

Lecture 11: Universal and Perfect Hashing

Michael Dinitz

September 30, 2025

601.433/633 Introduction to Algorithms

Introduction

Another approach to dictionaries (insert, lookup, delete): hashing

- ▶ Can improve operations to $O(1)$, but with many caveats!

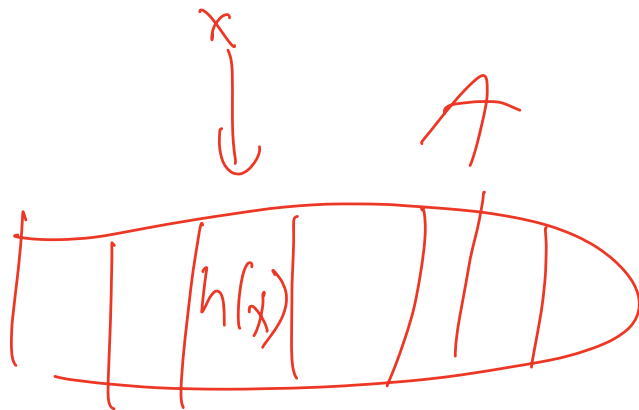
Should have seen some discussion of hashing in data structures. Also in CLRS.

- ▶ Separate chaining vs. open addressing

Today: discussion of caveats, more advanced versions of hashing (universal and perfect)

Hashing Basics

- ▶ Keys from universe U (think very large)
- ▶ Set $S \subseteq U$ of keys we actually care about (think relatively small). $|S| = N$.
- ▶ Hash table A (array) of size M .
- ▶ Hash function $h: U \rightarrow [M]$
 - ▶ $[M] = \{1, 2, \dots, M\}$
- ▶ Idea: store x in $A[h(x)]$

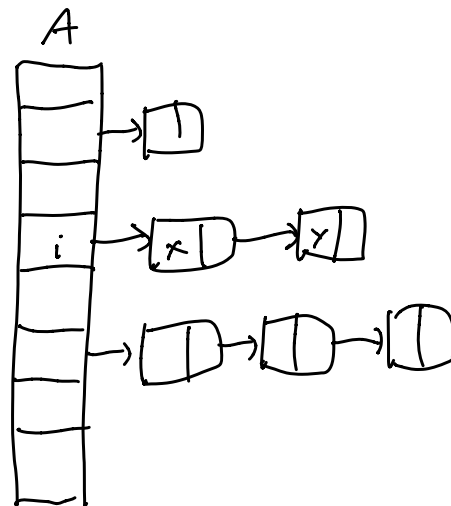


Hashing Basics

- ▶ Keys from universe \mathbf{U} (think very large)
- ▶ Set $\mathbf{S} \subseteq \mathbf{U}$ of keys we actually care about (think relatively small). $|\mathbf{S}| = \mathbf{N}$.
- ▶ Hash table \mathbf{A} (array) of size \mathbf{M} .
- ▶ Hash function $\mathbf{h} : \mathbf{U} \rightarrow [\mathbf{M}]$
 - ▶ $[\mathbf{M}] = \{1, 2, \dots, \mathbf{M}\}$
- ▶ Idea: store \mathbf{x} in $\mathbf{A}[\mathbf{h}(\mathbf{x})]$

One more component: *collision resolution*

- ▶ Today: *separate chaining*
- ▶ $\mathbf{A}[\mathbf{i}]$ is a linked list containing all \mathbf{x} inserted where $\mathbf{h}(\mathbf{x}) = \mathbf{i}$.



Dictionary Operations

Lookup(x): Walk down the list at $A[h(x)]$ until we find x (or walk to the end of the list)

Insert(x): Add x to the beginning of the list at $A[h(x)]$.

Delete(x): Walk down the list at $A[h(x)]$ until we find x . Remove it from the list.

Dictionary Operations

Lookup(x): Walk down the list at $A[h(x)]$ until we find x (or walk to the end of the list)

Insert(x): Add x to the beginning of the list at $A[h(x)]$.

Delete(x): Walk down the list at $A[h(x)]$ until we find x . Remove it from the list.

Question: What should hash function be?

Dictionary Operations

Lookup(x): Walk down the list at $A[h(x)]$ until we find x (or walk to the end of the list)

Insert(x): Add x to the beginning of the list at $A[h(x)]$.

Delete(x): Walk down the list at $A[h(x)]$ until we find x . Remove it from the list.

Question: What should hash function be?

