Universal Hashing

Definition 29

Let $d \in \mathbb{N}$; $q \geq (d+1)n$ be a prime; and let $\bar{a} \in \{0, \ldots, q-1\}^{d+1}$. Define for $x \in \{0, \ldots, q-1\}$

$$h_{\bar{a}}(x) := \Big(\sum_{i=0}^{d} a_i x^i \bmod q\Big) \bmod n \enspace .$$

Let $\mathcal{H}_n^d := \{h_{\bar{a}} \mid \bar{a} \in \{0, \ldots, q-1\}^{d+1}\}$. The class $\mathcal{H}_n^d$ is $(e, d+1)$-independent.

Note that in the previous case we had $d = 1$ and chose $a_d \neq 0$.

Universal Hashing

For the coefficients $\bar{a} \in \{0, \ldots, q-1\}^{d+1}$ let $f_{\bar{a}}$ denote the polynomial

$$f_{\bar{a}}(x) = \Big(\sum_{i=0}^{d} a_i x^i\Big) \bmod q$$

The polynomial is defined by $d+1$ distinct points.

Universal Hashing

For the coefficients $\bar{a} \in \{0, \ldots, q-1\}^{d+1}$ let $f_{\bar{a}}$ denote the polynomial

$$f_{\bar{a}}(x) = \Big(\sum_{i=0}^{d} a_i x^i \Big) \bmod q$$

The polynomial is defined by $d+1$ distinct points.

Universal Hashing

For the coefficients $\bar{a} \in \{0, \dots, q-1\}^{d+1}$ let $f_{\bar{a}}$ denote the polynomial

$$f_{\bar{a}}(x) = \Big(\sum_{i=0}^{d} a_i x^i\Big) \bmod q$$

The polynomial is defined by $d+1$ distinct points.

Universal Hashing

Fix $\ell \leq d+1$; let $x_1, \dots, x_\ell \in \{0, \dots, q-1\}$ be keys, and let $t_1, \dots, t_\ell$ denote the corresponding hash-function values.

Let $A^\ell = \{h_{\bar{a}} \in \mathcal{H} \mid h_{\bar{a}}(x_i) = t_i$ for all $i \in \{1, \dots, \ell\}\}$
Then

$$h_{\bar{a}} \in A^\ell \Leftrightarrow h_{\bar{a}} = f_{\bar{a}} \bmod n \text{ and}$$

$$f_{\bar{a}}(x_i) \in \underbrace{\{t_i + \alpha \cdot n \mid \alpha \in \{0, \dots, \lceil \tfrac{q}{n} \rceil - 1\}\}}_{=: B_i}$$

In order to obtain the cardinality of A^ℓ we choose our polynomial by fixing $d+1$ points.

We first fix the values for inputs $x_1, \dots, x_\ell$.
We have

$$|B_1| \cdot \ldots \cdot |B_\ell|$$

possibilities to do this (so that $h_{\bar{a}}(x_i) = t_i$).

Universal Hashing

Fix $\ell \leq d+1$; let $x_1, \ldots, x_\ell \in \{0, \ldots, q-1\}$ be keys, and let $t_1, \ldots, t_\ell$ denote the corresponding hash-function values.

Let $A^\ell = \{h_{\bar{a}} \in \mathcal{H} \mid h_{\bar{a}}(x_i) = t_i \text{ for all } i \in \{1, \ldots, \ell\}\}$
Then

$$h_{\bar{a}} \in A^\ell \Leftrightarrow h_{\bar{a}} = f_{\bar{a}} \bmod n \text{ and}$$

$$f_{\bar{a}}(x_i) \in \underbrace{\{t_i + \alpha \cdot n \mid \alpha \in \{0, \ldots, \lceil \tfrac{q}{n} \rceil - 1\}\}}_{=:B_i}$$

In order to obtain the cardinality of A^ℓ we choose our polynomial by fixing $d+1$ points.

We first fix the values for inputs $x_1, \ldots, x_\ell$.
We have

$$|B_1| \cdot \ldots \cdot |B_\ell|$$

possibilities to do this (so that $h_{\bar{a}}(x_i) = t_i$).

Universal Hashing

Fix $\ell \leq d+1$; let $x_1, \dots, x_\ell \in \{0, \dots, q-1\}$ be keys, and let $t_1, \dots, t_\ell$ denote the corresponding hash-function values.

Let $A^\ell = \{h_{\bar{a}} \in \mathcal{H} \mid h_{\bar{a}}(x_i) = t_i \text{ for all } i \in \{1, \dots, \ell\}\}$

Then

$$h_{\bar{a}} \in A^\ell \Leftrightarrow h_{\bar{a}} = f_{\bar{a}} \bmod n \text{ and}$$

$$f_{\bar{a}}(x_i) \in \underbrace{\{t_i + \alpha \cdot n \mid \alpha \in \{0, \dots, \lceil \tfrac{q}{n} \rceil - 1\}\}}_{=: B_i}$$

In order to obtain the cardinality of A^ℓ we choose our polynomial by fixing $d+1$ points.

We first fix the values for inputs $x_1, \dots, x_\ell$.
We have

$$|B_1| \cdot \ldots \cdot |B_\ell|$$

possibilities to do this (so that $h_{\bar{a}}(x_i) = t_i$).

Universal Hashing

Fix $\ell \leq d+1$; let $x_1, \ldots, x_\ell \in \{0, \ldots, q-1\}$ be keys, and let $t_1, \ldots, t_\ell$ denote the corresponding hash-function values.

Let $A^\ell = \{h_{\bar{a}} \in \mathcal{H} \mid h_{\bar{a}}(x_i) = t_i \text{ for all } i \in \{1, \ldots, \ell\}\}$
Then

$$h_{\bar{a}} \in A^\ell \Leftrightarrow h_{\bar{a}} = f_{\bar{a}} \bmod n \text{ and}$$

$$f_{\bar{a}}(x_i) \in \underbrace{\{t_i + \alpha \cdot n \mid \alpha \in \{0, \ldots, \lceil \tfrac{q}{n} \rceil - 1\}\}}_{=: B_i}$$

In order to obtain the cardinality of A^ℓ we choose our polynomial by fixing $d+1$ points.

We first fix the values for inputs $x_1, \ldots, x_\ell$.
We have

$$|B_1| \cdot \ldots \cdot |B_\ell|$$

possibilities to do this (so that $h_{\bar{a}}(x_i) = t_i$).

Universal Hashing

Fix $\ell \leq d+1$; let $x_1, \dots, x_\ell \in \{0, \dots, q-1\}$ be keys, and let $t_1, \dots, t_\ell$ denote the corresponding hash-function values.

Let $A^\ell = \{h_{\bar{a}} \in \mathcal{H} \mid h_{\bar{a}}(x_i) = t_i \text{ for all } i \in \{1, \dots, \ell\}\}$
Then

$$h_{\bar{a}} \in A^\ell \Leftrightarrow h_{\bar{a}} = f_{\bar{a}} \bmod n \text{ and}$$

$$f_{\bar{a}}(x_i) \in \underbrace{\{t_i + \alpha \cdot n \mid \alpha \in \{0, \dots, \lceil \tfrac{q}{n} \rceil - 1\}\}}_{=: B_i}$$

In order to obtain the cardinality of A^ℓ we choose our polynomial by fixing $d+1$ points.

We first fix the values for inputs $x_1, \dots, x_\ell$.
We have

$$|B_1| \cdot \ldots \cdot |B_\ell|$$

possibilities to do this (so that $h_{\bar{a}}(x_i) = t_i$).

