Advanced Algorithms – COMS31900

Hashing part two

Static Perfect Hashing

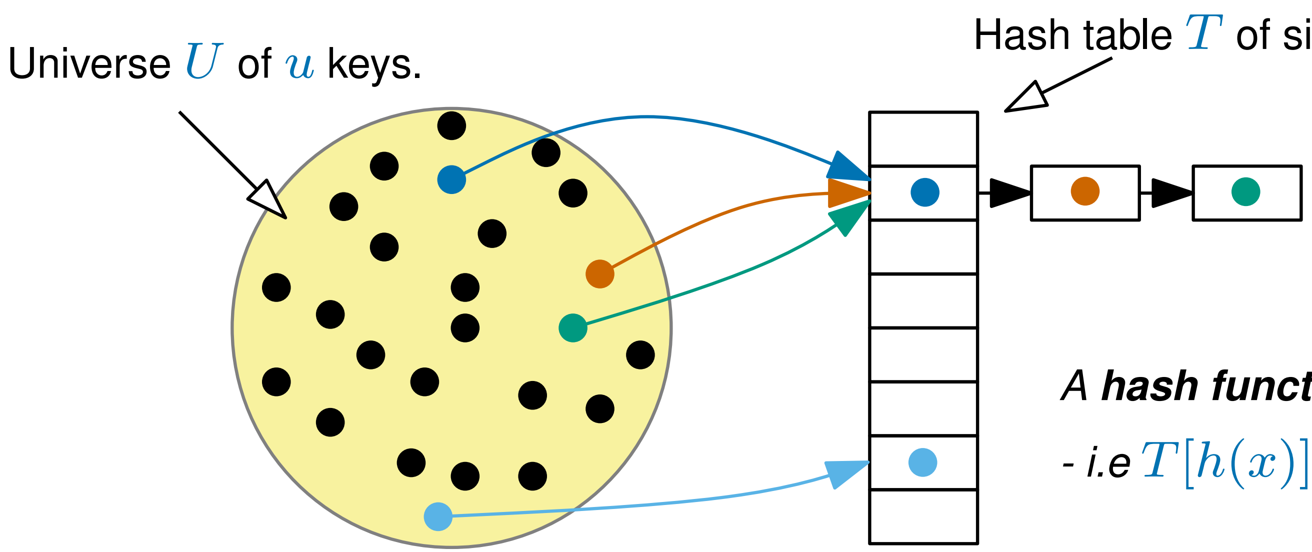
Raphaël Clifford

Slides by Benjamin Sach

Dictionaries and Hashing recap

- A **dynamic dictionary** stores $($*key*, *value*$)$-pairs and supports:

  add$($*key*, *value*$)$, lookup$($*key*$)$ (which returns *value*) and delete$($*key*$)$



Collisions were fixed by **chaining**

(building linked lists)

*A **hash function** maps a key x to position $h(x)$*

- i.e $T[h(x)] = ($key, value$)$.

n *arbitrary operations arrive online, one at a time.*

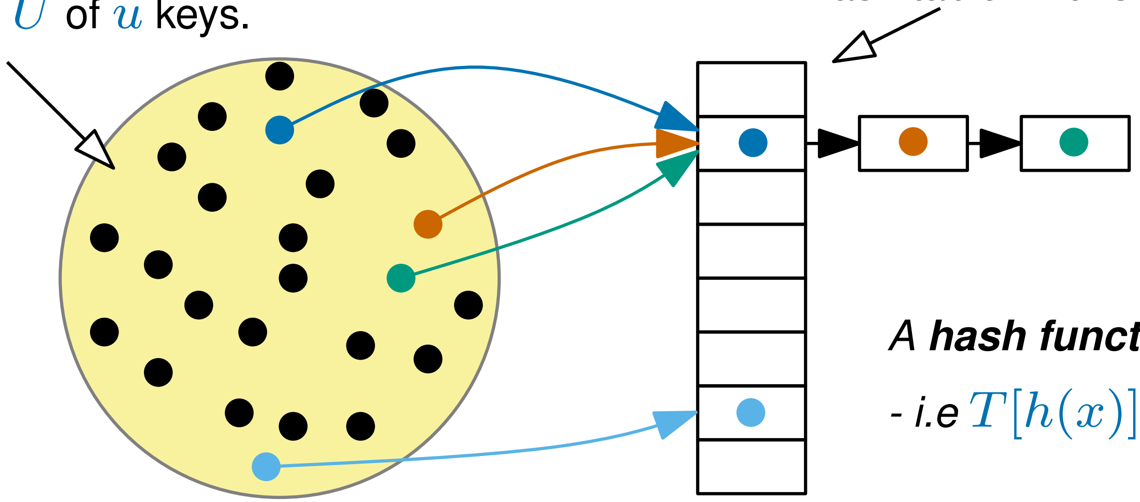
Dictionaries and Hashing recap

- A **dynamic dictionary** stores $($*key*, *value*$)$-pairs and supports:

  add$($*key*, *value*$)$, lookup$($*key*$)$ (which returns *value*) and delete$($*key*$)$

Universe U of u keys.

Hash table T of size $m \geqslant n$.



Collisions were fixed by **chaining**

(building linked lists)

*A **hash function** maps a key x to position $h(x)$*

- i.e $T[h(x)] = ($key, value$)$.

n arbitrary operations arrive online, one at a time.

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ (with $x \neq y$),

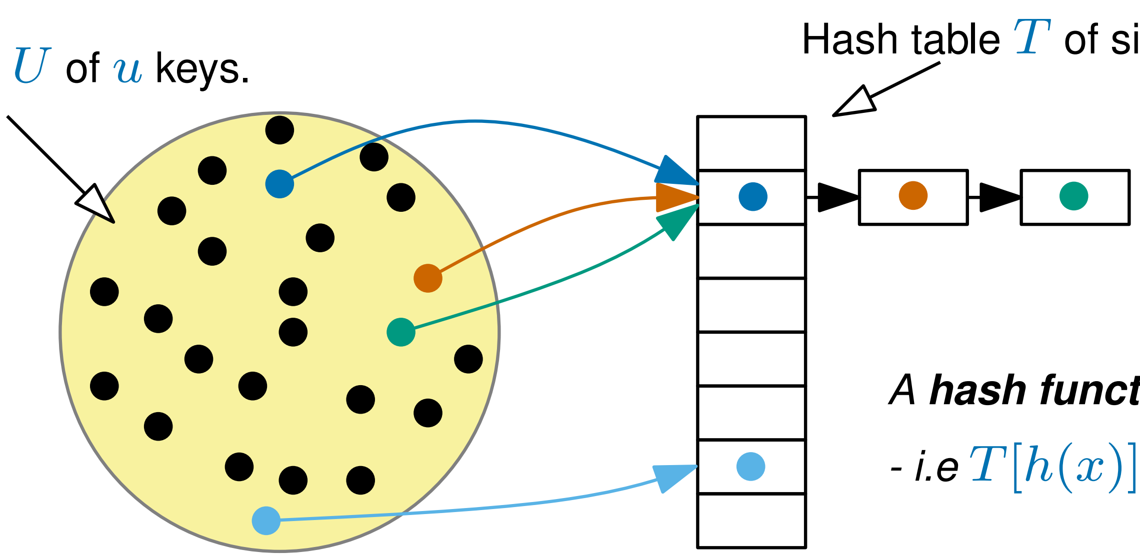
$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$

(h is picked uniformly at random from H)

Dictionaries and Hashing recap

- A **dynamic dictionary** stores $(key, value)$-pairs and supports:

  add$(key, value)$, lookup(key) (which returns *value*) and delete(key)



Collisions were fixed by **chaining**

(building linked lists)

*A **hash function** maps a key x to position $h(x)$*

- i.e $T[h(x)] = (key, value)$.

n *arbitrary operations arrive online, one at a time.*

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ (with $x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$

(h is picked uniformly at random from H)

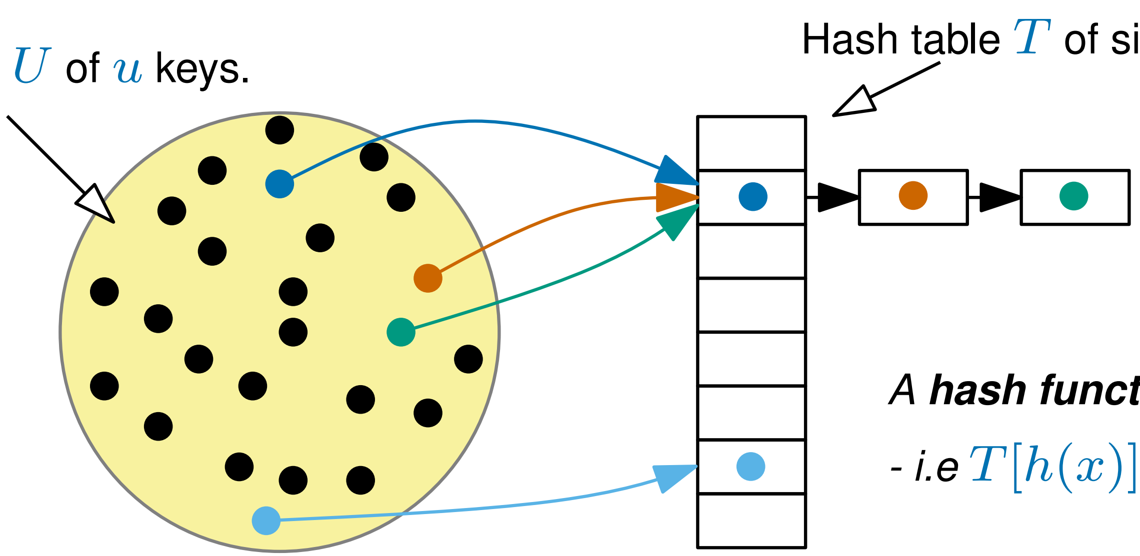
Using weakly universal hashing:

For *any* n operations, the *expected* run-time is $O(1)$ per operation.

Dictionaries and Hashing recap

- A **dynamic dictionary** stores $(key, value)$-pairs and supports:

  add$(key, value)$, lookup(key) (which returns *value*) and delete(key)



Collisions were fixed by **chaining**

(building linked lists)

*A **hash function** maps a key x to position $h(x)$*

- i.e $T[h(x)] = (key, value)$.

n arbitrary operations arrive online, one at a time.

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ (with $x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$

(h is picked uniformly at random from H)

Using weakly universal hashing:

For *any* n operations, the *expected* run-time is $O(1)$ per operation.

But this doesn't tell us much about the *worst-case behaviour*

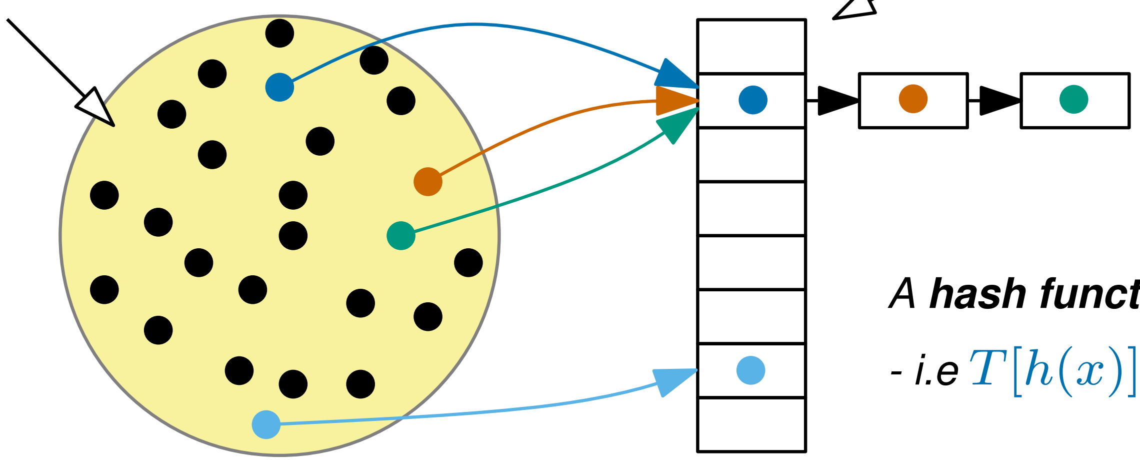
Static Dictionaries and Perfect hashing

- A **static dictionary** stores $(key, value)$-pairs and supports:

  lookup(key) (which returns *value*) - *no inserts or deletes are allowed*

Universe U of u keys.

Hash table T of size $m \geqslant n$.



Collisions were fixed by **chaining** *(building linked lists)*

*A **hash function** maps a key x to position $h(x)$*

- i.e $T[h(x)] = (key, value)$.

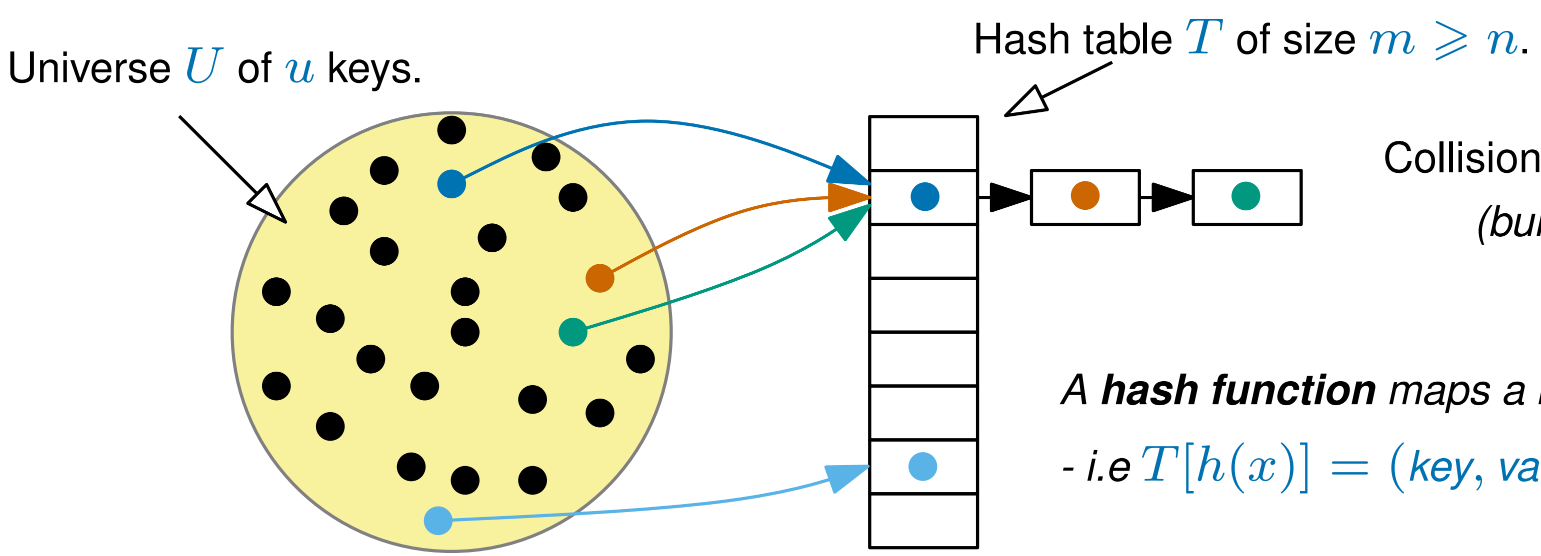
we are given n different $(key, value)$-pairs and want to pick a *good* h

---

Static Dictionaries and Perfect hashing

- A **static dictionary** stores $(key, value)$-pairs and supports:

  lookup(key) (which returns *value*) - *no inserts or deletes are allowed*



Collisions were fixed by **chaining** *(building linked lists)*

A ***hash function*** *maps a key x to position $h(x)$*

- i.e $T[h(x)] = (key, value)$.

we are given n different $(key, value)$-pairs and want to pick a *good* h

---

THEOREM

The FKS hashing scheme:

- Has no collisions
- Every lookup takes $O(1)$ *worst-case* time,
- Uses $O(n)$ space,
- Can be built in $O(n)$ expected time.

