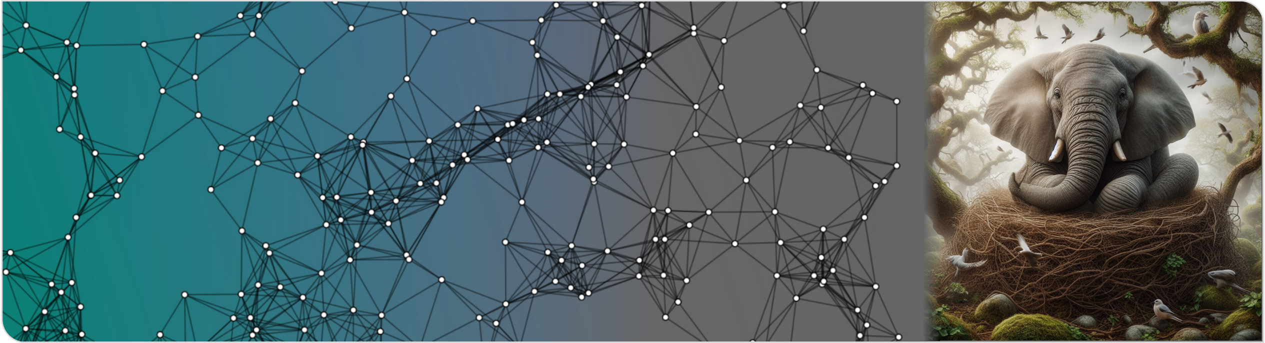
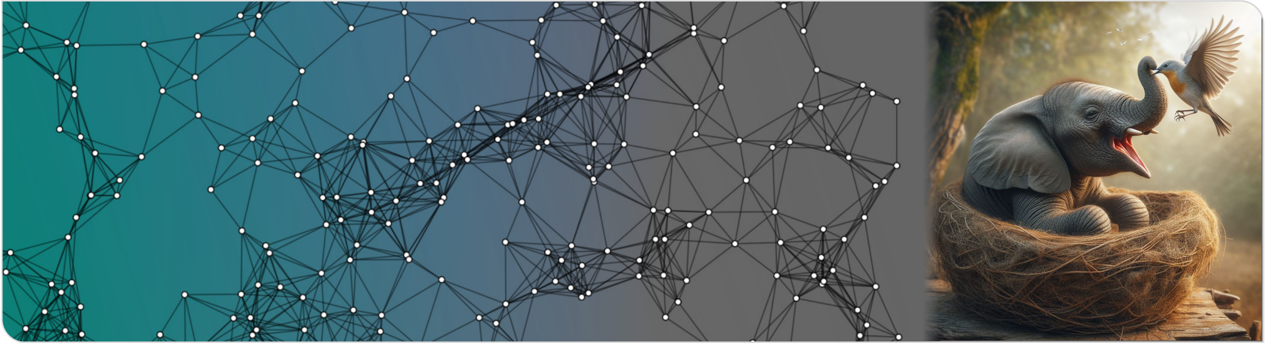
Probability and Computing – Cuckoo Hashing

Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Probability and Computing – Cuckoo Hashing

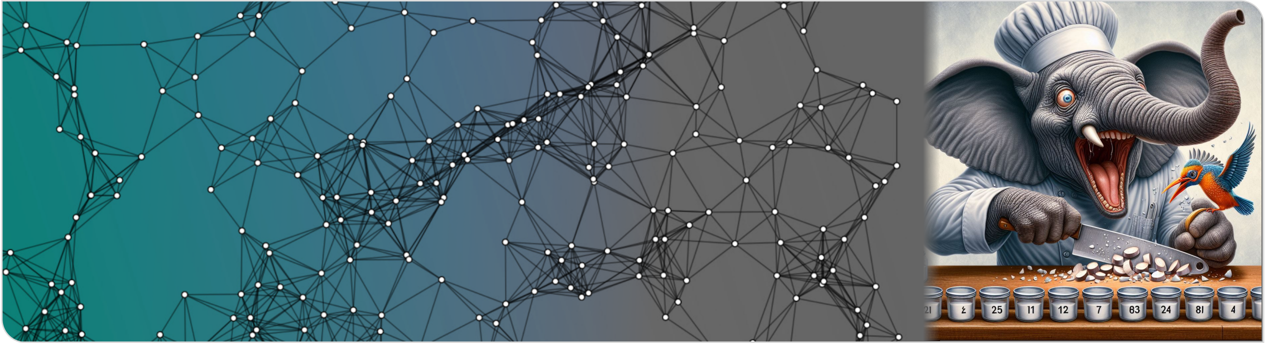
Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Karlsruhe Institute of Technology