Universal Hashing

Fix $\ell \leq d+1$; let $x_1, \ldots, x_\ell \in \{0, \ldots, q-1\}$ be keys, and let $t_1, \ldots, t_\ell$ denote the corresponding hash-function values.

Let $A^\ell = \{h_{\bar{a}} \in \mathcal{H} \mid h_{\bar{a}}(x_i) = t_i \text{ for all } i \in \{1, \ldots, \ell\}\}$
Then

$$h_{\bar{a}} \in A^\ell \Leftrightarrow h_{\bar{a}} = f_{\bar{a}} \bmod n \text{ and}$$

$$f_{\bar{a}}(x_i) \in \underbrace{\{t_i + \alpha \cdot n \mid \alpha \in \{0, \ldots, \lceil \tfrac{q}{n} \rceil - 1\}\}}_{=: B_i}$$

In order to obtain the cardinality of A^ℓ we choose our polynomial by fixing $d+1$ points.

We first fix the values for inputs $x_1, \ldots, x_\ell$.

We have

$$|B_1| \cdot \ldots \cdot |B_\ell|$$

possibilities to do this (so that $h_{\bar{a}}(x_i) = t_i$).

Universal Hashing

Fix $\ell \leq d+1$; let $x_1,\dots,x_\ell \in \{0,\dots,q-1\}$ be keys, and let $t_1,\dots,t_\ell$ denote the corresponding hash-function values.

Let $A^\ell = \{h_{\bar{a}} \in \mathcal{H} \mid h_{\bar{a}}(x_i) = t_i \text{ for all } i \in \{1,\dots,\ell\}\}$
Then

$$h_{\bar{a}} \in A^\ell \Leftrightarrow h_{\bar{a}} = f_{\bar{a}} \bmod n \text{ and}$$

$$f_{\bar{a}}(x_i) \in \underbrace{\{t_i + \alpha \cdot n \mid \alpha \in \{0,\dots,\lceil \tfrac{q}{n} \rceil - 1\}\}}_{=: B_i}$$

In order to obtain the cardinality of A^ℓ we choose our polynomial by fixing $d+1$ points.

We first fix the values for inputs $x_1,\dots,x_\ell$.
We have

$$|B_1| \cdot \ldots \cdot |B_\ell|$$

possibilities to do this (so that $h_{\bar{a}}(x_i) = t_i$).

Universal Hashing

Now, we choose $d-\ell+1$ other inputs and choose their value arbitrarily. We have $q^{d-\ell+1}$ possibilities to do this.

Therefore we have

$$|B_1|\cdot\ldots\cdot|B_\ell|\cdot q^{d-\ell+1}\leq\lceil\frac{q}{n}\rceil^\ell\cdot q^{d-\ell+1}$$

possibilities to choose $\bar{a}$ such that $h_{\bar{a}}\in A_\ell$.

Universal Hashing

Now, we choose $d - \ell + 1$ other inputs and choose their value arbitrarily. We have $q^{d-\ell+1}$ possibilities to do this.

Therefore we have

$$|B_1| \cdot \ldots \cdot |B_\ell| \cdot q^{d-\ell+1} \leq \lceil \frac{q}{n} \rceil^\ell \cdot q^{d-\ell+1}$$

possibilities to choose $\bar{a}$ such that $h_{\bar{a}} \in A_\ell$.

Universal Hashing

Therefore the probability of choosing $h_{\bar{a}}$ from A_ℓ is only

$$\frac{\lceil \frac{q}{n} \rceil^{\ell} \cdot q^{d-\ell+1}}{q^{d+1}}$$

Universal Hashing

Therefore the probability of choosing $h_{\bar{a}}$ from A_ℓ is only

$$\frac{\lceil \frac{q}{n} \rceil^\ell \cdot q^{d-\ell+1}}{q^{d+1}} \le \frac{(\frac{q+n}{n})^\ell}{q^\ell}$$

Universal Hashing

Therefore the probability of choosing $h_{\bar{a}}$ from A_ℓ is only

$$\frac{\lceil \frac{q}{n} \rceil^\ell \cdot q^{d-\ell+1}}{q^{d+1}} \le \frac{(\frac{q+n}{n})^\ell}{q^\ell} \le \Big(\frac{q+n}{q}\Big)^\ell \cdot \frac{1}{n^\ell}$$

Universal Hashing

Therefore the probability of choosing $h_{\bar{a}}$ from A_ℓ is only

$$\frac{\lceil \frac{q}{n} \rceil^\ell \cdot q^{d-\ell+1}}{q^{d+1}} \le \frac{(\frac{q+n}{n})^\ell}{q^\ell} \le \left(\frac{q+n}{q}\right)^\ell \cdot \frac{1}{n^\ell}$$

$$\le \left(1 + \frac{1}{\ell}\right)^\ell \cdot \frac{1}{n^\ell}$$

Universal Hashing

Therefore the probability of choosing $h_{\bar{a}}$ from A_ℓ is only

$$\frac{\lceil \frac{q}{n} \rceil^\ell \cdot q^{d-\ell+1}}{q^{d+1}} \le \frac{(\frac{q+n}{n})^\ell}{q^\ell} \le \left(\frac{q+n}{q}\right)^\ell \cdot \frac{1}{n^\ell}$$

$$\le \left(1 + \frac{1}{\ell}\right)^\ell \cdot \frac{1}{n^\ell} \le \frac{e}{n^\ell} \ .$$

Universal Hashing

Therefore the probability of choosing $h_{\bar{a}}$ from A_ℓ is only

$$\frac{\lceil \frac{q}{n} \rceil^\ell \cdot q^{d-\ell+1}}{q^{d+1}} \le \frac{(\frac{q+n}{n})^\ell}{q^\ell} \le \Big(\frac{q+n}{q}\Big)^\ell \cdot \frac{1}{n^\ell}$$

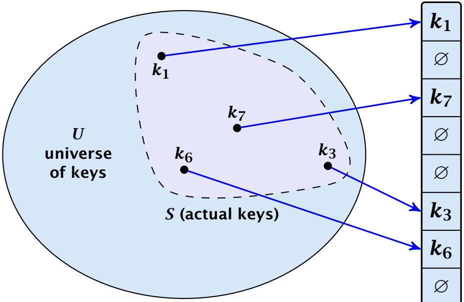
$$\le \Big(1 + \frac{1}{\ell}\Big)^\ell \cdot \frac{1}{n^\ell} \le \frac{e}{n^\ell} \ .$$

This shows that the $\mathcal{H}$ is $(e, d+1)$-universal.

The last step followed from $q \ge (d+1)n$, and $\ell \le d+1$.

Perfect Hashing

Suppose that we **know** the set S of actual keys (no insert/no delete). Then we may want to design a **simple** hash-function that maps all these keys to different memory locations.



Perfect Hashing

Let $m = |S|$. We could simply choose the hash-table size very large so that we don't get any collisions.

Using a universal hash-function the expected number of collisions is

$$\mathrm{E}[\#\text{Collisions}] = \binom{m}{2} \cdot \frac{1}{n} .$$

If we choose $n = m^2$ the expected number of collisions is strictly less than $\frac{1}{2}$.

Can we get an upper bound on the probability of having collisions?

The probability of having 1 or more collisions can be at most $\frac{1}{2}$ as otherwise the expectation would be larger than $\frac{1}{2}$.

Perfect Hashing

Let $m = |S|$. We could simply choose the hash-table size very large so that we don't get any collisions.