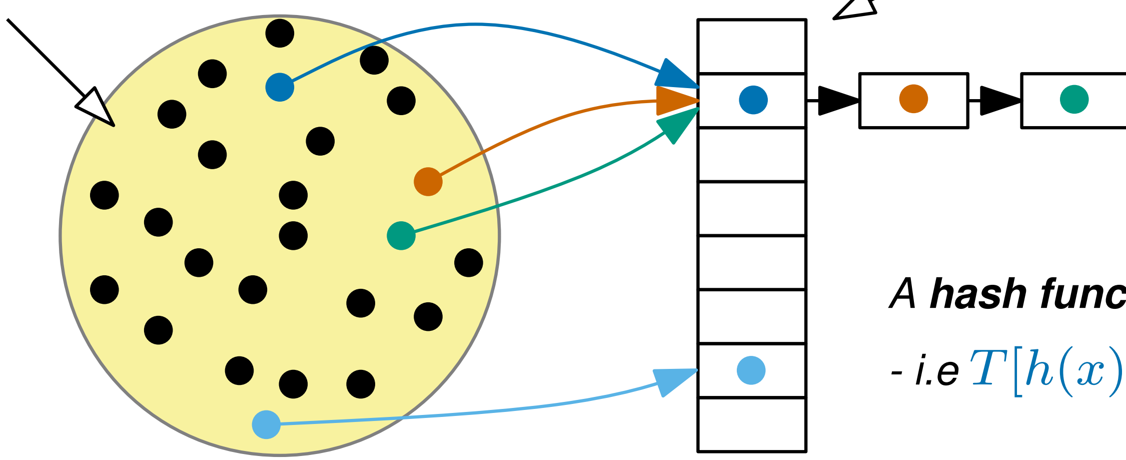
Static Dictionaries and Perfect hashing

- A **static dictionary** stores $(key, value)$-pairs and supports:

  lookup(key) (which returns *value*) - *no inserts or deletes are allowed*

Universe U of u keys.

Hash table T of size $m \geqslant n$.



Collisions were fixed by **chaining**

(building linked lists)

*A **hash function** maps a key x to position $h(x)$*

- i.e $T[h(x)] = (key, value)$.

we are given n different $(key, value)$-pairs and want to pick a *good* h

---

Theorem

The FKS hashing scheme:

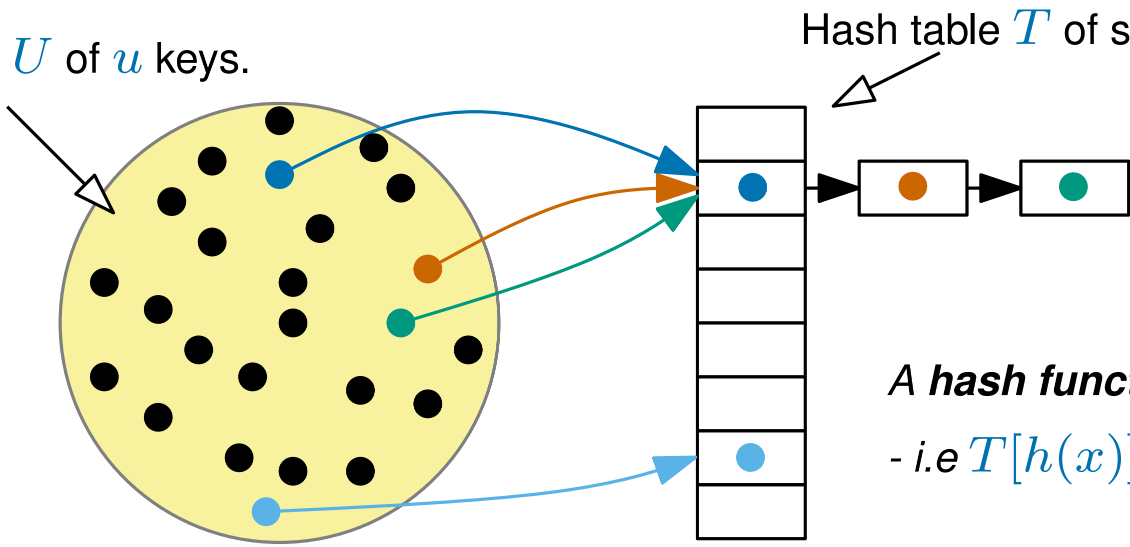
- Has no collisions
- Every lookup takes $O(1)$ *worst-case* time,
- Uses $O(n)$ space,
- Can be built in $O(n)$ expected time.

The rest of this lecture is devoted to the FKS scheme

Static Dictionaries and Perfect hashing

- A **static dictionary** stores $(key, value)$-pairs and supports:

  lookup(key) (which returns *value*) - *no inserts or deletes are allowed*



Collisions were fixed by **chaining**

(building linked lists)

*A **hash function** maps a key x to position $h(x)$*

- i.e $T[h(x)] = (key, value)$.

we are given n different $(key, value)$-pairs and want to pick a *good* h

---

THEOREM

The FKS hashing scheme:

- Has no collisions
- Every lookup takes $O(1)$ *worst-case* time,
- Uses $O(n)$ space,
- Can be built in $O(n)$ expected time.

The rest of this lecture is devoted to the FKS scheme

The construction is based on weak universal hashing

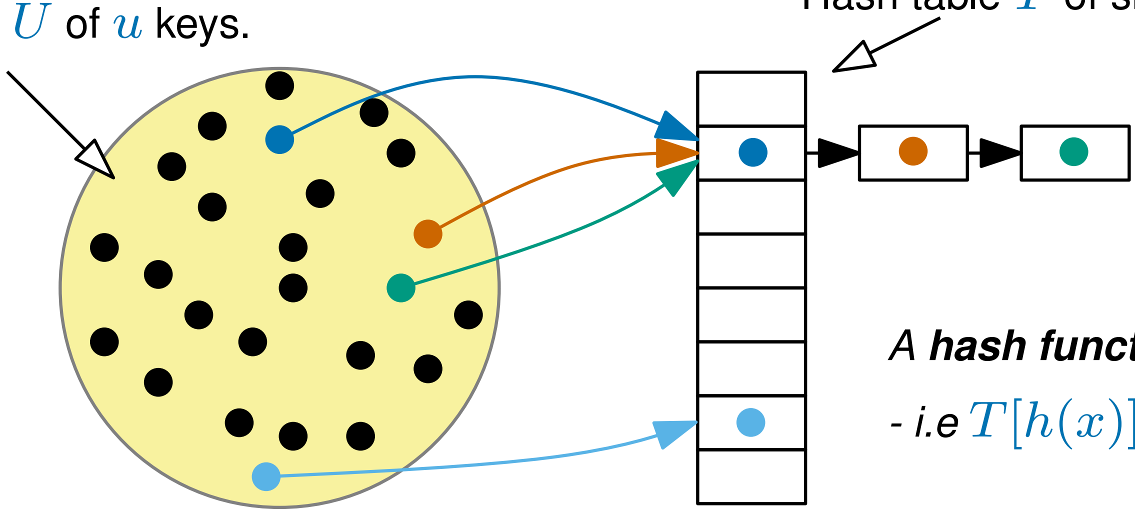
Static Dictionaries and Perfect hashing

- A **static dictionary** stores $(key, value)$-pairs and supports:

  lookup(key) (which returns *value*) - *no inserts or deletes are allowed*

Universe U of u keys.

Hash table T of size $m \geqslant n$.



Collisions were fixed by **chaining**

(building linked lists)

*A **hash function** maps a key x to position $h(x)$*

- i.e $T[h(x)] = (key, value)$.

we are given n different $(key, value)$-pairs and want to pick a *good* h

---

Theorem

The FKS hashing scheme:

- Has no collisions
- Every lookup takes $O(1)$ *worst-case* time,
- Uses $O(n)$ space,
- Can be built in $O(n)$ expected time.

The rest of this lecture is devoted to the FKS scheme

The construction is based on weak universal hashing

(with an $O(1)$ time hash function)

Perfect hashing - a first attempt

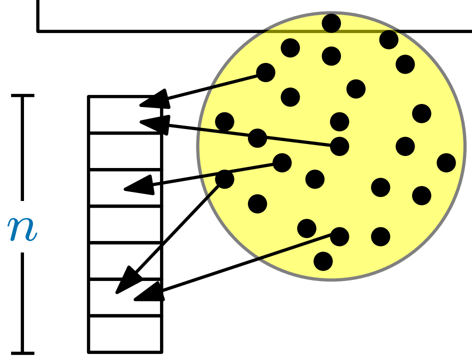
A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ $(x \neq y)$,

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H

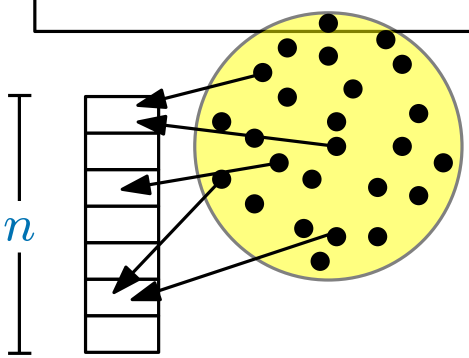


Step 1: Insert everything into a hash table of size $m = n$
using a weakly universal hash function

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



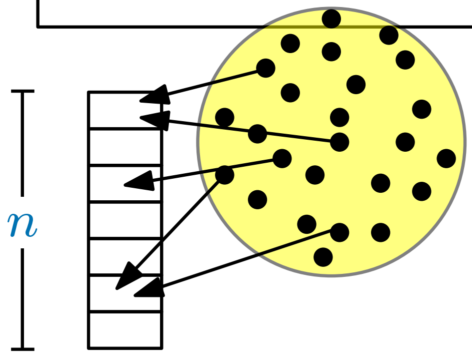
Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

(where any $h(x)$ can be computed in $O(1)$ time)

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H

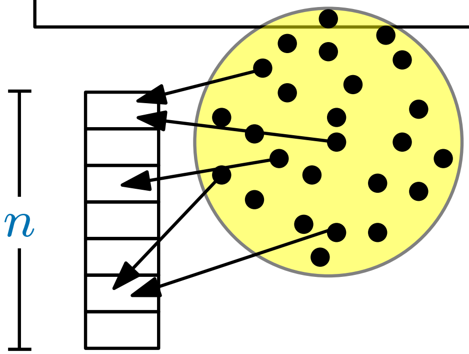


Step 1: Insert everything into a hash table of size $m = n$
using a weakly universal hash function

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



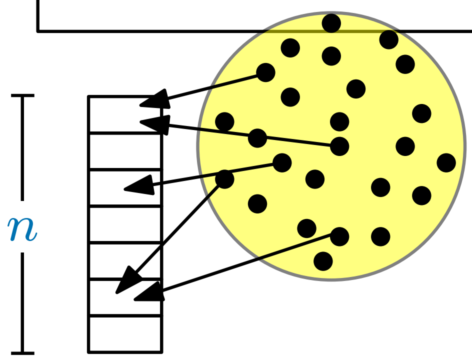
Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

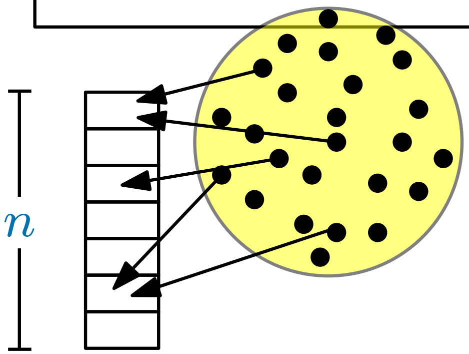
Step 2: Check for collisions

Step 3: Profit!

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

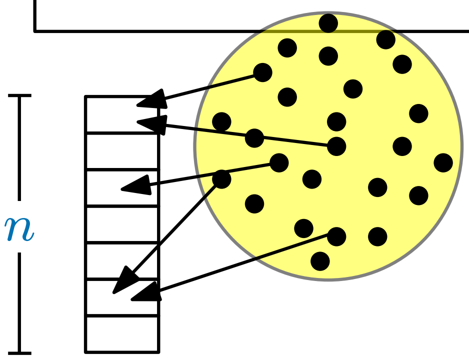
Step 2: Check for collisions

Step 3: *Repeat if necessary*

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

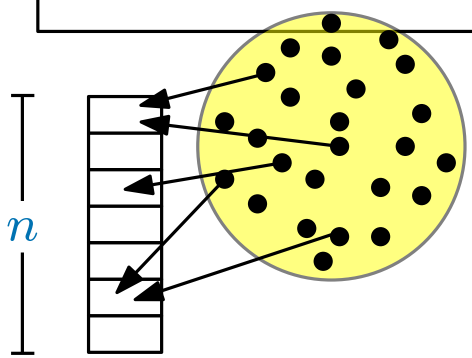
Step 3: *Repeat if necessary*

How many collisions do we get on average?