Probability and Computing – Cuckoo Hashing

Stefan Walzer, Maximilian Katzmann | WS 2023/2024

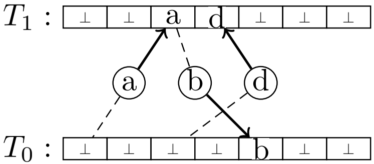


Content

Cuckoo Hashing

Setup

$S \subseteq D$	key set of size n
T_0, T_1	two tables of size m
$h_0, h_1 \sim \mathcal{U}([m]^D)$	two hash functions (SUHA)
$\frac{n}{m} = 1 - \beta$	for some $\beta > 0$
	(⚠ load factor $\alpha = \frac{n}{2m}$)



Algorithm lookup(x)**:**
- **return** $x \in \{T_0[h_0(x)], T_1[h_1(x)]\}$

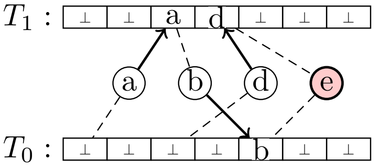
Algorithm delete(x)**:**
- **if** $T_0[h_0(x)] = x$ **then**
  - $T_0[h_0(x)] \leftarrow \perp$
- **else if** $T_1[h_1(x)] = x$ **then**
  - $T_1[h_1(x)] \leftarrow \perp$

Algorithm insert(x)**:**
- **for** $i = 0$ **to** LIMIT **do**
  - $b \leftarrow i \bmod 2$
  - swap(x, $T_b[h_b(x)]$)
  - **if** $x = \perp$ **then**
    - **return** SUCCESS
- **return** FAILURE

Cuckoo Hashing

Setup

$S \subseteq D$	key set of size n
T_0, T_1	two tables of size m
$h_0, h_1 \sim \mathcal{U}([m]^D)$	two hash functions (SUHA)
$\frac{n}{m} = 1 - \beta$	for some $\beta > 0$
	(⚠ load factor $\alpha = \frac{n}{2m}$)



Algorithm lookup(x)**:**

  return $x \in \{T_0[h_0(x)], T_1[h_1(x)]\}$

Algorithm delete(x)**:**

  if $T_0[h_0(x)] = x$ **then**

    $T_0[h_0(x)] \leftarrow \perp$

  else if $T_1[h_1(x)] = x$ **then**

    $T_1[h_1(x)] \leftarrow \perp$

Algorithm insert(x)**:**

  for $i = 0$ **to** LIMIT **do**

    $b \leftarrow i \bmod 2$

    swap(x, $T_b[h_b(x)]$)

    if $x = \perp$ **then**

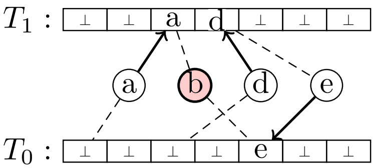
      return SUCCESS

  return FAILURE

Cuckoo Hashing

Setup

$S \subseteq D$	key set of size n
T_0, T_1	two tables of size m
$h_0, h_1 \sim \mathcal{U}([m]^D)$	two hash functions (SUHA)
$\frac{n}{m} = 1 - \beta$	for some $\beta > 0$
	(⚠ load factor $\alpha = \frac{n}{2m}$)