Using a universal hash-function the expected number of collisions is

$$\mathrm{E}[\#\text{Collisions}] = \binom{m}{2} \cdot \frac{1}{n} .$$

If we choose $n = m^2$ the expected number of collisions is strictly less than $\frac{1}{2}$.

Can we get an upper bound on the probability of having collisions?

The probability of having 1 or more collisions can be at most $\frac{1}{2}$ as otherwise the expectation would be larger than $\frac{1}{2}$.

Perfect Hashing

Let $m = |S|$. We could simply choose the hash-table size very large so that we don't get any collisions.

Using a universal hash-function the expected number of collisions is

$$\mathrm{E}[\#\text{Collisions}] = \binom{m}{2} \cdot \frac{1}{n} \ .$$

If we choose $n = m^2$ the expected number of collisions is strictly less than $\frac{1}{2}$.

Can we get an upper bound on the probability of having collisions?

The probability of having 1 or more collisions can be at most $\frac{1}{2}$ as otherwise the expectation would be larger than $\frac{1}{2}$.

Perfect Hashing

Let $m = |S|$. We could simply choose the hash-table size very large so that we don't get any collisions.

Using a universal hash-function the expected number of collisions is

$$\mathrm{E}[\#\text{Collisions}] = \binom{m}{2} \cdot \frac{1}{n} \ .$$

If we choose $n = m^2$ the expected number of collisions is strictly less than $\frac{1}{2}$.

Can we get an upper bound on the probability of having collisions?

The probability of having 1 or more collisions can be at most $\frac{1}{2}$ as otherwise the expectation would be larger than $\frac{1}{2}$.

Perfect Hashing

Let $m = |S|$. We could simply choose the hash-table size very large so that we don't get any collisions.

Using a universal hash-function the expected number of collisions is

$$\mathrm{E}[\#\text{Collisions}] = \binom{m}{2} \cdot \frac{1}{n} \ .$$

If we choose $n = m^2$ the expected number of collisions is strictly less than $\frac{1}{2}$.

Can we get an upper bound on the probability of having collisions?

The probability of having 1 or more collisions can be at most $\frac{1}{2}$ as otherwise the expectation would be larger than $\frac{1}{2}$.

Perfect Hashing

Let $m = |S|$. We could simply choose the hash-table size very large so that we don't get any collisions.

Using a universal hash-function the expected number of collisions is

$$\mathrm{E}[\#\text{Collisions}] = \binom{m}{2} \cdot \frac{1}{n} \ .$$

If we choose $n = m^2$ the expected number of collisions is strictly less than $\frac{1}{2}$.

Can we get an upper bound on the probability of having collisions?

The probability of having 1 or more collisions can be at most $\frac{1}{2}$ as otherwise the expectation would be larger than $\frac{1}{2}$.

Perfect Hashing

We can find such a hash-function by a few trials.

However, a hash-table size of $n = m^2$ is very very high.

We construct a two-level scheme. We first use a hash-function that maps elements from S to m buckets.

Let m_j denote the number of items that are hashed to the j-th bucket. For each bucket we choose a second hash-function that maps the elements of the bucket into a table of size m_j^2. The second function can be chosen such that all elements are mapped to different locations.

Perfect Hashing

We can find such a hash-function by a few trials.

However, a hash-table size of $n = m^2$ is very very high.

We construct a two-level scheme. We first use a hash-function that maps elements from S to m buckets.

Let m_j denote the number of items that are hashed to the j-th bucket. For each bucket we choose a second hash-function that maps the elements of the bucket into a table of size m_j^2. The second function can be chosen such that all elements are mapped to different locations.

Perfect Hashing

We can find such a hash-function by a few trials.

However, a hash-table size of $n = m^2$ is very very high.

We construct a two-level scheme. We first use a hash-function that maps elements from S to m buckets.

Let m_j denote the number of items that are hashed to the j-th bucket. For each bucket we choose a second hash-function that maps the elements of the bucket into a table of size m_j^2. The second function can be chosen such that all elements are mapped to different locations.

Perfect Hashing

We can find such a hash-function by a few trials.

However, a hash-table size of $n = m^2$ is very very high.

We construct a two-level scheme. We first use a hash-function that maps elements from S to m buckets.

Let m_j denote the number of items that are hashed to the j-th bucket. For each bucket we choose a second hash-function that maps the elements of the bucket into a table of size m_j^2. The second function can be chosen such that all elements are mapped to different locations.

Perfect Hashing

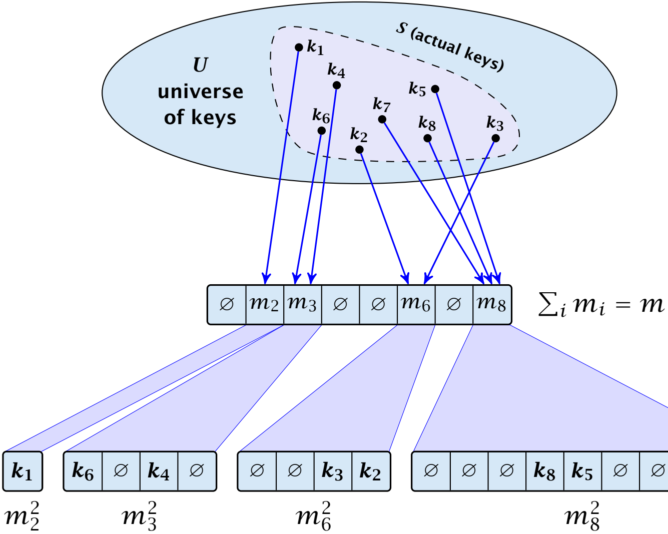
We can find such a hash-function by a few trials.

However, a hash-table size of $n = m^2$ is very very high.

We construct a two-level scheme. We first use a hash-function that maps elements from S to m buckets.

Let m_j denote the number of items that are hashed to the j-th bucket. For each bucket we choose a second hash-function that maps the elements of the bucket into a table of size m_j^2. The second function can be chosen such that all elements are mapped to different locations.

Perfect Hashing



Perfect Hashing

Perfect Hashing

The total memory that is required by all hash-tables is $\mathcal{O}(\sum_j m_j^2)$. Note that m_j is a random variable.

$$\mathrm{E}\Big[\sum_j m_j^2\Big]$$

Perfect Hashing

The total memory that is required by all hash-tables is $\mathcal{O}(\sum_j m_j^2)$. Note that m_j is a random variable.

$$\mathrm{E}\left[\sum_j m_j^2\right] = \mathrm{E}\left[2\sum_j \binom{m_j}{2} + \sum_j m_j\right]$$

Perfect Hashing

The total memory that is required by all hash-tables is $\mathcal{O}(\sum_j m_j^2)$. Note that m_j is a random variable.

$$\begin{aligned}\mathrm{E}\Big[\sum_j m_j^2\Big] &= \mathrm{E}\Big[2\sum_j \binom{m_j}{2} + \sum_j m_j\Big] \\ &= 2\,\mathrm{E}\Big[\sum_j \binom{m_j}{2}\Big] + \mathrm{E}\Big[\sum_j m_j\Big]\end{aligned}$$

Perfect Hashing

The total memory that is required by all hash-tables is $\mathcal{O}(\sum_j m_j^2)$. Note that m_j is a random variable.

$$\begin{aligned}
\mathrm{E}\left[\sum_j m_j^2\right] &= \mathrm{E}\left[2\sum_j \binom{m_j}{2} + \sum_j m_j\right] \\
&= 2\,\mathrm{E}\left[\sum_j \binom{m_j}{2}\right] + \mathrm{E}\left[\sum_j m_j\right]
\end{aligned}$$

The first expectation is simply the expected number of collisions, for the first level. Since we use universal hashing we have

Perfect Hashing

The total memory that is required by all hash-tables is $\mathcal{O}(\sum_j m_j^2)$. Note that m_j is a random variable.

$$
\begin{aligned}
\mathrm{E}\Big[\sum_j m_j^2\Big] &= \mathrm{E}\Big[2\sum_j \binom{m_j}{2} + \sum_j m_j\Big] \\
&= 2\,\mathrm{E}\Big[\sum_j \binom{m_j}{2}\Big] + \mathrm{E}\Big[\sum_j m_j\Big]
\end{aligned}
$$

The first expectation is simply the expected number of collisions, for the first level. Since we use universal hashing we have

$$
= 2\binom{m}{2}\frac{1}{m} + m = 2m - 1\ .
$$

Perfect Hashing

We need only $\mathcal{O}(m)$ time to construct a hash-function h with $\sum_j m_j^2 = \mathcal{O}(4m)$, because with probability at least $1/2$ a random function from a universal family will have this property.