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\big(h(x) = h(y)\big) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

number of collisions

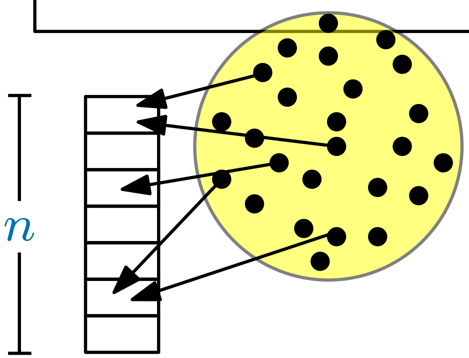
$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big)$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\big(h(x) = h(y)\big) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

number of collisions

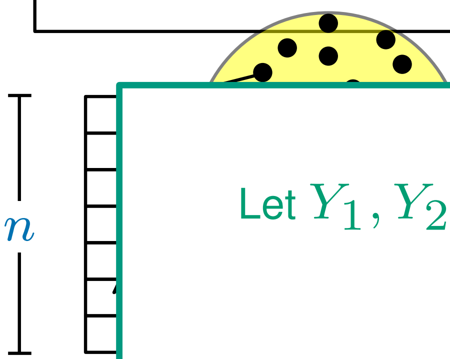
$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big) = \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y})$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



ble of size $m = n$

kly universal hash function

Linearity of Expectation

Let $Y_1, Y_2, \ldots, Y_k$ be k random variables. Then

$$\mathbb{E}\left(\sum_{i=1}^{k} Y_i\right) = \sum_{i=1}^{k} \mathbb{E}(Y_i)$$

erage?

number of collisions

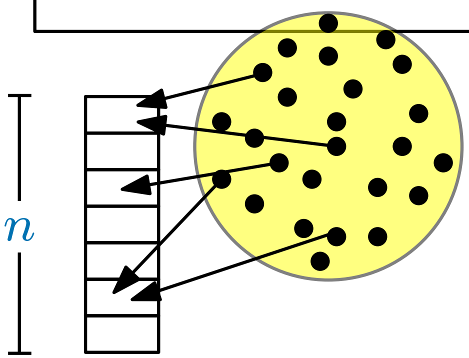
$$\mathbb{E}(C) = \mathbb{E}\left(\sum_{x,y \in T, x<y} I_{x,y}\right) = \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y})$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

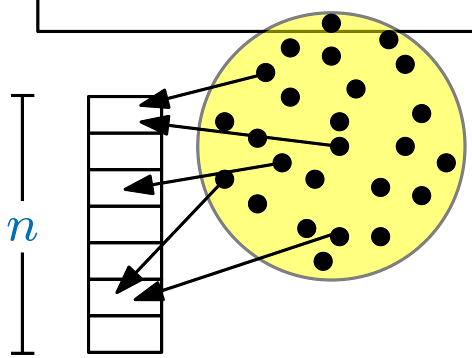
$$\underbrace{\mathbb{E}(C)}_{\text{number of collisions}} = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y} \Big) \underbrace{=}_{\text{linearity of expectation}} \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y})$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

number of collisions

linearity of expectation

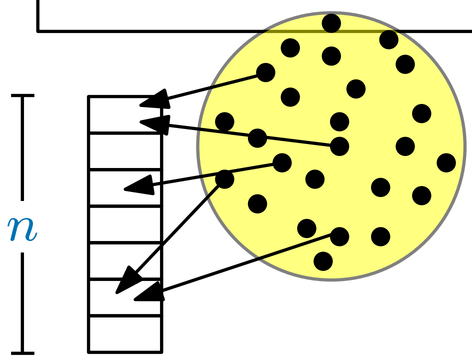
$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big) = \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y}) \leqslant \sum_{x,y \in T,\, x<y} \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m} \quad \text{where } h \text{ is picked uniformly at random from } H$$



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

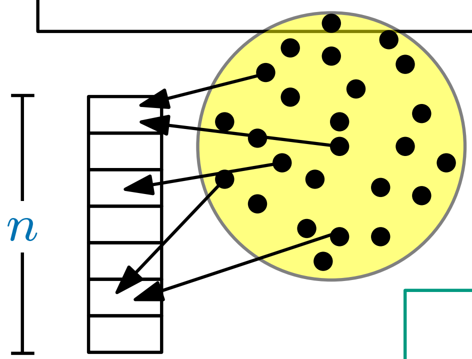
$$\underbrace{\mathbb{E}(C)}_{\text{number of collisions}} = \mathbb{E}\Big(\sum_{x,y\in T, x<y} I_{x,y}\Big) \underbrace{=}_{\text{linearity of expectation}} \sum_{x,y\in T,\, x<y} \mathbb{E}(I_{x,y}) \underbrace{\leqslant}_{\text{definition of expectation}} \sum_{x,y\in T,\, x<y} \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$
using a weakly universal hash function

Step 2: Check for collisions

By the definition of expectation...

$$\mathbb{E}(I_{x,y}) = 1 \cdot \Pr(I_{x,y} = 1) + 0 \cdot \Pr(I_{x,y} = 0) \leqslant \frac{1}{m}$$

number of collisions

linearity of expectation

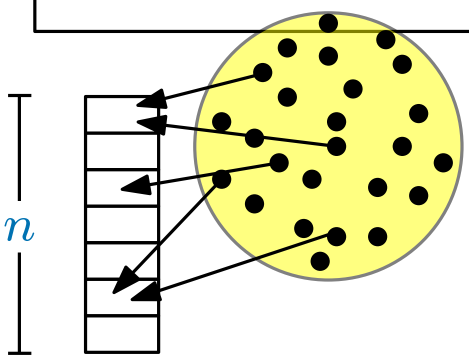
$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big) = \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y}) \leqslant \sum_{x,y \in T,\, x<y} \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m} \quad \text{where } h \text{ is picked uniformly at random from } H$$



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

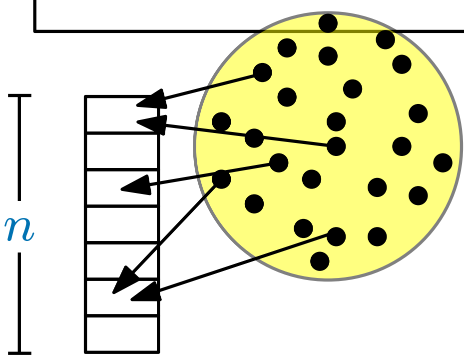
$$\underbrace{\mathbb{E}(C)}_{\text{number of collisions}} = \mathbb{E}\Big(\sum_{x,y\in T, x<y} I_{x,y}\Big) \overset{\text{linearity of expectation}}{=} \sum_{x,y\in T,\, x<y} \mathbb{E}(I_{x,y}) \overset{\text{definition of expectation}}{\leqslant} \sum_{x,y\in T,\, x<y} \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

number of collisions — linearity of expectation — definition of expectation

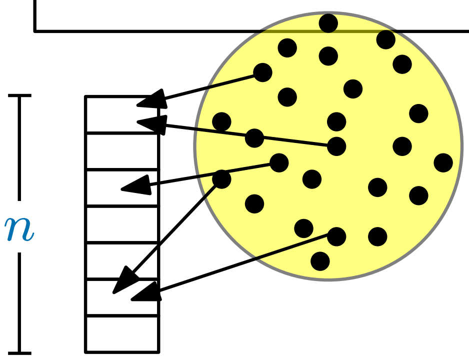
$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big) = \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y}) \leqslant \sum_{x,y \in T,\, x<y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ $(x \neq y)$,

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

$$\boxed{\binom{n}{2} = \frac{n(n-1)}{2}}$$

number of collisions

linearity of expectation

definition of expectation

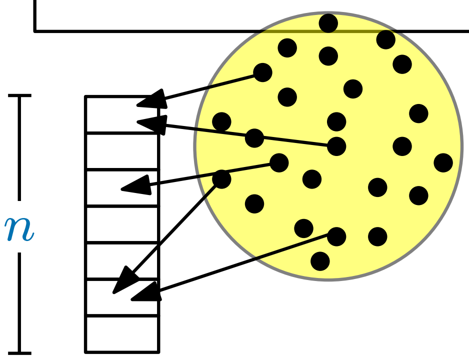
$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big) = \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y}) \leqslant \sum_{x,y \in T,\, x<y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

number of collisions | linearity of expectation | definition of expectation | $\leqslant n^2/2$

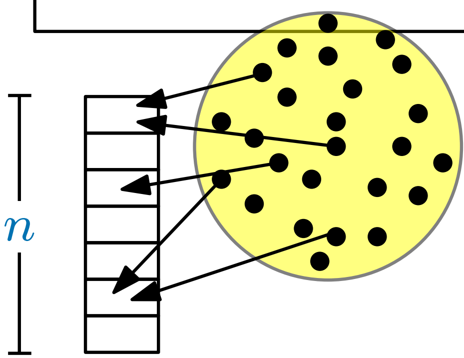
$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big) = \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y}) \leqslant \sum_{x,y \in T,\, x<y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ $(x \neq y)$,

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

number of collisions — linearity of expectation — definition of expectation — $\leqslant n^2/2$

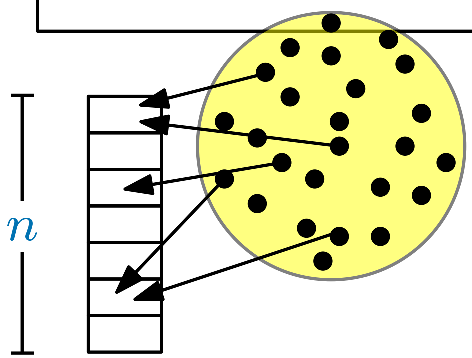
$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big) = \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y}) \leqslant \sum_{x,y \in T,\, x<y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m} \leqslant \frac{n^2}{2m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a first attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ $(x \neq y)$,

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m} \quad \text{where } h \text{ is picked uniformly at random from } H$$



Step 1: Insert everything into a hash table of size $m = n$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

number of collisions — linearity of expectation — definition of expectation

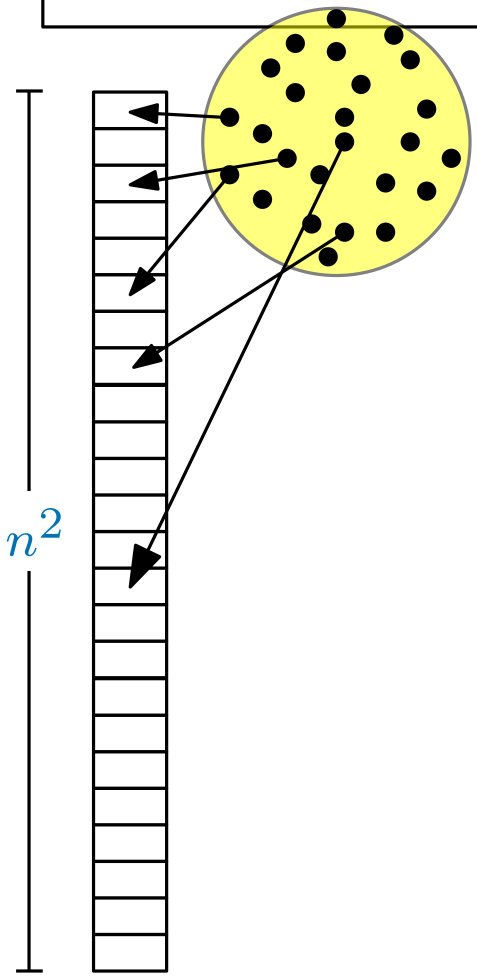
$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big) = \sum_{x,y\in T,\, x<y} \mathbb{E}(I_{x,y}) \leqslant \sum_{x,y\in T,\, x<y} \frac{1}{m} = \overbrace{\binom{n}{2}}^{\leqslant n^2/2} \cdot \frac{1}{m} \leqslant \frac{n^2}{2m} \leqslant \frac{n}{2}.$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a second attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

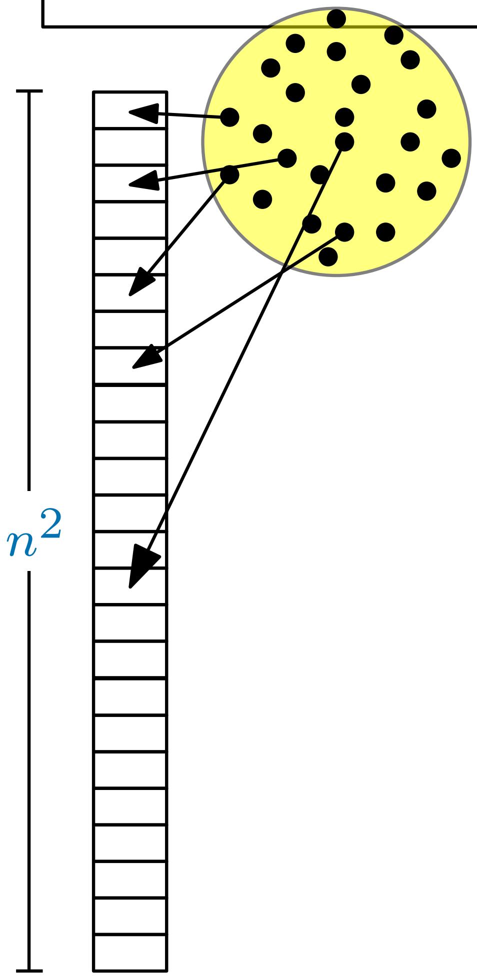
Step 2: Check for collisions

Step 3: *Repeat if necessary*

Perfect hashing - a second attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

Step 2: Check for collisions

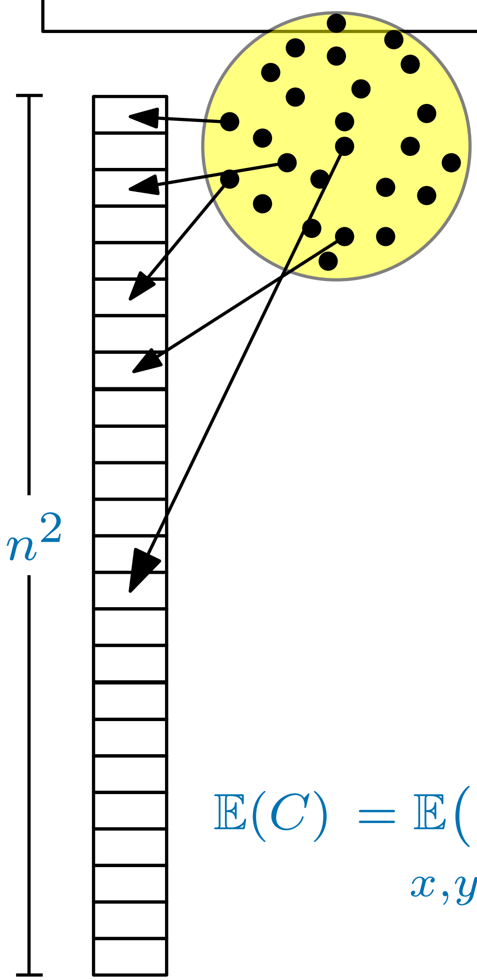
Step 3: *Repeat if necessary*

How many collisions do we get on average?

Perfect hashing - a second attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

number of collisions

linearity of expectation

definition of expectation

$\leqslant n^2/2$

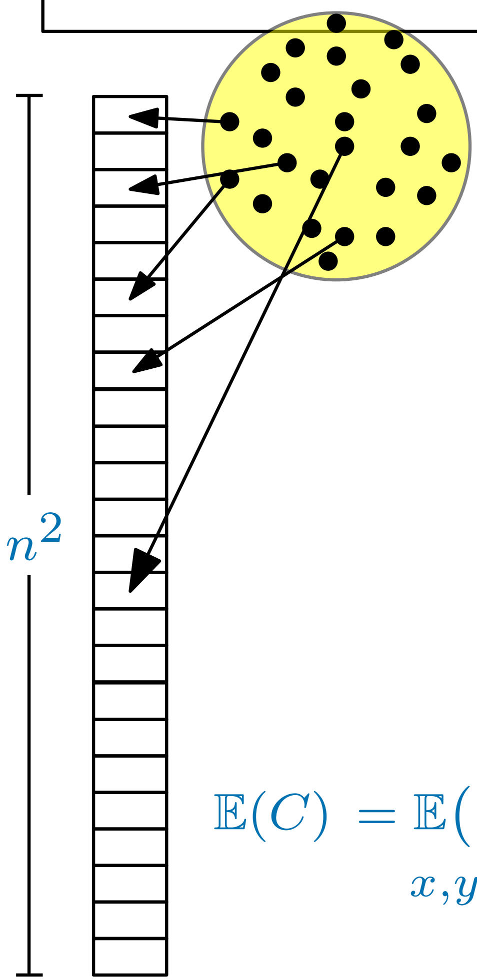
$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y\in T, x<y} I_{x,y}\Big) = \sum_{x,y\in T,\, x<y} \mathbb{E}(I_{x,y}) \leqslant \sum_{x,y\in T,\, x<y} \frac{1}{m} = \binom{n}{2}\cdot\frac{1}{m} \leqslant \frac{n^2}{2m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a second attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ $(x \neq y)$,

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m}$$ where h is picked uniformly at random from H