Properties we want:

Dictionary Operations

Lookup(x): Walk down the list at $A[h(x)]$ until we find x (or walk to the end of the list)

Insert(x): Add x to the beginning of the list at $A[h(x)]$.

Delete(x): Walk down the list at $A[h(x)]$ until we find x . Remove it from the list.

Question: What should hash function be?

Properties we want:

- ▶ Few collisions. Time of lookup, delete for x is $O(\text{length of list at } A[h(x)])$.

Dictionary Operations

Lookup(x): Walk down the list at $A[h(x)]$ until we find x (or walk to the end of the list)

Insert(x): Add x to the beginning of the list at $A[h(x)]$.

Delete(x): Walk down the list at $A[h(x)]$ until we find x . Remove it from the list.

Question: What should hash function be?

Properties we want:

- ▶ Few collisions. Time of lookup, delete for x is $O(\text{length of list at } A[h(x)])$.
- ▶ Small M . Ideally, $M = O(N)$.

Dictionary Operations

Lookup(x): Walk down the list at $A[h(x)]$ until we find x (or walk to the end of the list)

Insert(x): Add x to the beginning of the list at $A[h(x)]$.

Delete(x): Walk down the list at $A[h(x)]$ until we find x . Remove it from the list.

Question: What should hash function be?

Properties we want:

- ▶ Few collisions. Time of lookup, delete for x is $O(\text{length of list at } A[h(x)])$.
- ▶ Small M . Ideally, $M = O(N)$.
- ▶ h fast to compute.

Bad News

Theorem

For any hash function h , if $|U| \geq (N - 1)M + 1$, then there exists a set S of N elements that all hash to the same location.

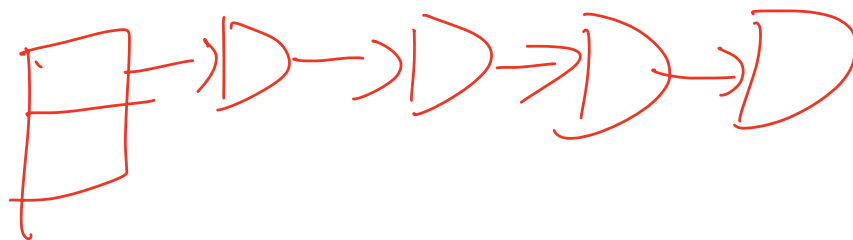
Bad News

Theorem

For any hash function h , if $|U| \geq (N - 1)M + 1$, then there exists a set S of N elements that all hash to the same location.

Proof.

Pigeonhole principle / contradiction / contrapositive. □



Bad News

Theorem

For any hash function h , if $|U| \geq (N - 1)M + 1$, then there exists a set S of N elements that all hash to the same location.

Proof.

Pigeonhole principle / contradiction / contrapositive. □

So worst case behavior always bad! How can we get around this?

Bad News

Theorem

For any hash function h , if $|U| \geq (N - 1)M + 1$, then there exists a set S of N elements that all hash to the same location.

Proof.

Pigeonhole principle / contradiction / contrapositive. □

So worst case behavior always bad! How can we get around this?

- ▶ Option 1: don't worry about it, hope world isn't adversarial.

Bad News

Theorem

For any hash function h , if $|U| \geq (N - 1)M + 1$, then there exists a set S of N elements that all hash to the same location.

Proof.

Pigeonhole principle / contradiction / contrapositive. □

So worst case behavior always bad! How can we get around this?

- ▶ Option 1: don't worry about it, hope world isn't adversarial.
- ▶ Option 2: Randomness! *Random function* $h: U \rightarrow [M]$
 - ▶ For each $x \in U$, choose $y \in [M]$ uniformly at random and set $h(x) = y$.
 - ▶ Hopefully good behavior in expectation.

Bad News

Theorem

For any hash function h , if $|U| \geq (N - 1)M + 1$, then there exists a set S of N elements that all hash to the same location.

Proof.

Pigeonhole principle / contradiction / contrapositive. □

So worst case behavior always bad! How can we get around this?

- ▶ Option 1: don't worry about it, hope world isn't adversarial.
- ▶ Option 2: Randomness! *Random function* $h: U \rightarrow [M]$
 - ▶ For each $x \in U$, choose $y \in [M]$ uniformly at random and set $h(x) = y$.
 - ▶ Hopefully good behavior in expectation.
 - ▶ Problem: How can we store/remember/create h ?

