**

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

**Proof**

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

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

Conceptions: What is a Hash Function? Use Case 1: Hash Table with Chaining Use Case 2: Linear Probing Conclusion References
Theorem: Linear Probing with \( \phi \)-independence

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

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

Reasoning:

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

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

Proof Sketch

$$E[T]$$

Reasoning:

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

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

Proof Sketch

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

Reasoning:

Conceptions: What is a Hash Function?
Use Case 1: Hash Table with Chaining
Use Case 2: Linear Probing
Conclusion
References
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:

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

Proof Sketch

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

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!)
**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:

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

**Proof Sketch**

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

1. $\sum_{k \geq 1} k^2 \cdot \Pr[\{|y \in S | h(y) \in \{1, \ldots, k\}| \geq k]\]

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

**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 $d = 8$. 

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References

34/36  WS 2023/2024  Stefan Walzer, Maximilian Katzmann: Classic Hash Tables

ITI, Algorithm Engineering & Scalable Algorithms
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:

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

Proof Sketch

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

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

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

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

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 $d = 8$. 

Conceptions: What is a Hash Function?

Use Case 1: Hash Table with Chaining

Use Case 2: Linear Probing

Conclusion

References
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

\begin{align*}
\mathbb{E}[T] &\leq \mathbb{E}[B] \leq \ldots \\
&\leq \sum_{k \geq 1} k^2 \cdot \Pr[|\{y \in S \mid h(y) \in \{1, \ldots, k\}\}| \geq k] \\
&\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}) \\
&= \frac{\pi^2}{6} \mathcal{O}((1 - \alpha)^{-8}) = \mathcal{O}(1). \quad \square
\end{align*}

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 $O\left(\frac{1}{(1-\alpha)^2}\right)$.
  $\Rightarrow$ A. Pagh, R. Pagh, and Ruzic 2011

- An (artificially bad) 4-independent family gives $\Omega(\log n)$.
  $\Rightarrow$ Pătrașcu and Thorup 2016

- A (well-designed) 4-independent family gives $O\left(\frac{1}{(1-\alpha)^2}\right)$.
  $\Rightarrow$ Pătrăscu and Thorup 2013
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 $O(1)$ per operation.
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 $O(1)$ per operation.

Non-Technical Takeaway: Approaches to analyse hashing based algorithms

- High performance hash function (fast, not analysable)
- Algorithm or data structure using hashing
- Analysis using SUHA
- Hash function from universal class (possibly fast)

- Can use
- Justifies & deepens understanding of
- Rigorously justifies
- Requires & deepens understanding of

Analysis using universal hashing builds on...
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 $O(1)$ per operation.

Non-Technical Takeaway: Approaches to analyse hashing based algorithms

- High performance hash function (fast, not analysable)
- Algorithm or data structure using hashing
- Hash function from universal class (possibly fast)

We’ll always use SUHA in the following. Less probability theory, more 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?*

*Conceptions: What is a Hash Function? Use Case 1: Hash Table with Chaining Use Case 2: Linear Probing Conclusion References
Anhang: Mögliche Prüfungsfragen II

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?*
References