Step 1: Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

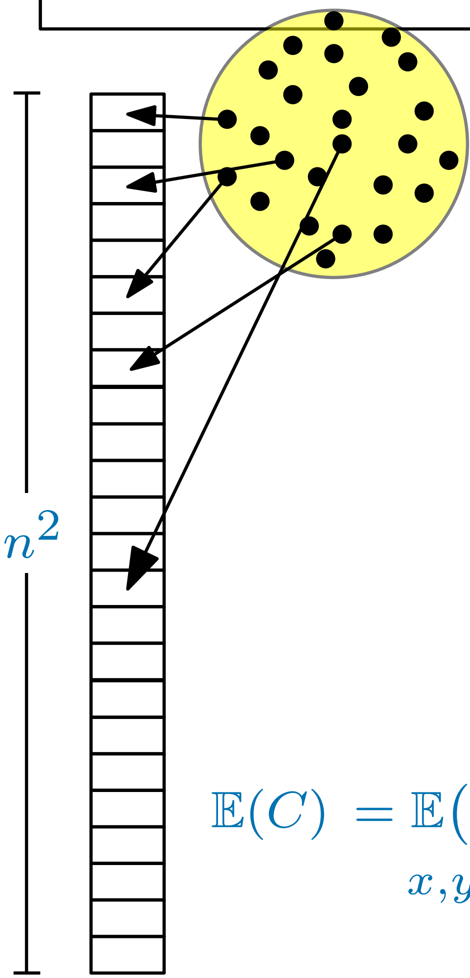
$$\mathbb{E}(C) \overset{\text{number of collisions}}{=} \mathbb{E}\Big(\sum_{x,y\in T, x<y} I_{x,y}\Big) \overset{\text{linearity of expectation}}{=} \sum_{x,y\in T,\, x<y} \mathbb{E}(I_{x,y}) \overset{\text{definition of expectation}}{\leqslant} \sum_{x,y\in T,\, x<y} \frac{1}{m} = \overbrace{\binom{n}{2}}^{\leqslant n^2/2} \cdot \frac{1}{m} \leqslant \frac{n^2}{2m} \leqslant \frac{1}{2}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a second attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m} \quad \text{where } h \text{ is picked uniformly at random from } H$$



Step 1: Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

$$\mathbb{E}(C) \overset{\text{number of collisions}}{=} \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big) \overset{\text{linearity of expectation}}{=} \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y}) \overset{\text{definition of expectation}}{\leqslant} \sum_{x,y \in T,\, x<y} \frac{1}{m} = \underbrace{\binom{n}{2}}_{\leqslant n^2/2} \cdot \frac{1}{m} \leqslant \frac{n^2}{2m} \leqslant \frac{1}{2}$$

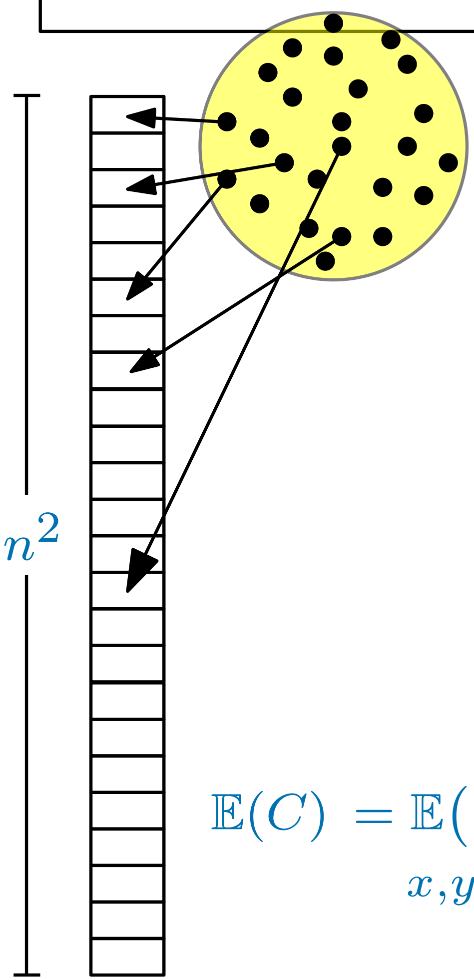
much better!

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

Perfect hashing - a second attempt

A set H of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr\left(h(x) = h(y)\right) \leqslant \frac{1}{m} \quad \text{where } h \text{ is picked uniformly at random from } H$$



Step 1: Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

(except we cheated)

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

number of collisions — linearity of expectation — definition of expectation — $\leqslant n^2/2$

$$\mathbb{E}(C) = \mathbb{E}\Big(\sum_{x,y \in T, x<y} I_{x,y}\Big) = \sum_{x,y \in T,\, x<y} \mathbb{E}(I_{x,y}) \leqslant \sum_{x,y \in T,\, x<y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m} \leqslant \frac{n^2}{2m} \leqslant \frac{1}{2}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

much better!

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there was a collision*

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there was a collision*

How many times do we repeat on average?

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there was a collision*

How many times do we repeat on average?

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geqslant 1) \leqslant \frac{1}{2}$

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$

Step 2: Che

Step 3: *Rep*

How n

Markov's inequality

If X is a non-negative r.v., then for all $a > 0$,

$$\Pr(X \geqslant a) \leqslant \frac{\mathbb{E}(X)}{a}.$$

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geqslant 1) \leqslant \frac{1}{2}$

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there was a collision*

How many times do we repeat on average?

Markov's inequality

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geqslant 1) \leqslant \frac{1}{2}$

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there was a collision*

How many times do we repeat on average?

Markov's inequality

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geqslant 1) \leqslant \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there was a collision*

How many times do we repeat on average?

Markov's inequality

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geqslant 1) \leqslant \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$

i.e. at least as good as tossing a heads on a fair coin

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there was a collision*

How many times do we repeat on average?

Markov's inequality

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geqslant 1) \leqslant \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$

i.e. at least as good as tossing a heads on a fair coin

$$\mathbb{E}(\text{runs}) \leqslant \mathbb{E}(\text{coin tosses to get a heads}) = 2$$

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there was a collision*

How many times do we repeat on average?

Markov's inequality

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geqslant 1) \leqslant \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$

i.e. at least as good as tossing a heads on a fair coin

$$\mathbb{E}(\text{runs}) \leqslant \mathbb{E}(\text{coin tosses to get a heads}) = 2$$

$$\mathbb{E}(\text{construction time}) = O(m) \cdot \mathbb{E}(\text{runs}) = O(m) = O(n^2)$$

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there was a collision*

How many times do we repeat on average?

Markov's inequality

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geqslant 1) \leqslant \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$

i.e. at least as good as tossing a heads on a fair coin

$$\mathbb{E}(\text{runs}) \leqslant \mathbb{E}(\text{coin tosses to get a heads}) = 2$$

$$\mathbb{E}(\text{construction time}) = O(m) \cdot \mathbb{E}(\text{runs}) = O(m) = O(n^2)$$

. . . and then the look-up time is always $O(1)$

Expected construction time

Step 1: Insert everything into a hash table of size $m = n^2$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there was a collision*

How many times do we repeat on average?

Markov's inequality

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geqslant 1) \leqslant \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$

i.e. at least as good as tossing a heads on a fair coin

$$\mathbb{E}(\text{runs}) \leqslant \mathbb{E}(\text{coin tosses to get a heads}) = 2$$

$$\mathbb{E}(\text{construction time}) = O(m) \cdot \mathbb{E}(\text{runs}) = O(m) = O(n^2)$$

. . . and then the look-up time is always $O(1)$

(because any $h(x)$ can be computed in $O(1)$ time)

Expected construction time

Step 1: Insert everything into a hash table of size $m = n$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there are more than n collisions*

Expected construction time

Step 1: Insert everything into a hash table of size $m = n$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there are more than n collisions*

This looks rubbish but

it will be useful in a bit!

Expected construction time

Step 1: Insert everything into a hash table of size $m = n$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there are more than n collisions*

This looks rubbish but it will be useful in a bit!

How many times do we repeat on average?

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{n}{2}$

The probability of at least n collisions: $\Pr(C \geqslant n) \leqslant \frac{1}{2}$

Expected construction time

Step 1: Insert everything into a hash table of size $m = n$

ction

Markov's inequality

If X is a non-negative r.v., then for all $a > 0$,

$$\Pr(X \geqslant a) \leqslant \frac{\mathbb{E}(X)}{a}\,.$$

This looks rubbish but it will be useful in a bit!

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{n}{2}$

The probability of at least n collisions: $\Pr(C \geqslant n) \leqslant \frac{1}{2}$ *(where $a = n$)*

Expected construction time

Step 1: Insert everything into a hash table of size $m = n$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there are more than n collisions*

This looks rubbish but it will be useful in a bit!

How many times do we repeat on average?

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{n}{2}$

The probability of at least n collisions: $\Pr(C \geqslant n) \leqslant \frac{1}{2}$

Expected construction time

Step 1: Insert everything into a hash table of size $m = n$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there are more than n collisions*

This looks rubbish but it will be useful in a bit!

How many times do we repeat on average?

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{n}{2}$

The probability of at least n collisions: $\Pr(C \geqslant n) \leqslant \frac{1}{2}$

The probability of at most n collisions is at least $\frac{1}{2}$

i.e. at least as good as tossing a heads on a fair coin

$$\mathbb{E}(\text{runs}) \leqslant \mathbb{E}(\text{coin tosses to get a heads}) = 2$$

$$\mathbb{E}(\text{construction time}) = O(m) \cdot \mathbb{E}(\text{runs}) = O(m) = O(n)$$

Expected construction time

Step 1: Insert everything into a hash table of size $m = n$
using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if there are more than n collisions*

This looks rubbish but it will be useful in a bit!

How many times do we repeat on average?

The expected number of collisions: $\mathbb{E}(C) \leqslant \frac{n}{2}$

The probability of at least n collisions: $\Pr(C \geqslant n) \leqslant \frac{1}{2}$

The probability of at most n collisions is at least $\frac{1}{2}$

i.e. at least as good as tossing a heads on a fair coin

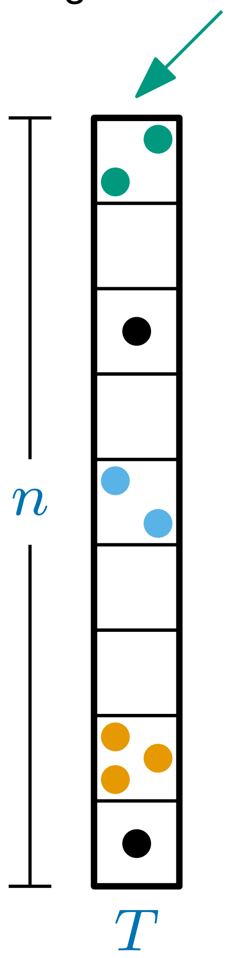
$$\mathbb{E}(\text{runs}) \leqslant \mathbb{E}(\text{coin tosses to get a heads}) = 2$$

$$\mathbb{E}(\text{construction time}) = O(m) \cdot \mathbb{E}(\text{runs}) = O(m) = O(n)$$

...but the look-up time could be rubbish (lots of collisions)

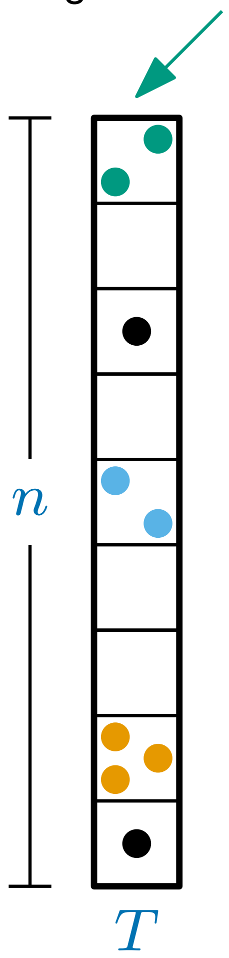
Perfect hashing - attempt three

Step 1: Insert everything into a hash table, T, of size n
using a weakly universal hash function, h



Perfect hashing - attempt three

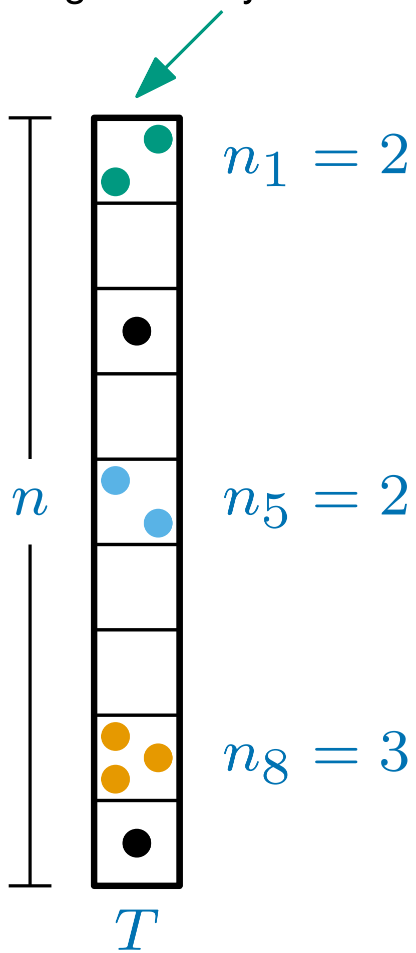
Step 1: Insert everything into a hash table, T, of size n
using a weakly universal hash function, h



Let n_i be the number of items in $T[i]$

Perfect hashing - attempt three

Step 1: Insert everything into a hash table, T, of size n
using a weakly universal hash function, h

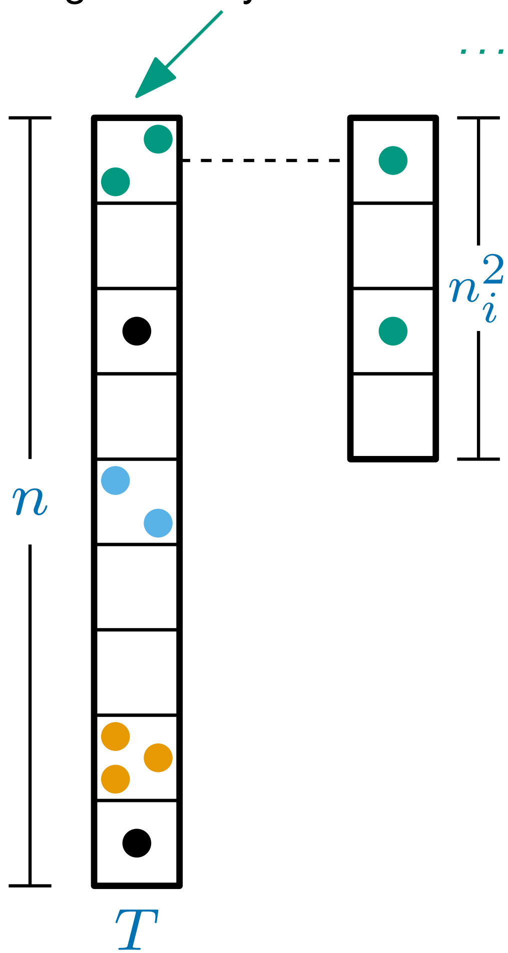


Let n_i be the number of items in $T[i]$

Perfect hashing - attempt three

Step 1: Insert everything into a hash table, T, of size n
using a weakly universal hash function, h