Algorithm lookup(x)**:**
- **return** $x \in \{T_0[h_0(x)], T_1[h_1(x)]\}$

Algorithm delete(x)**:**

```
if T_0[h_0(x)] = x then
    T_0[h_0(x)] ← ⊥
else if T_1[h_1(x)] = x then
    T_1[h_1(x)] ← ⊥
```

Algorithm insert(x)**:**

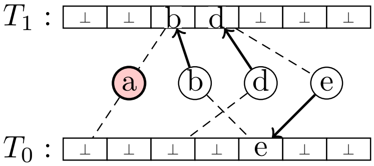
```
for i = 0 to LIMIT do
    b ← i mod 2
    swap(x, T_b[h_b(x)])
    if x = ⊥ then
        return SUCCESS
return FAILURE
```

Cuckoo Hashing

Karlsruhe Institute of Technology

Setup

$S \subseteq D$	key set of size n
T_0, T_1	two tables of size m
$h_0, h_1 \sim \mathcal{U}([m]^D)$	two hash functions (SUHA)
$\frac{n}{m} = 1 - \beta$	for some $\beta > 0$
	(⚠ load factor $\alpha = \frac{n}{2m}$)



Algorithm lookup(x)**:**
- **return** $x \in \{T_0[h_0(x)], T_1[h_1(x)]\}$

Algorithm delete(x)**:**
- **if** $T_0[h_0(x)] = x$ **then**
  - $T_0[h_0(x)] \leftarrow \perp$
- **else if** $T_1[h_1(x)] = x$ **then**
  - $T_1[h_1(x)] \leftarrow \perp$

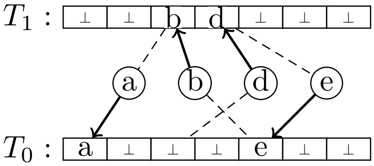
Algorithm insert(x)**:**
- **for** $i = 0$ **to** LIMIT **do**
  - $b \leftarrow i \bmod 2$
  - swap(x, $T_b[h_b(x)]$)
  - **if** $x = \perp$ **then**
    - **return** SUCCESS
- **return** FAILURE

Cuckoo Hashing

Karlsruhe Institute of Technology

Setup

$S \subseteq D$	key set of size n
T_0, T_1	two tables of size m
$h_0, h_1 \sim \mathcal{U}([m]^D)$	two hash functions (SUHA)
$\frac{n}{m} = 1 - \beta$	for some $\beta > 0$
	(⚠ load factor $\alpha = \frac{n}{2m}$)



Algorithm lookup(x)**:**
- **return** $x \in \{T_0[h_0(x)], T_1[h_1(x)]\}$

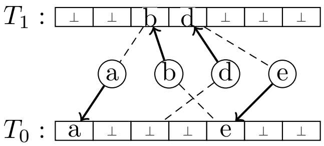
Algorithm delete(x)**:**
- **if** $T_0[h_0(x)] = x$ **then**
  - $T_0[h_0(x)] \leftarrow \perp$
- **else if** $T_1[h_1(x)] = x$ **then**
  - $T_1[h_1(x)] \leftarrow \perp$

Algorithm insert(x)**:**
- **for** $i = 0$ **to** LIMIT **do**
  - $b \leftarrow i \bmod 2$
  - swap(x, $T_b[h_b(x)]$)
  - **if** $x = \perp$ **then**
    - **return** SUCCESS
- **return** FAILURE

Cuckoo Hashing Theorem

Algorithm insert(x):

for $i = 0$ **to** LIMIT **do**
  $b \leftarrow i \bmod 2$
  swap(x, $T_b[h_b(x)]$)
  if $x = \perp$ **then**
    return SUCCESS
return FAILURE



Theorem (Analysis with LIMIT $= \infty$)

Assume we insert all $x \in S$ and then another key y. Let E be the event that this succeeds and

$$T = \begin{cases} \text{insertion time of } y & \text{if } E \text{ occurs} \\ 0 & \textit{otherwise} \end{cases}$$

Then i $\Pr[E] = 1 - \mathcal{O}(1/m)$ and ii $\mathbb{E}[T] = \mathcal{O}(1)$.