Then we construct a hash-table h_j for every bucket. This takes expected time $\mathcal{O}(m_j)$ for every bucket. A random function h_j is collision-free with probability at least $1/2$. We need $\mathcal{O}(m_j)$ to test this.

We only need that the hash-functions are chosen from a universal family!!!

Cuckoo Hashing

Goal:

Try to generate a hash-table with constant worst-case search time in a dynamic scenario.

- Two hash-tables $T_1[0, \dots, n-1]$ and $T_2[0, \dots, n-1]$, with hash-functions h_1, and h_2.
- An object x is either stored at location $T_1[h_1(x)]$ or $T_2[h_2(x)]$.
- A search clearly takes constant time if the above constraint is met.

Cuckoo Hashing

Goal:
Try to generate a hash-table with constant worst-case search time in a dynamic scenario.

- Two hash-tables $T_1[0,\dots,n-1]$ and $T_2[0,\dots,n-1]$, with hash-functions h_1, and h_2.
- An object x is either stored at location $T_1[h_1(x)]$ or $T_2[h_2(x)]$.
- A search clearly takes constant time if the above constraint is met.

Cuckoo Hashing

Goal:

Try to generate a hash-table with constant worst-case search time in a dynamic scenario.

- Two hash-tables $T_1[0,\ldots,n-1]$ and $T_2[0,\ldots,n-1]$, with hash-functions h_1, and h_2.
- An object x is either stored at location $T_1[h_1(x)]$ or $T_2[h_2(x)]$.
- A search clearly takes constant time if the above constraint is met.

Cuckoo Hashing

Goal:
Try to generate a hash-table with constant worst-case search time in a dynamic scenario.

- Two hash-tables $T_1[0,\ldots,n-1]$ and $T_2[0,\ldots,n-1]$, with hash-functions h_1, and h_2.
- An object x is either stored at location $T_1[h_1(x)]$ or $T_2[h_2(x)]$.
- A search clearly takes constant time if the above constraint is met.

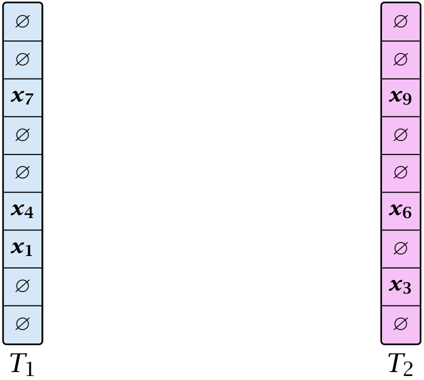
Cuckoo Hashing

Goal:
Try to generate a hash-table with constant worst-case search time in a dynamic scenario.

- Two hash-tables $T_1[0, \dots, n-1]$ and $T_2[0, \dots, n-1]$, with hash-functions h_1, and h_2.
- An object x is either stored at location $T_1[h_1(x)]$ or $T_2[h_2(x)]$.
- A search clearly takes constant time if the above constraint is met.

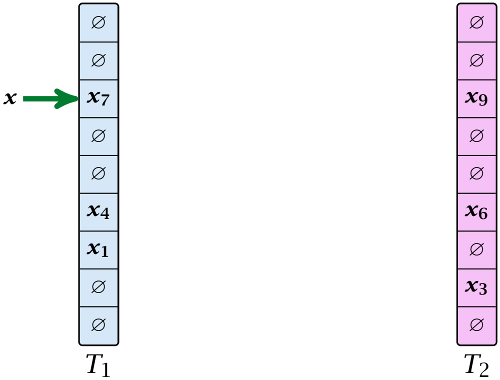
Cuckoo Hashing

Insert:



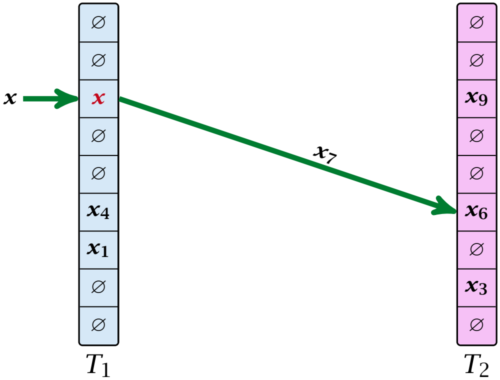
Cuckoo Hashing

Insert:



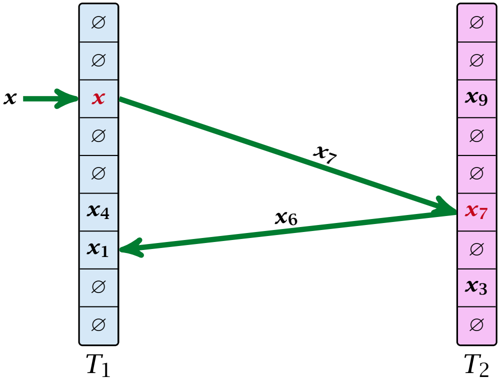
Cuckoo Hashing

Insert:



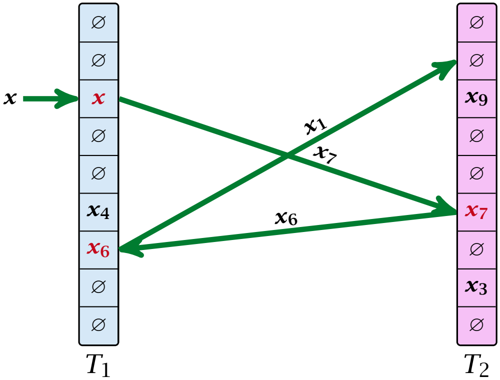
Cuckoo Hashing

Insert:



Cuckoo Hashing

Insert:



Cuckoo Hashing

Algorithm 13 Cuckoo-Insert(x)

if $T_1[h_1(x)] = x \lor T_2[h_2(x)] = x$ **then return**
$\text{steps} \leftarrow 1$
while $\text{steps} \le \text{maxsteps}$ **do**
    exchange x and $T_1[h_1(x)]$
    if $x = \text{null}$ **then return**
    exchange x and $T_2[h_2(x)]$
    if $x = \text{null}$ **then return**
    $\text{steps} \leftarrow \text{steps} + 1$
rehash() // change hash-functions; rehash everything
Cuckoo-Insert(x)

Cuckoo Hashing

- We call one iteration through the while-loop a step of the algorithm.
- We call a sequence of iterations through the while-loop without the termination condition becoming true a phase of the algorithm.
- We say a phase is successful if it is not terminated by the maxstep-condition, but the while loop is left because $x = \text{null}$.