. . . but don't use chaining



Let n_i be the number of items in $T[i]$

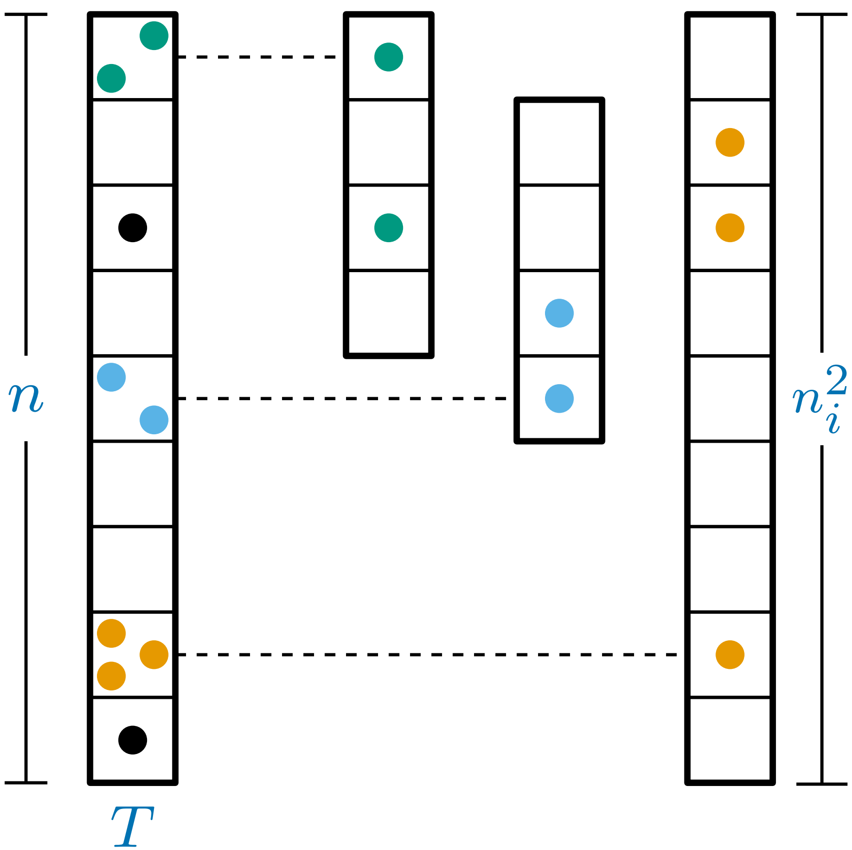
Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2

using another *weakly universal hash function denoted* h_i *(there is one for each* i*)*

Perfect hashing - attempt three

Step 1: Insert everything into a hash table, T, of size n
using a weakly universal hash function, h

. . . but don't use chaining



Let n_i be the number of items in $T[i]$

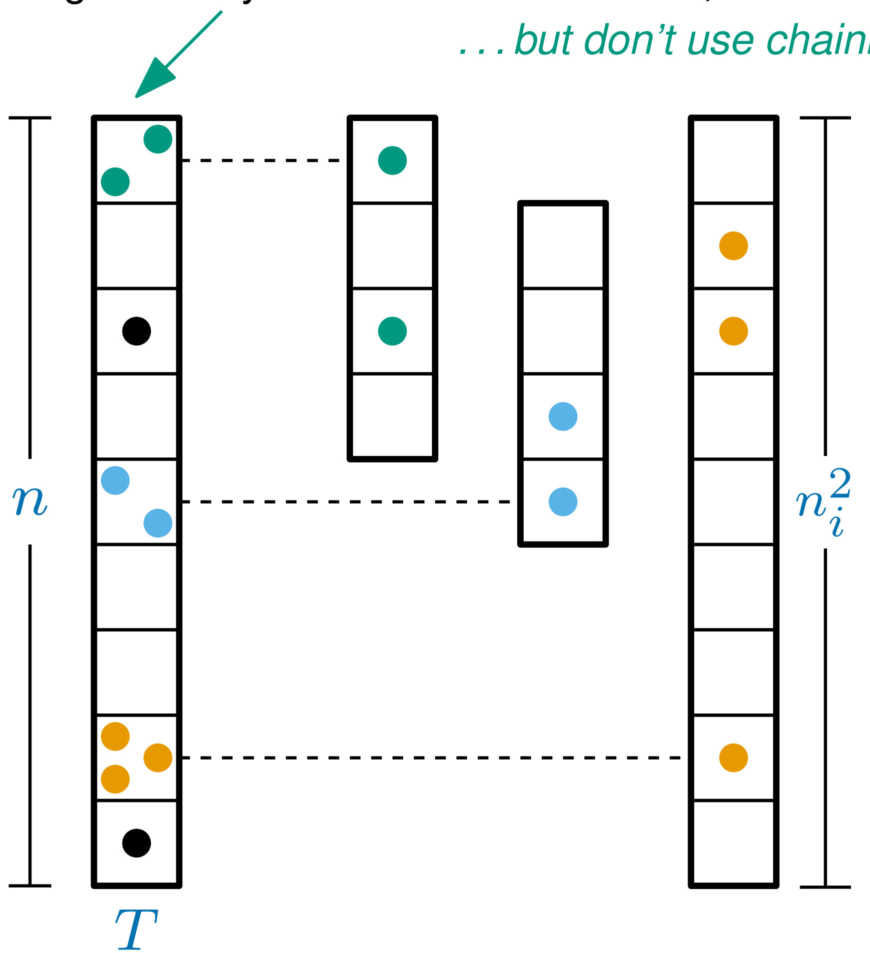
Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2

using another *weakly universal hash function denoted* h_i *(there is one for each* i*)*

Perfect hashing - attempt three

Step 1: Insert everything into a hash table, T, of size n
using a weakly universal hash function, h

. . . but don't use chaining



Let n_i be the number of items in $T[i]$

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2

using another *weakly universal hash function denoted* h_i *(there is one for each* i*)*

(Step 3) *Immediately repeat a step if either*

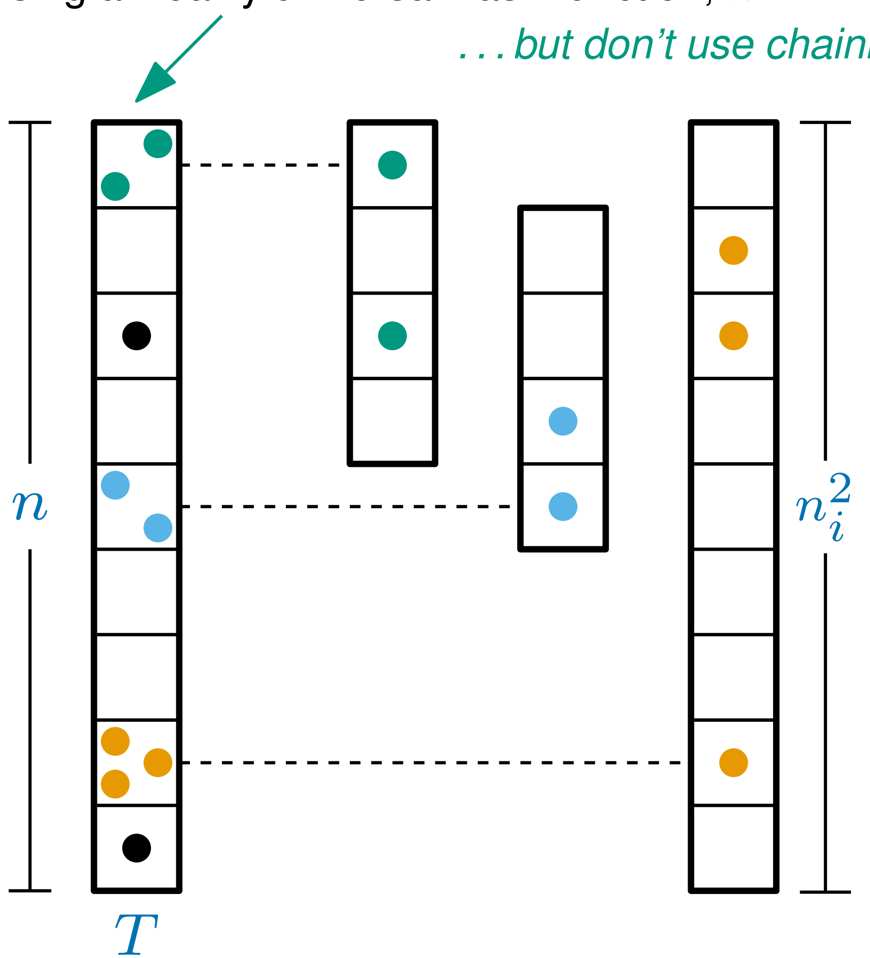
a) T has more than n collisions

b) some T_i has a collision

Perfect hashing - attempt three

Step 1: Insert everything into a hash table, T, of size n
using a weakly universal hash function, h

...but don't use chaining



Let n_i be the number of items in $T[i]$

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2

using another *weakly universal hash function denoted* h_i *(there is one for each* i*)*

(Step 3) *Immediately repeat a step if either*

a) T has more than n collisions

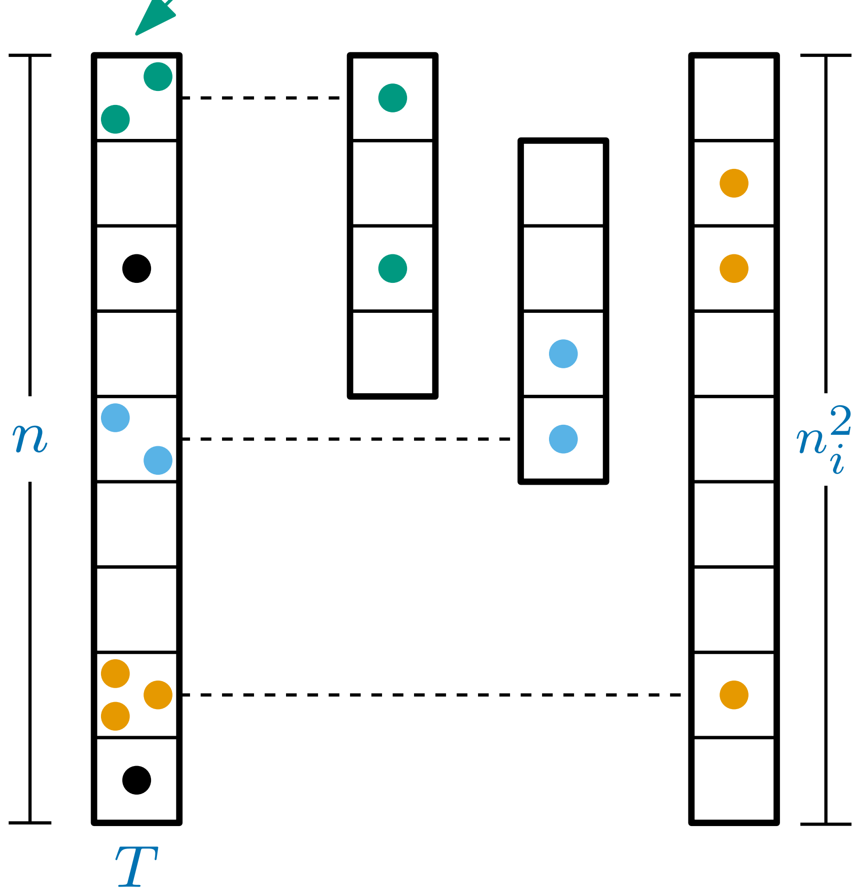
b) some T_i has a collision

i.e. check (and if necessary rebuild) each table immediately after building it

Perfect hashing - attempt three

Step 1: Insert everything into a hash table, T, of size n
using a weakly universal hash function, h

. . . but don't use chaining



Let n_i be the number of items in $T[i]$

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2

using another *weakly universal hash function denoted* h_i *(there is one for each* i*)*

(Step 3) *Immediately repeat a step if either*

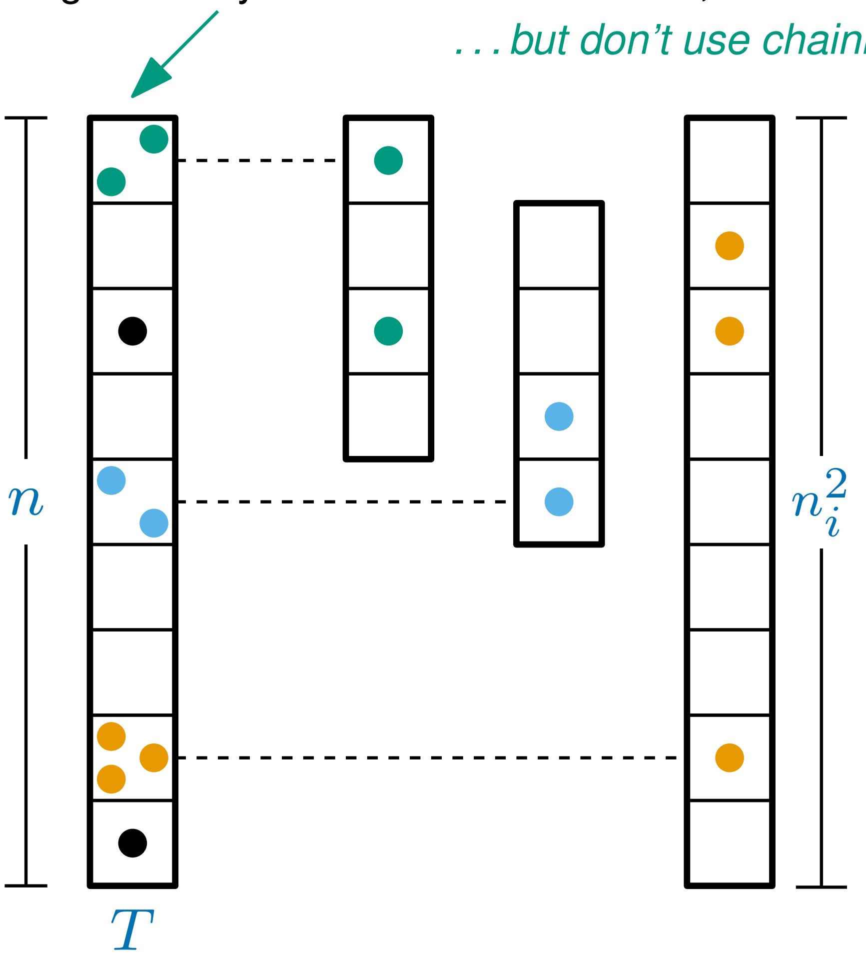
a) T has more than n collisions

b) some T_i has a collision

Perfect hashing - attempt three

Step 1: Insert everything into a hash table, T, of size n
using a weakly universal hash function, h

. . . but don't use chaining



Let n_i be the number of items in $T[i]$

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2

using another *weakly universal hash function denoted* h_i *(there is one for each* i*)*