Theorem (full analysis, not here)

If we

- set LIMIT $= \Omega(\log n)$ appropriately
- rebuild the table with fresh hash functions when LIMIT is reached

we obtain a hash table where lookup and delete take $\mathcal{O}(1)$ time and insert takes *expected* $\mathcal{O}(1)$ time.

Proof of i: Success probability is $1 - \mathcal{O}(1/m)$

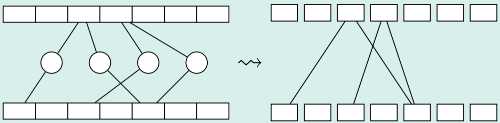
Karlsruhe Institute of Technology

The Cuckoo Graph

Consider the bipartite *cuckoo graph*

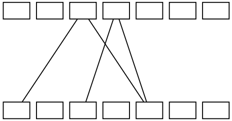
$$G = ([m], [m], \{(h_0(x), h_1(x)) \mid x \in S\}$$

the key x corresponds to the edge $(h_0(x), h_1(x))$ and each table position to a vertex.



Proof of i: Success probability is $1 - \mathcal{O}(1/m)$

Karlsruhe Institute of Technology



Keys and buckets in the infinite loop

Assume $\bar{E}$ occurs, i.e. an insertion fails due to an infinite loop. Let $G^* = (V^*, E^*)$ be the subgraph of G with

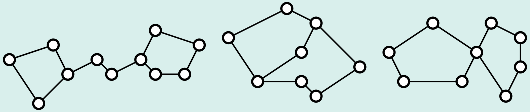
- V^*: table positions touched infinitely often
- E^*: keys touched infinitely often.

Properties of G^*:

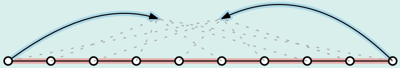
- connected
- $|E^*| = |V^*| + 1$ can you see why?
- $\deg_{E^*}(v) \geq 2$.

Possibilities for G^*

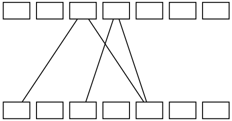
There are three options:



In all three cases: Simple path through $|V^*|$ and two extra edges connecting inwards:



Proof of i: Success probability is $1 - \mathcal{O}(1/m)$



Keys and buckets in the infinite loop

Assume $\bar{E}$ occurs, i.e. an insertion fails due to an infinite loop. Let $G^* = (V^*, E^*)$ be the subgraph of G with

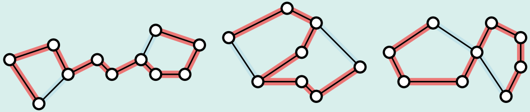
- V^*: table positions touched infinitely often
- E^*: keys touched infinitely often.

Properties of G^*:

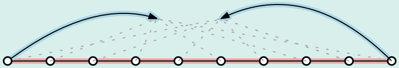
- connected
- $|E^*| = |V^*| + 1$ can you see why?
- $\deg_{E^*}(v) \geq 2$.

Possibilities for G^*

There are three options:



In all three cases: Simple path through $|V^*|$ and two extra edges connecting inwards:

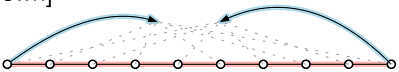


Proof of $\boxed{\text{i}}$: Success probability is $1 - \mathcal{O}(1/m)$

$$\Pr[\bar{E}] = \Pr[\exists \text{path as shown}]$$

$$= \Pr[\exists k \in \mathbb{N} : \exists x_0, \ldots, x_{k+1} \in S : x_0, \ldots, x_{k+1} \text{ form a path as shown}]$$

$$\overset{\text{union bound}}{\leq} \sum_{k=1}^{n} \sum_{x_0,\ldots,x_{k+1} \in S} \Pr[x_0, \ldots, x_{k+1} \text{ form a path as shown}]$$



$$\leq \sum_{k=1}^{n} \underbrace{n^{k+2}}_{\text{a}} \cdot \underbrace{2}_{\text{b}} \cdot \underbrace{\tfrac{1}{m^{k+1}}}_{\text{c}} \cdot \underbrace{\left(\tfrac{k+1}{2m}\right)^2}_{\text{d}}$$