Cuckoo Hashing

- We call one iteration through the while-loop a step of the algorithm.
- We call a sequence of iterations through the while-loop without the termination condition becoming true a phase of the algorithm.
- We say a phase is successful if it is not terminated by the $\mathrm{maxstep}$-condition, but the while loop is left because $x = \mathrm{null}$.

Cuckoo Hashing

- We call one iteration through the while-loop a step of the algorithm.
- We call a sequence of iterations through the while-loop without the termination condition becoming true a phase of the algorithm.
- We say a phase is successful if it is not terminated by the $\mathrm{maxstep}$-condition, but the while loop is left because $x = \mathrm{null}$.

Cuckoo Hashing

What is the expected time for an insert-operation?

We first analyze the probability that we end-up in an infinite loop (that is then terminated after $\mathrm{maxsteps}$ steps).

Formally what is the probability to enter an infinite loop that touches s different keys?

Cuckoo Hashing

What is the expected time for an insert-operation?

We first analyze the probability that we end-up in an infinite loop (that is then terminated after $\mathrm{maxsteps}$ steps).

Formally what is the probability to enter an infinite loop that touches s different keys?

Cuckoo Hashing

What is the expected time for an insert-operation?

We first analyze the probability that we end-up in an infinite loop (that is then terminated after $\mathrm{maxsteps}$ steps).

Formally what is the probability to enter an infinite loop that touches s different keys?

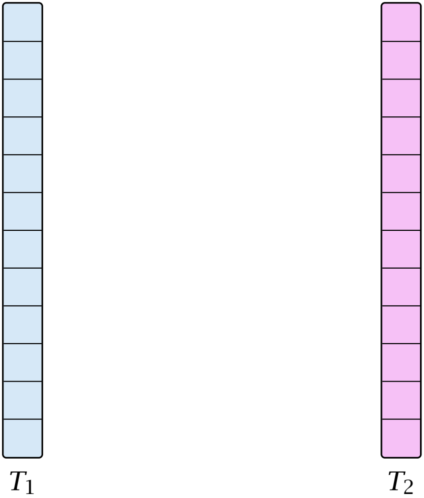
Cuckoo Hashing

What is the expected time for an insert-operation?

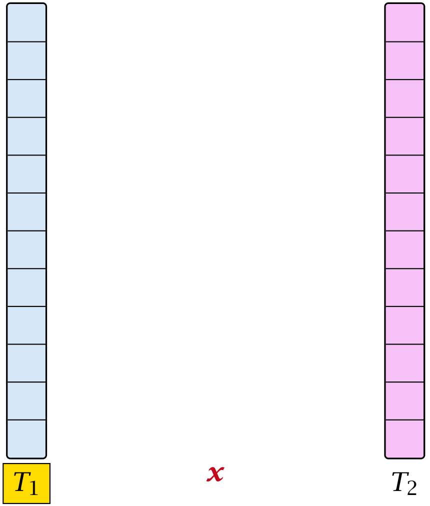
We first analyze the probability that we end-up in an infinite loop (that is then terminated after $\mathrm{maxsteps}$ steps).

Formally what is the probability to enter an infinite loop that touches s different keys?