(Step 3) *Immediately repeat a step if either*

a) T has more than n collisions

b) some T_i has a collision

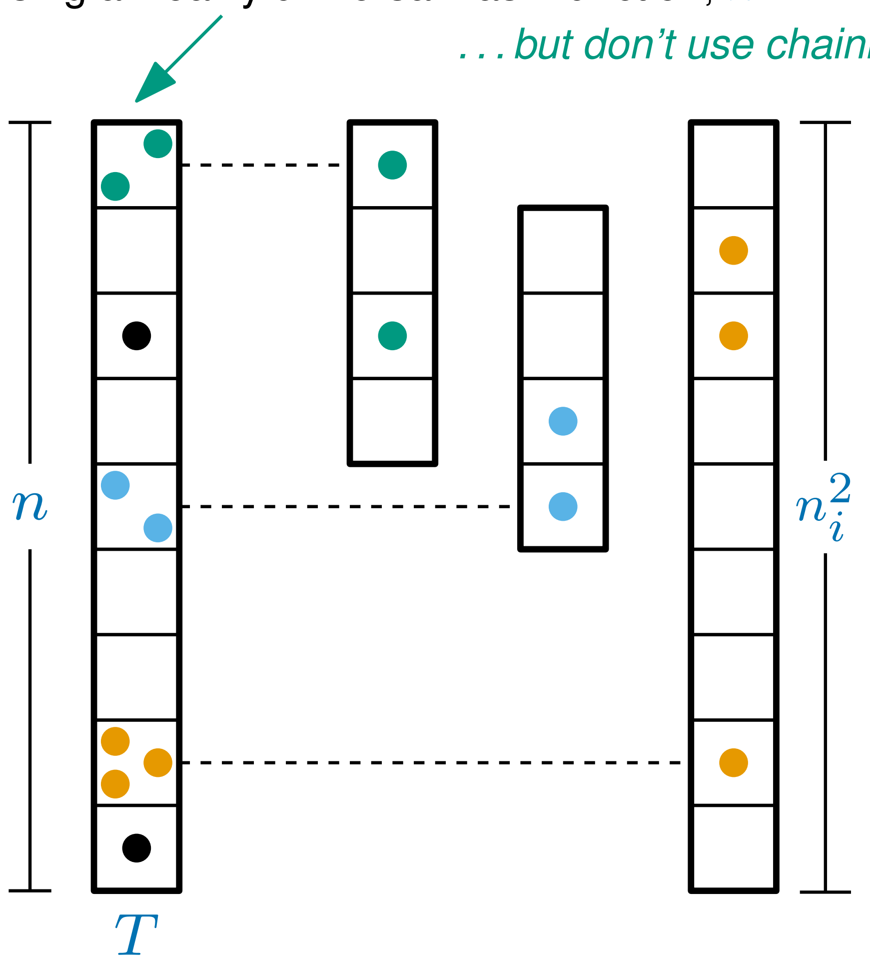
The look-up time is always O(1)

1. Compute $i = h(x)$ (x is the key)
2. Compute $j = h_i(x)$
3. The item is in $T_i[j]$

Perfect hashing - attempt three

Step 1: Insert everything into a hash table, T, of size n
using a weakly universal hash function, h

...but don't use chaining



Let n_i be the number of items in $T[i]$

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2

using another *weakly universal hash function denoted* h_i *(there is one for each* i*)*

(Step 3) *Immediately repeat a step if either*

a) T has more than n collisions

b) some T_i has a collision

The look-up time is always O(1)

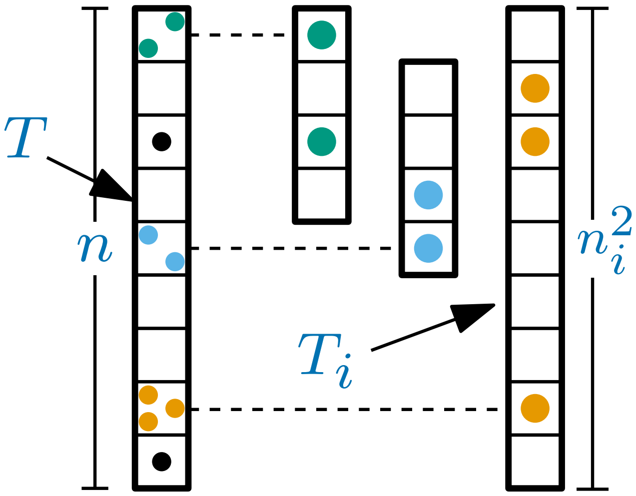
1. Compute $i = h(x)$ (x is the key)
2. Compute $j = h_i(x)$
3. The item is in $T_i[j]$

Two questions remain:

What is the expected construction time?

What is the space usage?

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

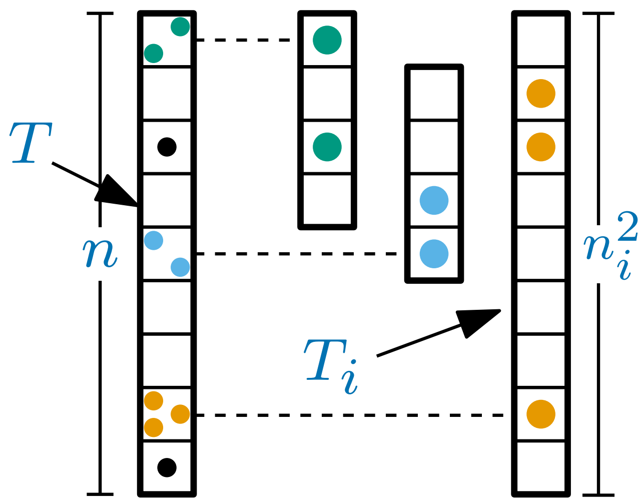
(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

How much space does this use?

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

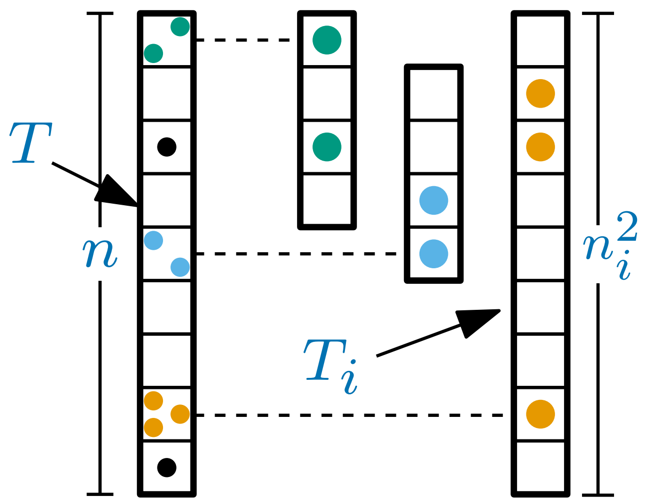
a) T has more than n collisions

b) some T_i has a collision

How much space does this use?

The size of T is $O(n)$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

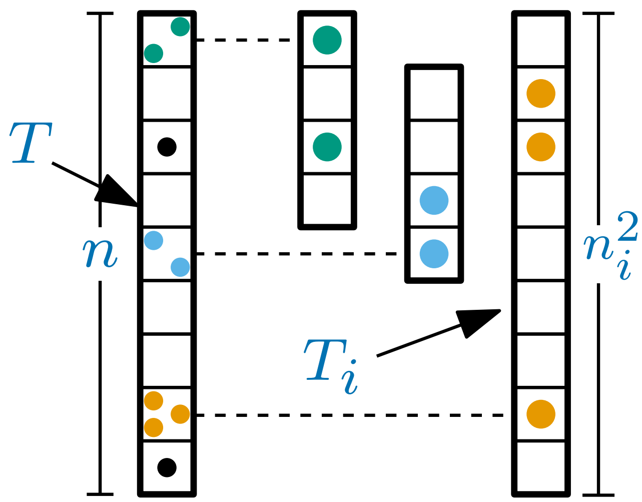
b) some T_i has a collision

How much space does this use?

The size of T is $O(n)$

The size of T_i is $O({n_i}^2)$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

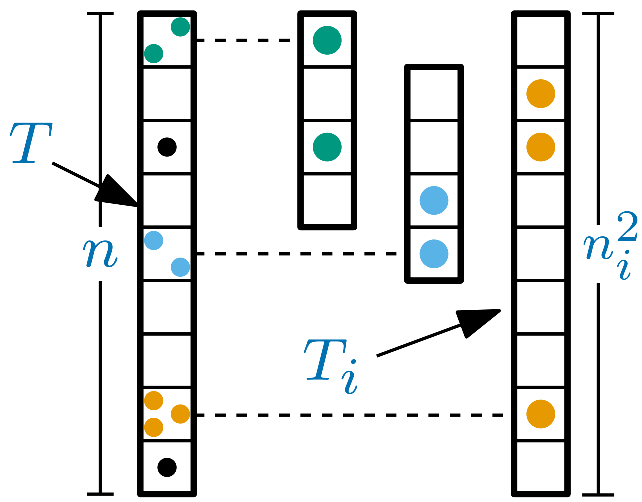
How much space does this use?

The size of T is $O(n)$

The size of T_i is $O({n_i}^2)$

Storing h_i uses $O(1)$ space

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n
using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

How much space does this use?

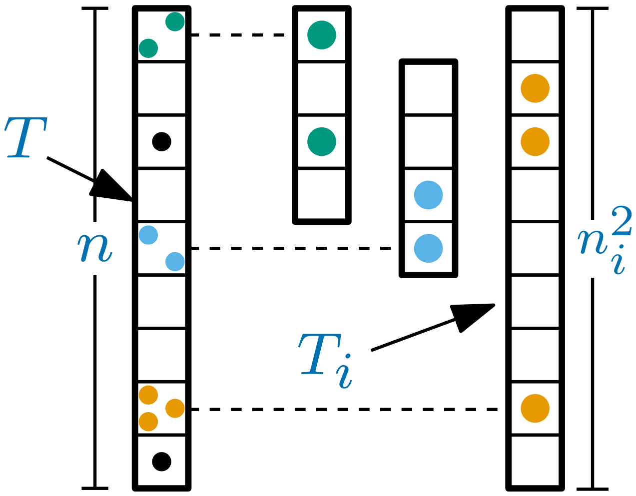
The size of T is $O(n)$

The size of T_i is $O({n_i}^2)$

Storing h_i uses $O(1)$ space

So the total space is...

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

How much space does this use?

The size of T is $O(n)$

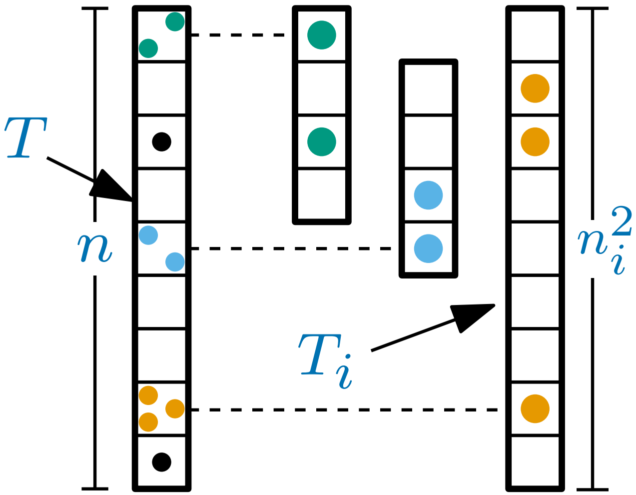
The size of T_i is $O({n_i}^2)$

Storing h_i uses $O(1)$ space

So the total space is...

$$O(n)+\sum_i O(n_i^2)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

How much space does this use?

The size of T is $O(n)$

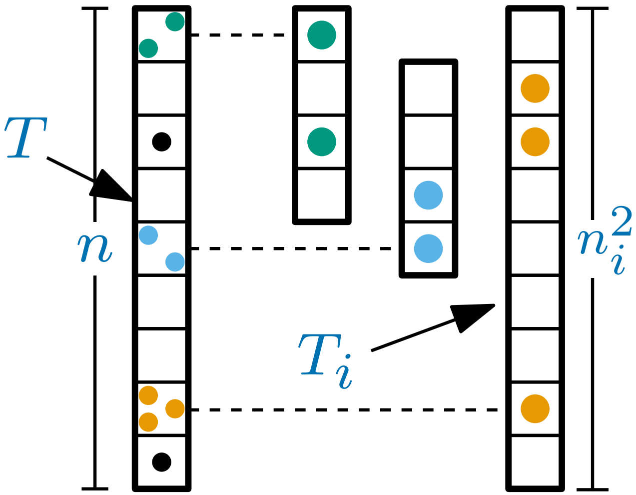
The size of T_i is $O({n_i}^2)$

Storing h_i uses $O(1)$ space

So the total space is...

$$O(n)+\sum_i O(n_i^2) = O(n)+O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n
using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

How much space does this use?

The size of T is $O(n)$

The size of T_i is $O({n_i}^2)$

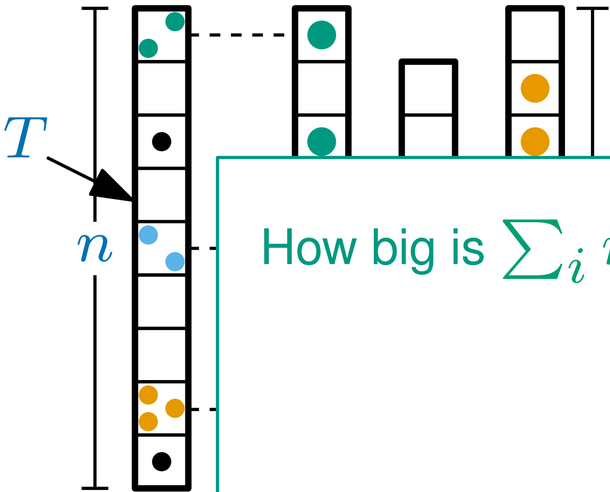
Storing h_i uses $O(1)$ space

how big is this?

So the total space is...

$$O(n)+\sum_i O(n_i^2) = O(n)+O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

able T_i

How big is $\sum_i n_i^2$?

How mu

The s

The s

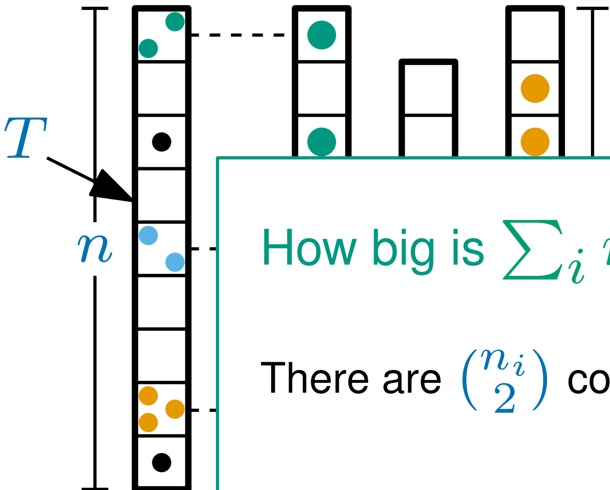
Storing h_i uses $O(1)$ space

how big is this?

So the total space is...

$$O(n)+\sum_i O(n_i^2) = O(n)+O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

able T_i

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$

How mu

The s

The s

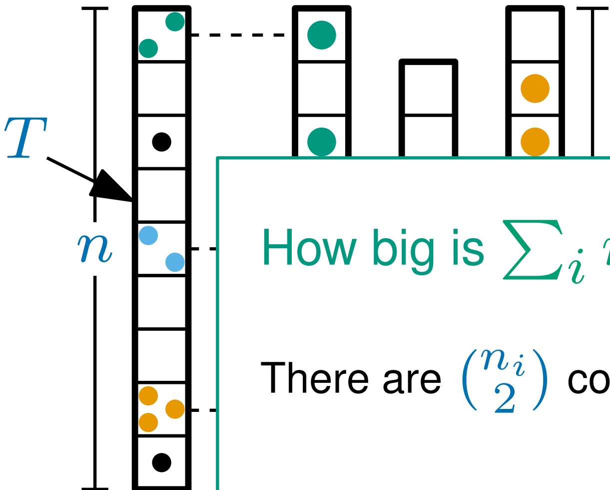
Storing h_i uses $O(1)$ space

how big is this?

So the total space is...