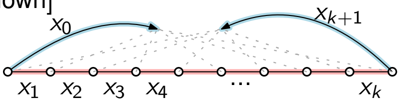
$$\leq \frac{1}{2} \sum_{k=1}^{n} m^{k+2-k-1-2}(1-\beta)^{k+2}(k+1)^2$$

$$\leq \frac{1}{2m} \sum_{k=1}^{\infty} (1-\beta)^{k+2}(k+1)^2 = \frac{1}{m} \cdot \mathcal{O}\left(\frac{1}{\beta^3}\right) = \frac{1}{m} \cdot \mathcal{O}(1) \quad \square$$

- a Choose sequence of $k+2$ keys.
- b Choose to start in left or right table.
- c Neighbouring keys share a hash.
- d Two bordering keys connect back inward.

Proof of i: Success probability is $1 - \mathcal{O}(1/m)$

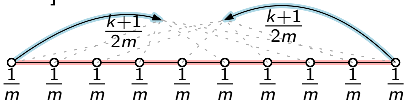
$$\Pr[\bar{E}] = \Pr[\exists \text{path as shown}]$$
$$= \Pr[\exists k \in \mathbb{N} : \exists x_0, \dots, x_{k+1} \in S : x_0, \dots, x_{k+1} \text{ form a path as shown}]$$
$$\overset{\text{union bound}}{\leq} \sum_{k=1}^{n} \sum_{x_0,\dots,x_{k+1} \in S} \Pr[x_0, \dots, x_{k+1} \text{ form a path as shown}]$$
$$\leq \sum_{k=1}^{n} \underbrace{n^{k+2}}_{a} \cdot \underbrace{2}_{b} \cdot \underbrace{\frac{1}{m^{k+1}}}_{c} \cdot \underbrace{\left(\frac{k+1}{2m}\right)^2}_{d}$$
$$\leq \frac{1}{2} \sum_{k=1}^{n} m^{k+2-k-1-2}(1-\beta)^{k+2}(k+1)^2$$
$$\leq \frac{1}{2m} \sum_{k=1}^{\infty} (1-\beta)^{k+2}(k+1)^2 = \frac{1}{m} \cdot \mathcal{O}\left(\frac{1}{\beta^3}\right) = \frac{1}{m} \cdot \mathcal{O}(1) \quad \square$$



- a Choose sequence of $k+2$ keys.
- b Choose to start in left or right table.
- c Neighbouring keys share a hash.
- d Two bordering keys connect back inward.

Proof of i: Success probability is $1 - \mathcal{O}(1/m)$

$$
\begin{aligned}
\Pr[\bar{E}] &= \Pr[\exists \text{path as shown}] \\
&= \Pr[\exists k \in \mathbb{N} : \exists x_0, \ldots, x_{k+1} \in S : x_0, \ldots, x_{k+1} \text{ form a path as shown}] \\
&\overset{\text{union bound}}{\leq} \sum_{k=1}^{n} \sum_{x_0,\ldots,x_{k+1} \in S} \Pr[x_0, \ldots, x_{k+1} \text{ form a path as shown}] \\
&\leq \sum_{k=1}^{n} \underbrace{n^{k+2}}_{a} \cdot \underbrace{2}_{b} \cdot \underbrace{\tfrac{1}{m^{k+1}}}_{c} \cdot \underbrace{\left(\tfrac{k+1}{2m}\right)^2}_{d} \\
&\leq \tfrac{1}{2} \sum_{k=1}^{n} m^{k+2-k-1-2} (1-\beta)^{k+2} (k+1)^2 \\
&\leq \tfrac{1}{2m} \sum_{k=1}^{\infty} (1-\beta)^{k+2} (k+1)^2 = \tfrac{1}{m} \cdot \mathcal{O}\left(\tfrac{1}{\beta^3}\right) = \tfrac{1}{m} \cdot \mathcal{O}(1) \quad \square
\end{aligned}
$$



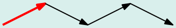
- a Choose sequence of $k+2$ keys.
- b Choose to start in left or right table.
- c Neighbouring keys share a hash.
- d Two bordering keys connect back inward.