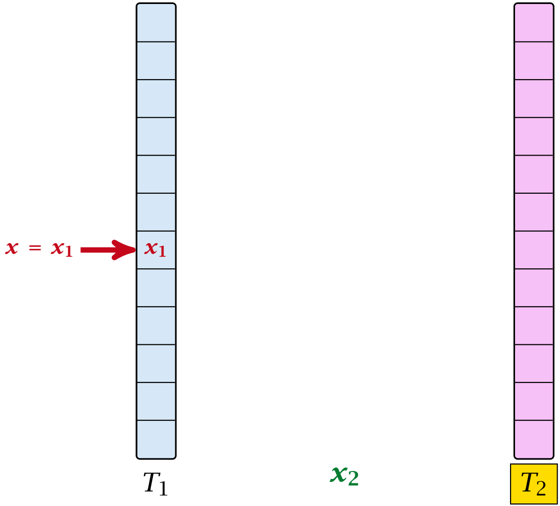
Cuckoo Hashing: Insert



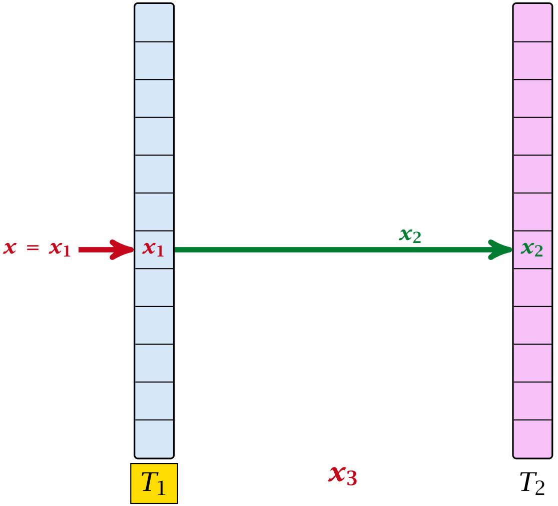
Cuckoo Hashing: Insert



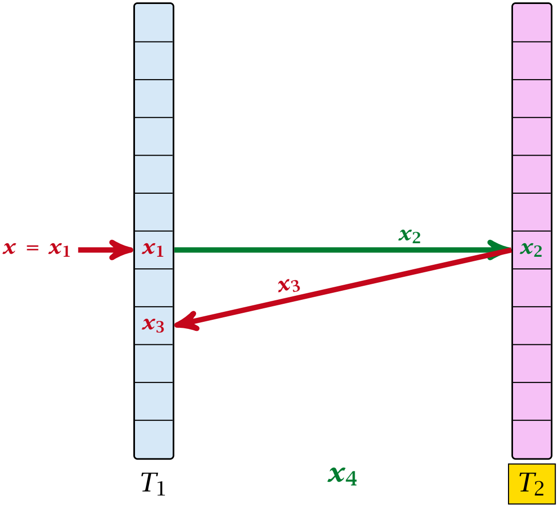
Cuckoo Hashing: Insert



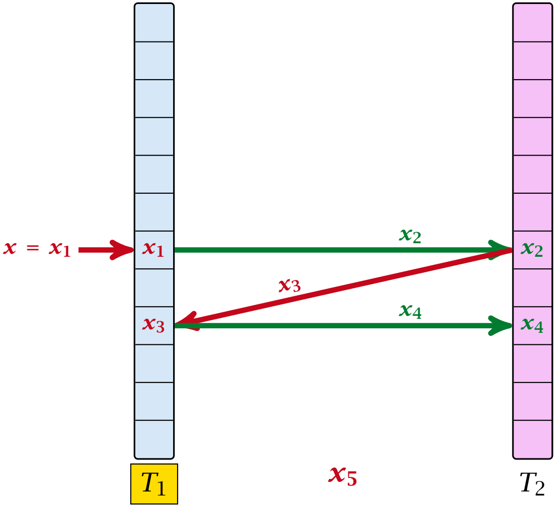
Cuckoo Hashing: Insert



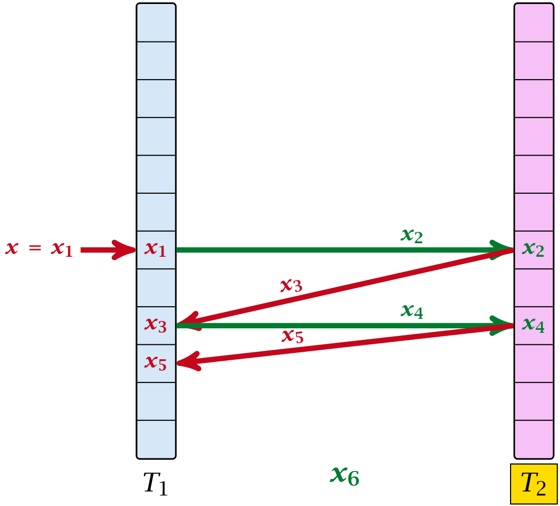
Cuckoo Hashing: Insert



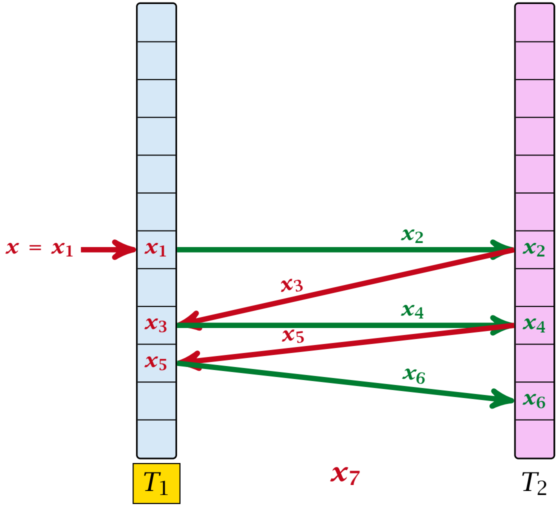
Cuckoo Hashing: Insert



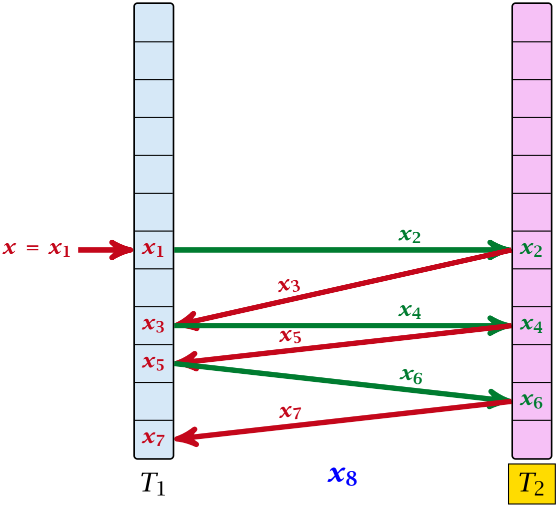
Cuckoo Hashing: Insert



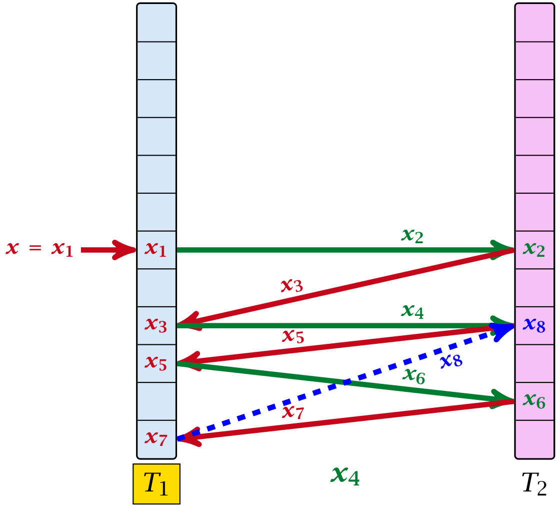
Cuckoo Hashing: Insert



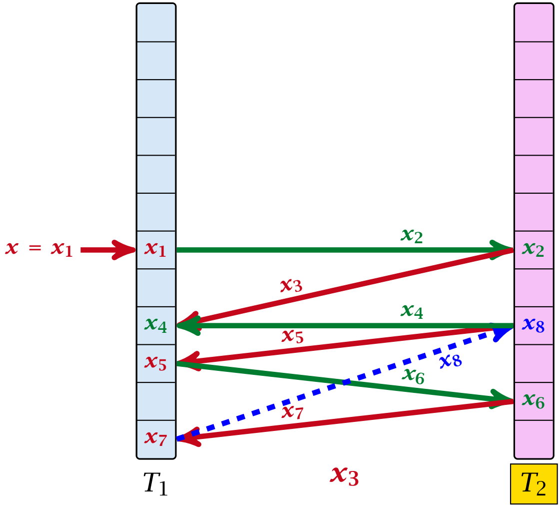
Cuckoo Hashing: Insert



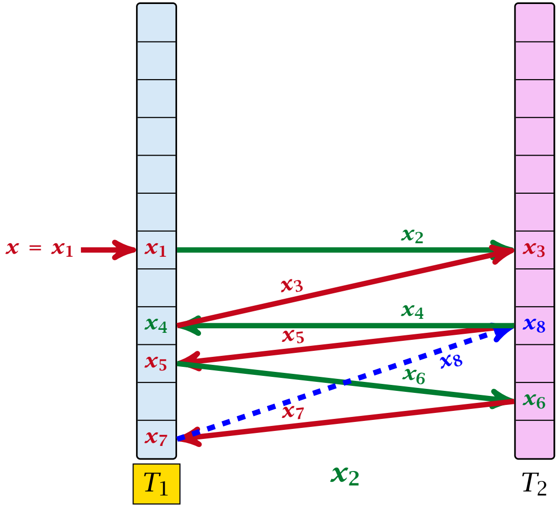
Cuckoo Hashing: Insert



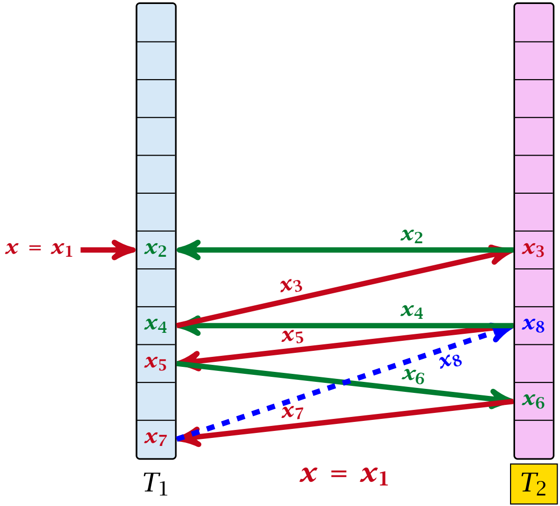
Cuckoo Hashing: Insert



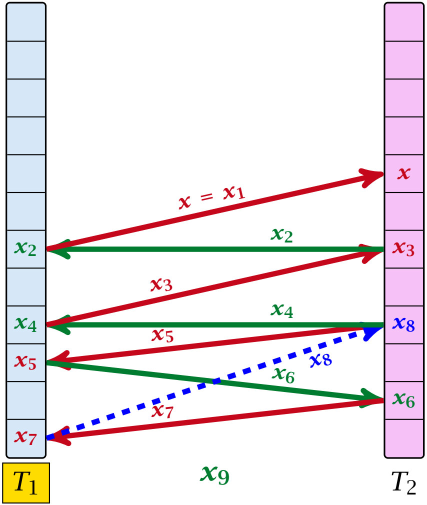
Cuckoo Hashing: Insert



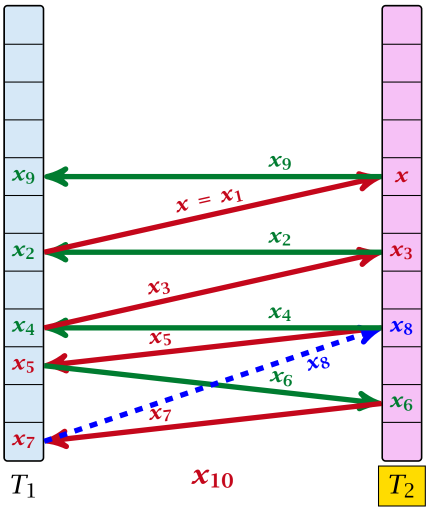
Cuckoo Hashing: Insert



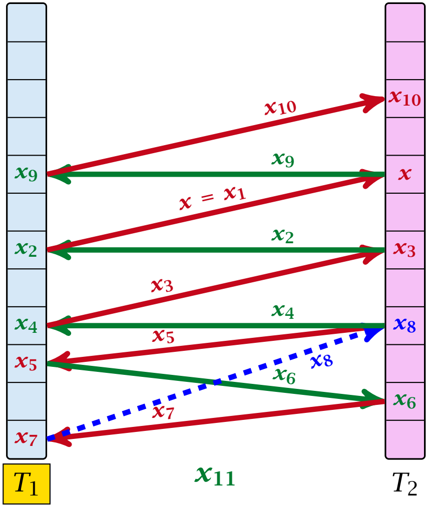
Cuckoo Hashing: Insert



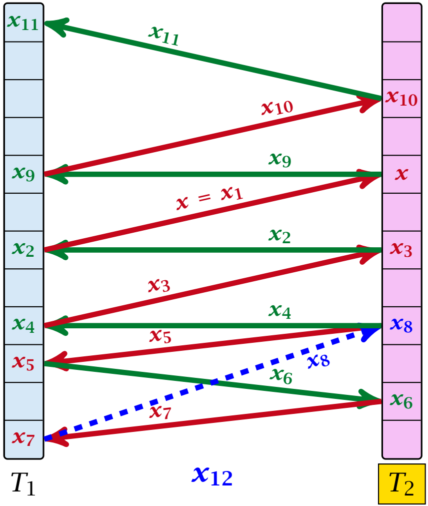
Cuckoo Hashing: Insert



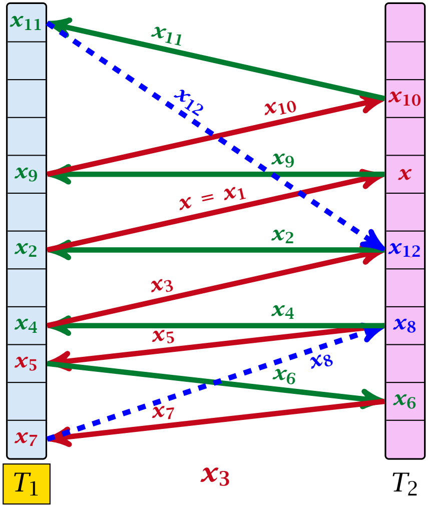
Cuckoo Hashing: Insert



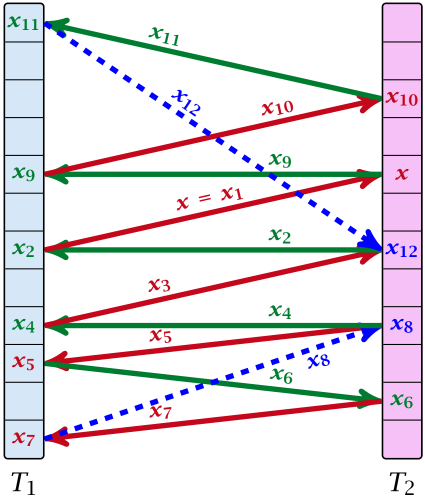
Cuckoo Hashing: Insert



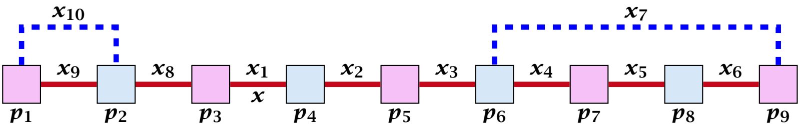
Cuckoo Hashing: Insert



Cuckoo Hashing: Insert

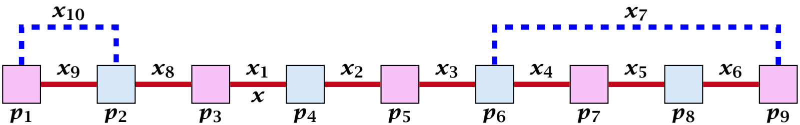


Cuckoo Hashing



A cycle-structure of size s is defined by

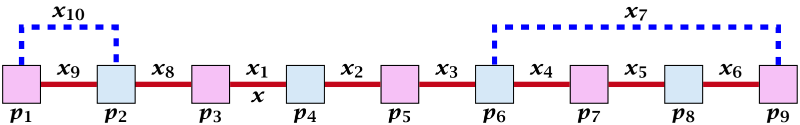
Cuckoo Hashing



A cycle-structure of size s is defined by

- $s-1$ different cells (alternating btw. cells from T_1 and T_2).