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

able T_i

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in T

How mu

The s

The s

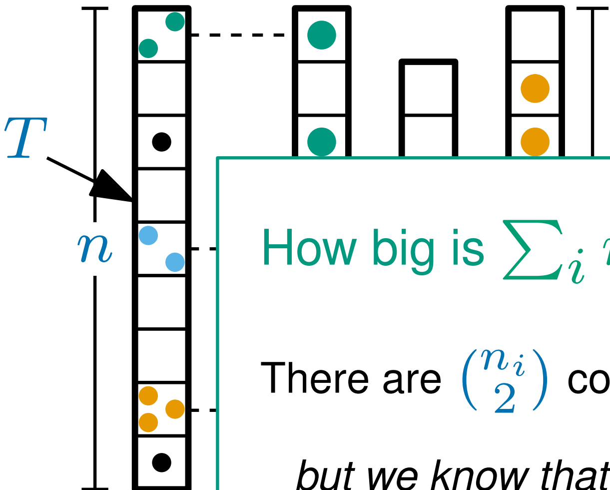
Storing h_i uses $O(1)$ space

how big is this?

So the total space is...

$$O(n)+\sum_i O(n_i^2) = O(n)+O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

able T_i

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in T

but we know that there are at most n collisions in T . . .

How mu

The s

The s

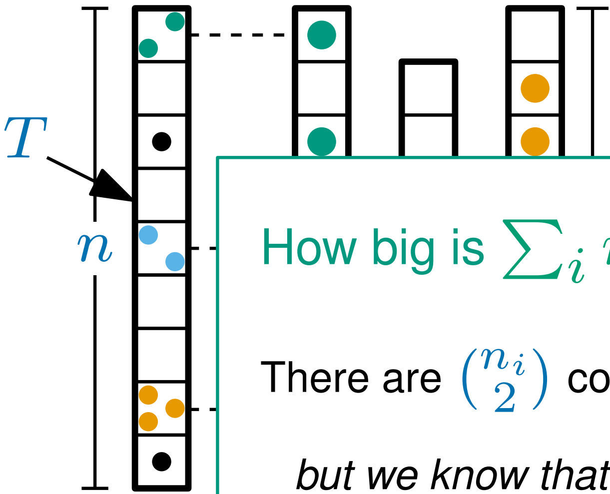
Storing h_i uses $O(1)$ space

how big is this?

So the total space is. . .

$$O(n)+\sum_i O(n_i^2) = O(n)+O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

able T_i

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in T

but we know that there are at most n collisions in T . . .

$$\sum_i \binom{n_i}{2} \leqslant n$$

How mu

The s

The s

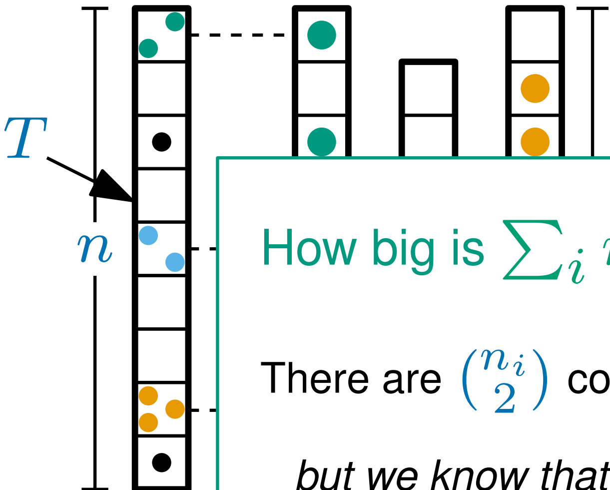
Storing h_i uses $O(1)$ space

So the total space is. . .

how big is this?

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

able T_i

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in T

but we know that there are at most n collisions in T . . .

$$\sum_i \frac{n_i^2}{4} \leqslant \sum_i \binom{n_i}{2} \leqslant n$$

How mu

The s

The s

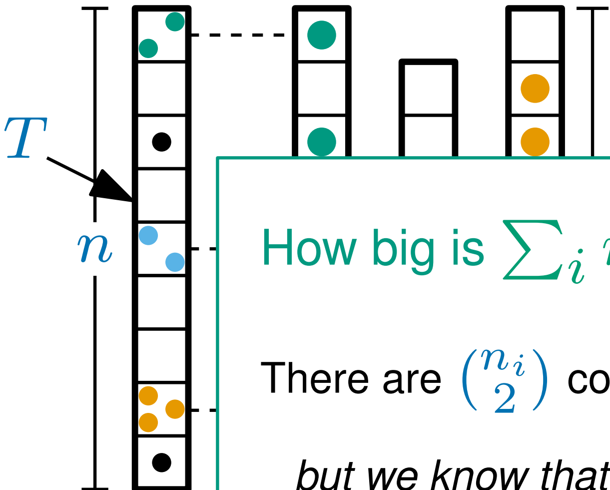
Storing h_i uses $O(1)$ space

how big is this?

So the total space is. . .

$$O(n)+\sum_i O(n_i^2) = O(n)+O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

able T_i

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in T

but we know that there are at most n collisions in T . . .

$$\sum_i \frac{n_i^2}{4} \leqslant \sum_i \binom{n_i}{2} \leqslant n$$

$$\binom{n_i}{2} = \frac{n_i(n_i-1)}{2} \geqslant \frac{n_i^2}{4}$$

How mu

The s

The s

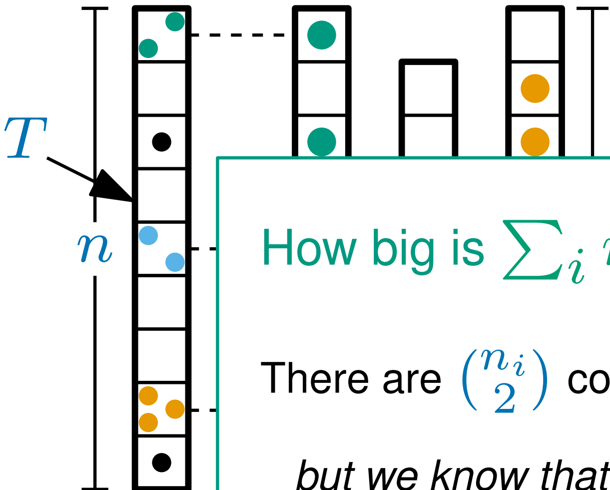
Storing h_i uses $O(1)$ space

how big is this?

So the total space is. . .

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

able T_i

How mu

The s

The s

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in T

but we know that there are at most n collisions in T . . .

$$\sum_i \frac{n_i^2}{4} \leqslant \sum_i \binom{n_i}{2} \leqslant n$$

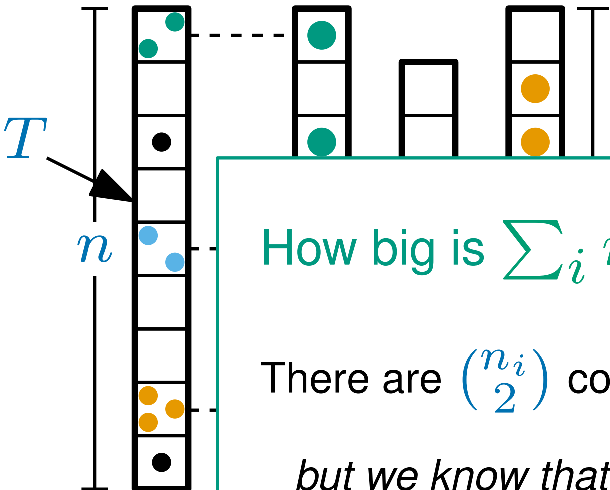
Storing h_i uses $O(1)$ space

how big is this?

So the total space is. . .

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

able T_i

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in T

but we know that there are at most n collisions in T ...

$$\sum_i \frac{n_i^2}{4} \leqslant \sum_i \binom{n_i}{2} \leqslant n \quad \text{or} \quad \sum_i n_i^2 \leqslant 4n$$

How mu

The s

The s

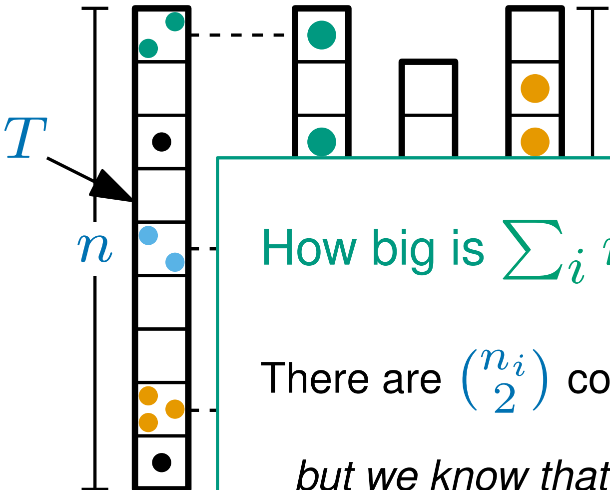
Storing h_i uses $O(1)$ space

how big is this?

So the total space is...

$$O(n)+\sum_i O(n_i^2) = O(n)+O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

able T_i

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in T

but we know that there are at most n collisions in T . . .

$$\sum_i \frac{n_i^2}{4} \leqslant \sum_i \binom{n_i}{2} \leqslant n \quad \text{or} \quad \sum_i n_i^2 \leqslant 4n$$

How mu

The s

The s

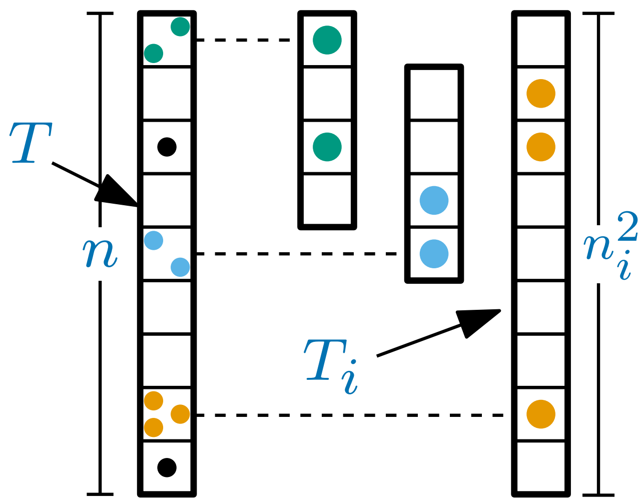
Storing h_i uses $O(1)$ space

how big is this?

So the total space is. . .

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right) = O(n)$$

Perfect Hashing - Space usage



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

How much space does this use?

The size of T is $O(n)$

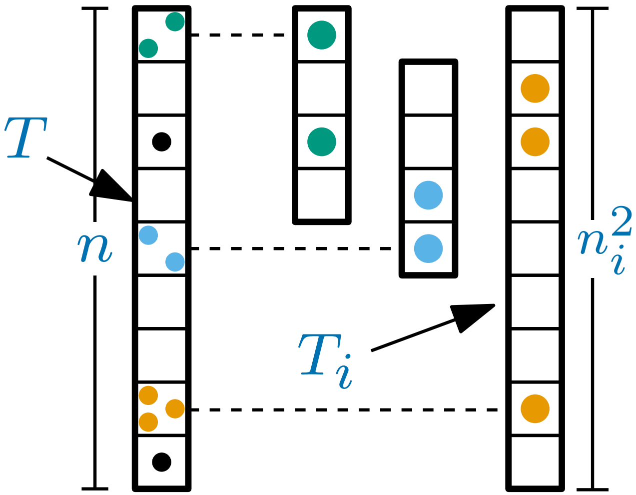
The size of T_i is $O({n_i}^2)$

Storing h_i uses $O(1)$ space

So the total space is...

$$O(n)+\sum_i O(n_i^2) = O(n)+O\left(\sum_i n_i^2\right) = O(n)$$

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

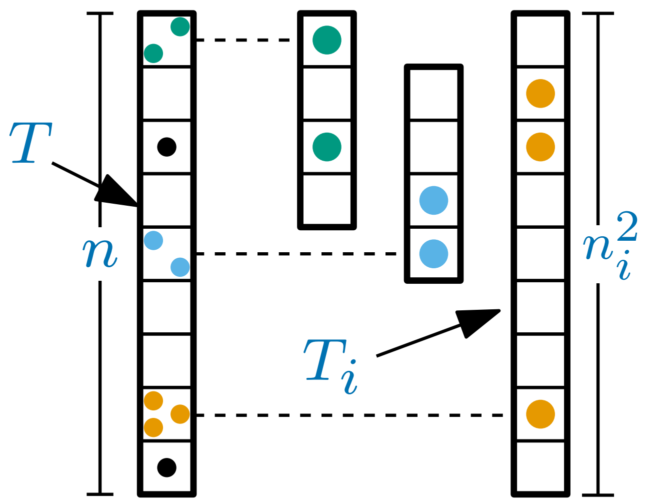
Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n
using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

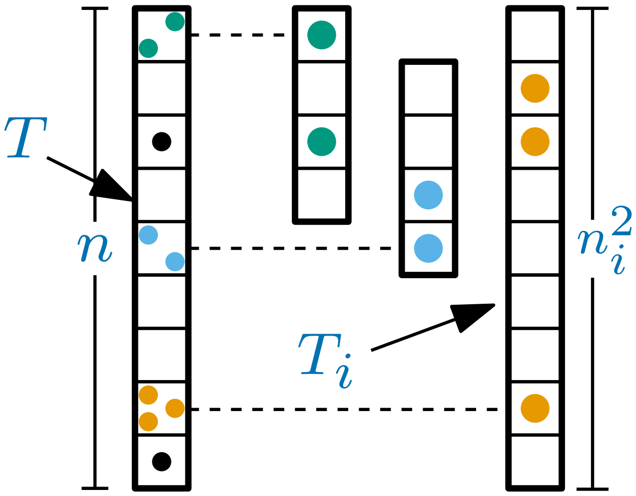
a) T has more than n collisions

b) some T_i has a collision

The expected construction time for T is $O(n)$

(we considered this on a previous slide)

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n
using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

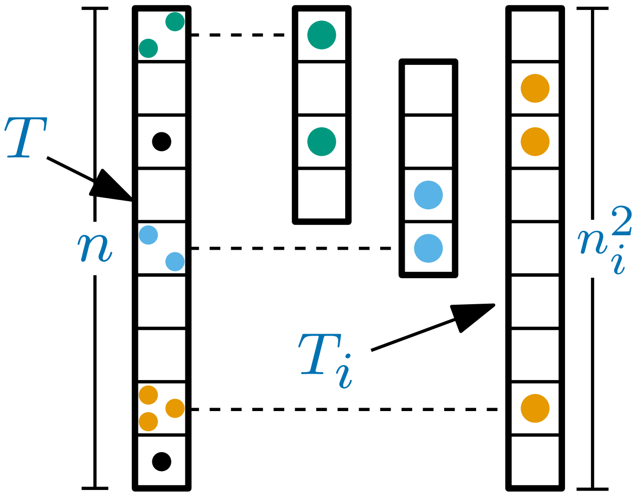
b) some T_i has a collision

The expected construction time for T is $O(n)$

(we considered this on a previous slide)

The expected construction time for *each* T_i is $O({n_i}^2)$

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

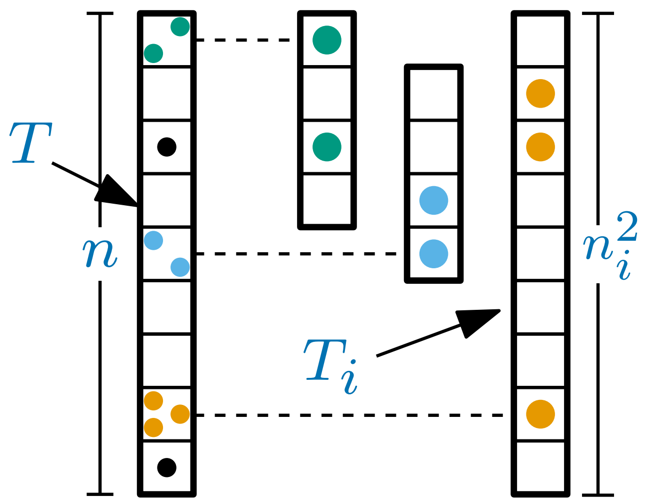
The expected construction time for T is $O(n)$