Proof of ii: Expected insertion time is $\mathcal{O}(1)$

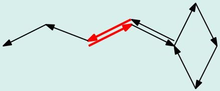
Lemma

If the insertion of y takes $t \in \mathbb{N}$ steps then the cuckoo graph G contained (previously) a path of length $\lceil (t-2)/3 \rceil$ starting from $h_0(y)$ or from $h_1(y)$.

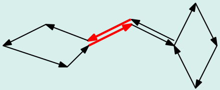
Proof.



no turning back
$\rightsquigarrow$ path of length $t-1$
starting from $h_0(y)$



turn back once
$\rightsquigarrow$ path of length $\lceil (t-2)/3 \rceil$
starting from $h_0(y)$ or $h_1(y)$



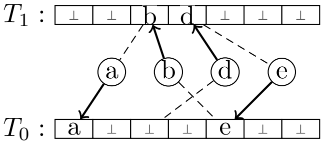
turn back twice
impossible: insertion would fail

$\square$

Proof of ii: Expected insertion time is $\mathcal{O}(1)$ (continued)

$$\begin{aligned}
\mathbb{E}[T] &= \sum_{t\geq 1} \Pr[T \geq t] && \text{(see complexity classes slide 10)}\\
&\leq \sum_{t\geq 1} \Pr[\exists \text{path of length } \lceil (t-2)/3 \rceil \text{ starting from } h_0(y) \text{ or } h_1(y)] && \text{by Lemma}\\
&\leq 2 \cdot \sum_{t\geq 1} \Pr[\exists \text{path of length } \lceil (t-2)/3 \rceil \text{ starting from } h_0(y)] && \text{union bound + symmetry}\\
&\leq 2\left(2 + 3 \cdot \sum_{t\geq 1} \Pr[\exists \text{path of length } t \text{ starting from } h_0(y)]\right) && \sum_{i\geq 1} f(\lceil t/3 \rceil) = 3 \cdot f(1) + 3 \cdot f(2) + \ldots\\
&\leq 4 + 6 \cdot \sum_{t\geq 1} \sum_{x_1,\ldots,x_t \in S} \Pr[x_1, \ldots, x_t \text{ form path starting from } h_0(y)] && \text{union bound}\\
&\leq 4 + 6 \cdot \sum_{t\geq 1} n^t m^{-t} = 6 \sum_{t\geq 0} (1-\beta)^t = \mathcal{O}(1/\beta) = \mathcal{O}(1). \quad \square
\end{aligned}$$

Conclusion



Cuckoo Hashing

- hash table with *worst case* constant access times
- analysis considers path in graphs similar to the Erdős-Renyi model
- many variations and spin-offs (not discussed here)

Anhang: Mögliche Prüfungsfragen I

Karlsruhe Institute of Technology

- Was ist und was kann Cuckoo Hashing?
  - Was ist die Grundidee? Wie funktionieren die Operationen?
  - Worauf ist bei der Wahl der Tabellengröße / beim Load Factor zu achten?
  - Was kann man über die Laufzeit der Operationen sagen?
  - Welche Vorteile und Nachteile ergeben sich im Vergleich zu anderen Techniken wie linearem Sondieren?
- Analyse:
  - Eine Einfügung, die fehlschlägt, entspricht gewissen Strukturen im Cuckoo-Graphen. Welchen?
  - Wie haben wir gezeigt, dass solche Strukturen unwahrscheinlich sind?
  - Wie haben wir die erwartete Einfügezeit abgeschätzt?