- s distinct keys $x = x_1, x_2, \ldots, x_s$, linking the cells.
- The leftmost cell is “linked forward” to some cell on the right.
- The rightmost cell is “linked backward” to a cell on the left.
- One link represents key x; this is where the counting starts.

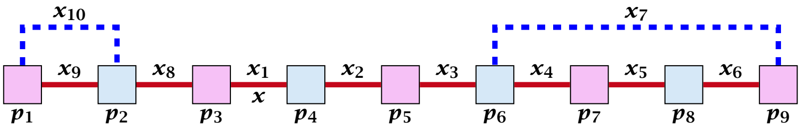
Cuckoo Hashing



A cycle-structure of size s is defined by

- $s-1$ different cells (alternating btw. cells from T_1 and T_2).
- s distinct keys $x = x_1, x_2, \ldots, x_s$, linking the cells.
- The leftmost cell is "linked forward" to some cell on the right.
- The rightmost cell is "linked backward" to a cell on the left.
- One link represents key x; this is where the counting starts.

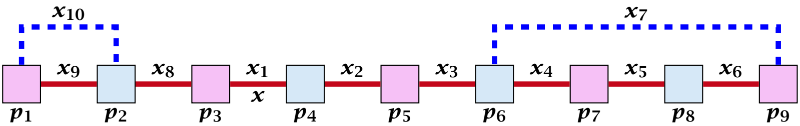
Cuckoo Hashing



A cycle-structure of size s is defined by

- $s-1$ different cells (alternating btw. cells from T_1 and T_2).
- s distinct keys $x = x_1, x_2, \ldots, x_s$, linking the cells.
- The leftmost cell is “linked forward” to some cell on the right.
- The rightmost cell is “linked backward” to a cell on the left.
- One link represents key x; this is where the counting starts.

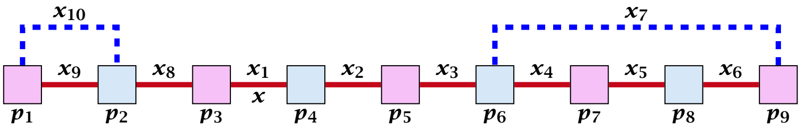
Cuckoo Hashing



A cycle-structure of size s is defined by

- $s-1$ different cells (alternating btw. cells from T_1 and T_2).
- s distinct keys $x = x_1, x_2, \ldots, x_s$, linking the cells.
- The leftmost cell is “linked forward” to some cell on the right.
- The rightmost cell is “linked backward” to a cell on the left.
- One link represents key x; this is where the counting starts.