(we considered this on a previous slide)

The expected construction time for *each* T_i is $O({n_i}^2)$

- we insert n_i items into a table of size $m = n_i^2$
- then repeat if there was a collision

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

The expected construction time for T is $O(n)$

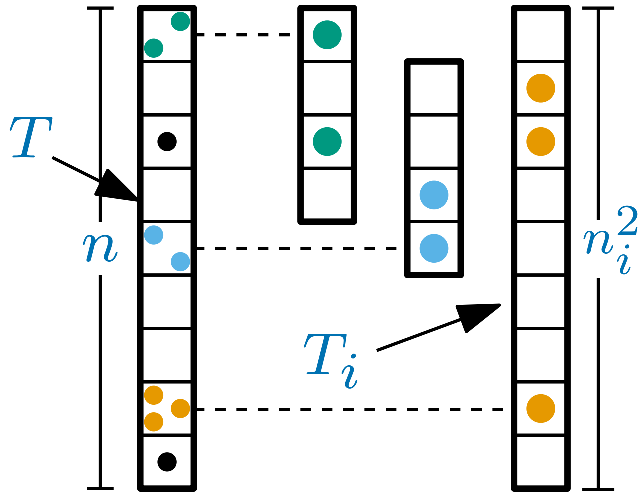
(we considered this on a previous slide)

The expected construction time for *each* T_i is $O({n_i}^2)$

- we insert n_i items into a table of size $m = n_i^2$
- then repeat if there was a collision

(we also considered this on a previous slide)

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n
using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

The expected construction time for T is $O(n)$

(we considered this on a previous slide)

The expected construction time for *each* T_i is $O({n_i}^2)$

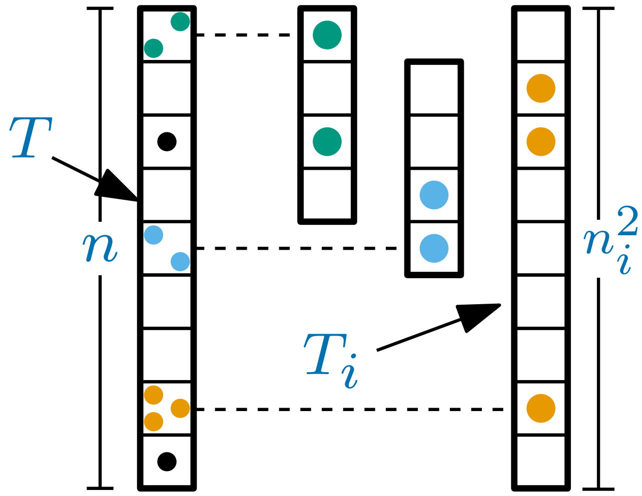
- we insert n_i items into a table of size $m = n_i^2$
- then repeat if there was a collision

(we also considered this on a previous slide)

The overall expected constuction time is therefore:

$$\mathbb{E}(\text{construction time}) = \mathbb{E}\left(\text{construction time of } T + \sum_i \text{construction time of } T_i\right)$$

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

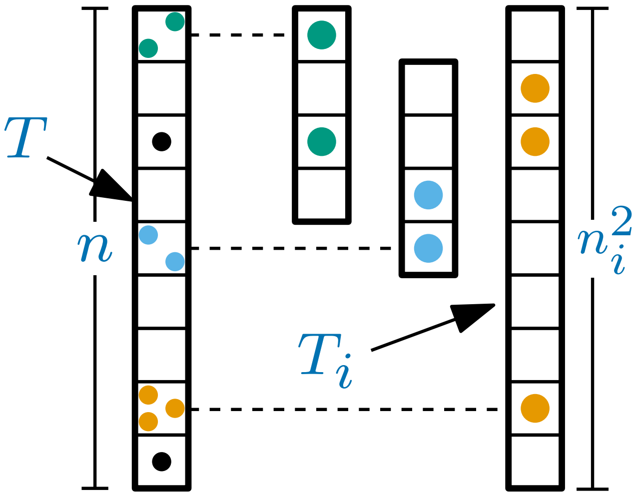
The expected construction time for T is $O(n)$

The expected construction time for *each* T_i is $O({n_i}^2)$

The overall expected construction time is therefore:

$$\mathbb{E}(\text{construction time}) = \mathbb{E}\left(\text{construction time of } T + \sum_i \text{construction time of } T_i\right)$$

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

The expected construction time for T is $O(n)$

The expected construction time for *each* T_i is $O({n_i}^2)$

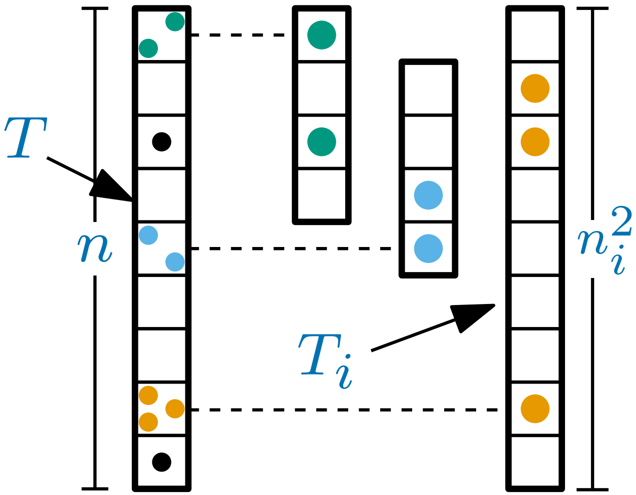
The overall expected construction time is therefore:

$$\mathbb{E}(\text{construction time}) = \mathbb{E}\left(\text{construction time of } T + \sum_i \text{construction time of } T_i\right)$$

$$= \mathbb{E}(\text{construction time of } T) + \sum_i \mathbb{E}(\text{construction time of } T_i)$$

University of BRISTOL

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

The expected construction time for T is $O(n)$

The expected construction time for *each* T_i is $O({n_i}^2)$

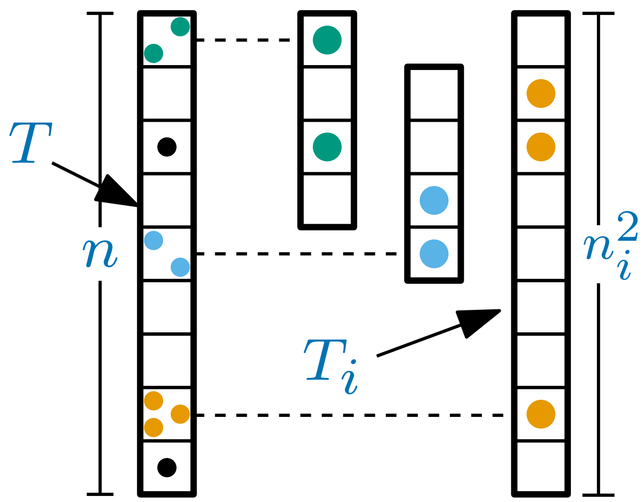
The overall expected construction time is therefore:

$$\mathbb{E}(\text{construction time}) = \mathbb{E}\left(\text{construction time of } T + \sum_i \text{construction time of } T_i\right)$$

$$= \mathbb{E}(\text{construction time of } T) + \sum_i \mathbb{E}(\text{construction time of } T_i)$$

$$= O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n
using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i
of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

The expected construction time for T is $O(n)$

The expected construction time for *each* T_i is $O({n_i}^2)$

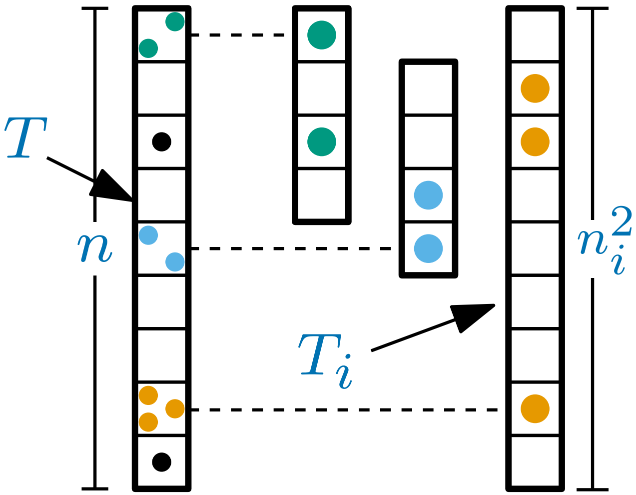
The overall expected construction time is therefore:

$$\mathbb{E}(\text{construction time}) = \mathbb{E}\left(\text{construction time of } T + \sum_i \text{construction time of } T_i\right)$$

$$= \mathbb{E}(\text{construction time of } T) + \sum_i \mathbb{E}(\text{construction time of } T_i)$$

$$= O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right) = O(n)$$

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

The expected construction time for T is $O(n)$

The expected construction time for *each* T_i is $O({n_i}^2)$

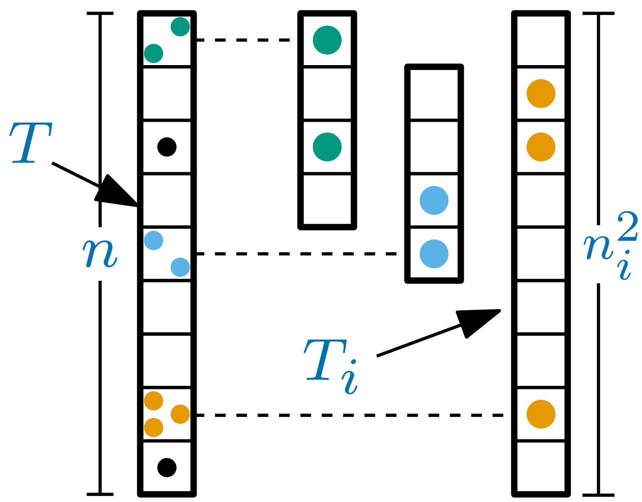
The overall expected construction time is therefore:

$$\mathbb{E}(\text{construction time}) = \mathbb{E}\left(\text{construction time of } T + \boxed{\sum_i n_i^2 \leqslant 4n} \text{ of } T_i\right)$$

$$= \mathbb{E}(\text{construction time of } T) + \sum_i \mathbb{E}(\text{construction time of } T_i)$$

$$= O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right) = O(n)$$

Perfect Hashing - Expected construction time



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

The expected construction time for T is $O(n)$

The expected construction time for *each* T_i is $O({n_i}^2)$

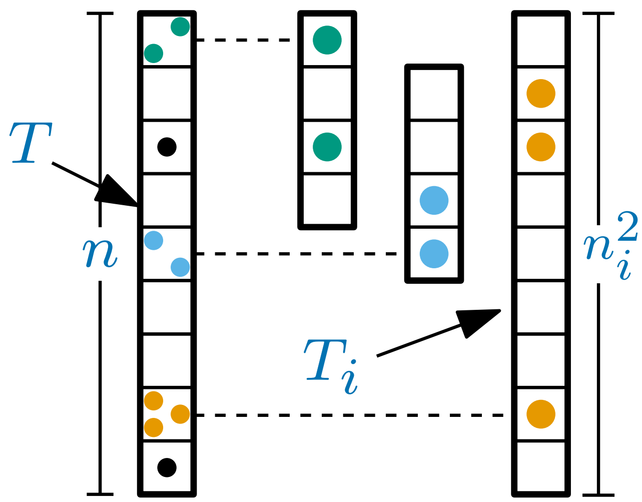
The overall expected construction time is therefore:

$$\mathbb{E}(\text{construction time}) = \mathbb{E}\left(\text{construction time of } T + \sum_i \text{construction time of } T_i\right)$$

$$= \mathbb{E}(\text{construction time of } T) + \sum_i \mathbb{E}(\text{construction time of } T_i)$$

$$= O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right) = O(n)$$

Perfect Hashing - Summary



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

Theorem

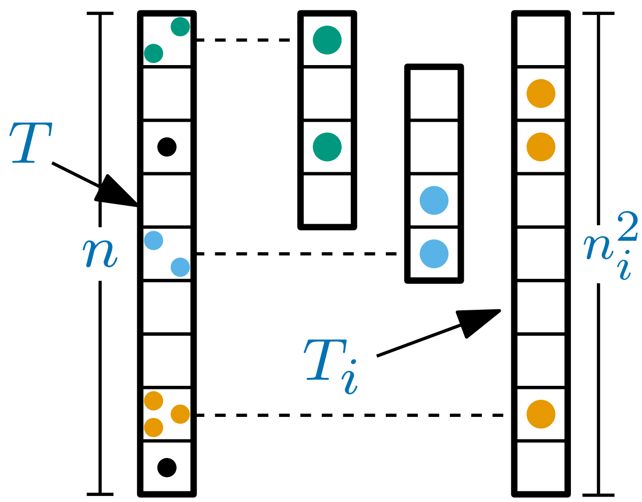
The FKS hashing scheme:

- Has no collisions
- Every lookup takes $O(1)$ *worst-case* time,
- Uses $O(n)$ space,
- Can be built in $O(n)$ expected time.

The look-up time is always $O(1)$

1. Compute $i = h(x)$ (x is the key)
2. Compute $j = h_i(x)$
3. The item is in $T_i[j]$

Perfect Hashing - Summary



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

THEOREM

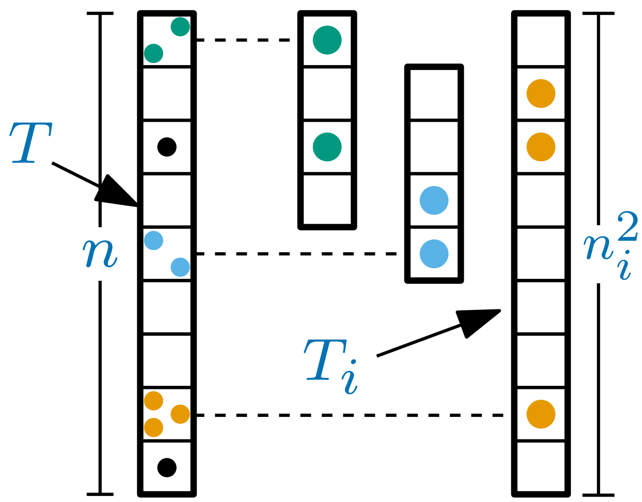
The FKS hashing scheme:

- Has no collisions
- Every lookup takes $O(1)$ *worst-case* time,
- Uses $O(n)$ space,
- Can be built in $O(n)$ expected time.

The look-up time is always $O(1)$

1. Compute $i = h(x)$ (x is the key)
2. Compute $j = h_i(x)$
3. The item is in $T_i[j]$

Perfect Hashing - Summary



Step 1: Insert everything into a hash table, T, of size n using a weakly universal (w.u.) hash function, h

Step 2: The n_i items in $T[i]$ are inserted into another hash table T_i of size n_i^2 using w.u hash function h_i

(Step 3) *Immediately repeat if either*

a) T has more than n collisions

b) some T_i has a collision

THEOREM

The FKS hashing scheme:

- Has no collisions
- Every lookup takes $O(1)$ *worst-case* time,
- Uses $O(n)$ space,
- Can be built in $O(n)$ expected time.

The look-up time is always $O(1)$

1. Compute $i = h(x)$ (x is the key)
2. Compute $j = h_i(x)$
3. The item is in $T_i[j]$