Cuckoo Hashing



A cycle-structure of size s is defined by

- $s-1$ different cells (alternating btw. cells from T_1 and T_2).
- s distinct keys $x = x_1, x_2, \ldots, x_s$, linking the cells.
- The leftmost cell is "linked forward" to some cell on the right.
- The rightmost cell is "linked backward" to a cell on the left.
- One link represents key x; this is where the counting starts.

Cuckoo Hashing

A cycle-structure is active if for every key x_ℓ (linking a cell p_i from T_1 and a cell p_j from T_2) we have

$$h_1(x_\ell) = p_i \qquad \text{and} \qquad h_2(x_\ell) = p_j$$

Observation:
If during a phase the insert-procedure runs into a cycle there must exist an active cycle structure of size $s \geq 3$.

Cuckoo Hashing

A cycle-structure is active if for every key x_ℓ (linking a cell p_i from T_1 and a cell p_j from T_2) we have

$$h_1(x_\ell) = p_i \qquad \text{and} \qquad h_2(x_\ell) = p_j$$

Observation:
If during a phase the insert-procedure runs into a cycle there must exist an active cycle structure of size $s \geq 3$.

Cuckoo Hashing

What is the probability that all keys in a cycle-structure of size s correctly map into their T_1-cell?

This probability is at most $\frac{\mu}{n^s}$ since h_1 is a (μ, s)-independent hash-function.

What is the probability that all keys in the cycle-structure of size s correctly map into their T_2-cell?

This probability is at most $\frac{\mu}{n^s}$ since h_2 is a (μ, s)-independent hash-function.

These events are independent.

Cuckoo Hashing

What is the probability that all keys in a cycle-structure of size s correctly map into their T_1-cell?

This probability is at most $\frac{\mu}{n^s}$ since h_1 is a (μ, s)-independent hash-function.

What is the probability that all keys in the cycle-structure of size s correctly map into their T_2-cell?

This probability is at most $\frac{\mu}{n^s}$ since h_2 is a (μ, s)-independent hash-function.

These events are independent.

Cuckoo Hashing

What is the probability that all keys in a cycle-structure of size s correctly map into their T_1-cell?

This probability is at most $\frac{\mu}{n^s}$ since h_1 is a (μ, s)-independent hash-function.

What is the probability that all keys in the cycle-structure of size s correctly map into their T_2-cell?

This probability is at most $\frac{\mu}{n^s}$ since h_2 is a (μ, s)-independent hash-function.

These events are independent.

Cuckoo Hashing

What is the probability that all keys in a cycle-structure of size s correctly map into their T_1-cell?

This probability is at most $\frac{\mu}{n^s}$ since h_1 is a (μ, s)-independent hash-function.

What is the probability that all keys in the cycle-structure of size s correctly map into their T_2-cell?

This probability is at most $\frac{\mu}{n^s}$ since h_2 is a (μ, s)-independent hash-function.

These events are independent.

Cuckoo Hashing

What is the probability that all keys in a cycle-structure of size s correctly map into their T_1-cell?

This probability is at most $\frac{\mu}{n^s}$ since h_1 is a (μ, s)-independent hash-function.

What is the probability that all keys in the cycle-structure of size s correctly map into their T_2-cell?

This probability is at most $\frac{\mu}{n^s}$ since h_2 is a (μ, s)-independent hash-function.

These events are independent.

Cuckoo Hashing

The probability that a given cycle-structure of size s is active is at most $\frac{\mu^2}{n^{2s}}$.

What is the probability that there exists an active cycle structure of size s?

Cuckoo Hashing

The probability that a given cycle-structure of size s is active is at most $\frac{\mu^2}{n^{2s}}$.

What is the probability that there exists an active cycle structure of size s?

Cuckoo Hashing

The number of cycle-structures of size s is at most

$$s^3 \cdot n^{s-1} \cdot m^{s-1} \; .$$

- There are at most s^2 possibilities where to attach the forward and backward links.
- There are at most s possibilities to choose where to place key x.
- There are m^{s-1} possibilities to choose the keys apart from x.
- There are n^{s-1} possibilities to choose the cells.

Cuckoo Hashing

The number of cycle-structures of size s is at most

$$s^3 \cdot n^{s-1} \cdot m^{s-1} .$$

- There are at most s^2 possibilities where to attach the forward and backward links.
- There are at most s possibilities to choose where to place key x.
- There are m^{s-1} possibilities to choose the keys apart from x.
- There are n^{s-1} possibilities to choose the cells.

Cuckoo Hashing

The number of cycle-structures of size s is at most

$$s^3 \cdot n^{s-1} \cdot m^{s-1} \ .$$

- There are at most s^2 possibilities where to attach the forward and backward links.
- There are at most s possibilities to choose where to place key x.
- There are m^{s-1} possibilities to choose the keys apart from x.
- There are n^{s-1} possibilities to choose the cells.

Cuckoo Hashing

The number of cycle-structures of size s is at most

$$s^3 \cdot n^{s-1} \cdot m^{s-1} \ .$$

- There are at most s^2 possibilities where to attach the forward and backward links.
- There are at most s possibilities to choose where to place key x.
- There are m^{s-1} possibilities to choose the keys apart from x.
- There are n^{s-1} possibilities to choose the cells.

Cuckoo Hashing

The number of cycle-structures of size s is at most

$$s^3 \cdot n^{s-1} \cdot m^{s-1} \ .$$

- There are at most s^2 possibilities where to attach the forward and backward links.
- There are at most s possibilities to choose where to place key x.
- There are m^{s-1} possibilities to choose the keys apart from x.
- There are n^{s-1} possibilities to choose the cells.

Cuckoo Hashing

The probability that there exists an active cycle-structure is therefore at most

$$\sum_{s=3}^{\infty} s^3 \cdot n^{s-1} \cdot m^{s-1} \cdot \frac{\mu^2}{n^{2s}}$$

Cuckoo Hashing

The probability that there exists an active cycle-structure is therefore at most

$$\sum_{s=3}^{\infty} s^3 \cdot n^{s-1} \cdot m^{s-1} \cdot \frac{\mu^2}{n^{2s}} = \frac{\mu^2}{nm} \sum_{s=3}^{\infty} s^3 \left(\frac{m}{n}\right)^s$$

Cuckoo Hashing

The probability that there exists an active cycle-structure is therefore at most

$$\sum_{s=3}^{\infty} s^3 \cdot n^{s-1} \cdot m^{s-1} \cdot \frac{\mu^2}{n^{2s}} = \frac{\mu^2}{nm} \sum_{s=3}^{\infty} s^3 \left(\frac{m}{n}\right)^s$$

$$\leq \frac{\mu^2}{m^2} \sum_{s=3}^{\infty} s^3 \left(\frac{1}{1+\epsilon}\right)^s$$

Cuckoo Hashing

The probability that there exists an active cycle-structure is therefore at most

$$\sum_{s=3}^{\infty} s^3 \cdot n^{s-1} \cdot m^{s-1} \cdot \frac{\mu^2}{n^{2s}} = \frac{\mu^2}{nm} \sum_{s=3}^{\infty} s^3 \left(\frac{m}{n}\right)^s$$

$$\leq \frac{\mu^2}{m^2} \sum_{s=3}^{\infty} s^3 \left(\frac{1}{1+\epsilon}\right)^s \leq \mathcal{O}\left(\frac{1}{m^2}\right) \quad .$$