Universal Hashing

Definition

A probability distribution \mathbf{H} over hash functions $\{\mathbf{h} : \mathbf{U} \rightarrow [\mathbf{M}]\}$ is *universal* if

$$\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M$$

for all $\mathbf{x}, \mathbf{y} \in \mathbf{U}$ with $\mathbf{x} \neq \mathbf{y}$.

Universal Hashing

Definition

A probability distribution \mathbf{H} over hash functions $\{\mathbf{h} : \mathbf{U} \rightarrow [\mathbf{M}]\}$ is *universal* if

$$\Pr_{\mathbf{h} \sim \mathbf{H}}[\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] \leq 1/\mathbf{M}$$

for all $\mathbf{x}, \mathbf{y} \in \mathbf{U}$ with $\mathbf{x} \neq \mathbf{y}$.

Clearly satisfied by \mathbf{H} = uniform distribution over all hash functions

Universal Hashing

Definition

A probability distribution \mathbf{H} over hash functions $\{\mathbf{h} : \mathbf{U} \rightarrow [\mathbf{M}]\}$ is *universal* if

$$\Pr_{\mathbf{h} \sim \mathbf{H}} [\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] \leq 1/\mathbf{M}$$

for all $\mathbf{x}, \mathbf{y} \in \mathbf{U}$ with $\mathbf{x} \neq \mathbf{y}$.

Clearly satisfied by \mathbf{H} = uniform distribution over all hash functions

Theorem

If \mathbf{H} is universal, then for every set $\mathbf{S} \subseteq \mathbf{U}$ with $|\mathbf{S}| = \mathbf{N}$ and for every $\mathbf{x} \in \mathbf{U}$, the expected number of collisions (when we draw \mathbf{h} from \mathbf{H}) between \mathbf{x} and elements of \mathbf{S} is at most \mathbf{N}/\mathbf{M} .

Universal Hashing

Definition

A probability distribution \mathbf{H} over hash functions $\{h : \mathbf{U} \rightarrow [\mathbf{M}]\}$ is *universal* if

$$\Pr_{h \sim \mathbf{H}}[h(\mathbf{x}) = h(\mathbf{y})] \leq 1/\mathbf{M}$$

for all $\mathbf{x}, \mathbf{y} \in \mathbf{U}$ with $\mathbf{x} \neq \mathbf{y}$.

Clearly satisfied by \mathbf{H} = uniform distribution over all hash functions

Theorem

If \mathbf{H} is universal, then for every set $\mathbf{S} \subseteq \mathbf{U}$ with $|\mathbf{S}| = \mathbf{N}$ and for every $\mathbf{x} \in \mathbf{U}$, the expected number of collisions (when we draw h from \mathbf{H}) between \mathbf{x} and elements of \mathbf{S} is at most \mathbf{N}/\mathbf{M} .

So Lookup(\mathbf{x}) and Delete(\mathbf{x}) have expected time $O(\mathbf{N}/\mathbf{M})$.

\implies If $\mathbf{M} = \Omega(\mathbf{N})$, operations in $O(1)$ time!

Main Proof

Theorem

If \mathbf{H} is universal, then for every set $\mathbf{S} \subseteq \mathbf{U}$ with $|\mathbf{S}| = \mathbf{N}$ and for every $\mathbf{x} \in \mathbf{U}$, the expected number of collisions (when we draw \mathbf{h} from \mathbf{H}) between \mathbf{x} and elements of \mathbf{S} is at most \mathbf{N}/\mathbf{M} .

Proof.

$$\text{Let } \mathbf{C}_{xy} = \begin{cases} 1 & \text{if } \mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

$$\implies \mathbf{E}[\mathbf{C}_{xy}] = \Pr_{\mathbf{h} \sim \mathbf{H}}[\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] \leq 1/\mathbf{M}$$

Main Proof

Theorem

If \mathbf{H} is universal, then for every set $\mathbf{S} \subseteq \mathbf{U}$ with $|\mathbf{S}| = \mathbf{N}$ and for every $\mathbf{x} \in \mathbf{U}$, the expected number of collisions (when we draw \mathbf{h} from \mathbf{H}) between \mathbf{x} and elements of \mathbf{S} is at most \mathbf{N}/\mathbf{M} .

Proof.

$$\text{Let } \mathbf{C}_{xy} = \begin{cases} 1 & \text{if } \mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

$$\implies E[\mathbf{C}_{xy}] = \Pr_{\mathbf{h} \sim \mathbf{H}}[\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] \leq 1/\mathbf{M}$$

Number of collisions between \mathbf{x} and \mathbf{S} is exactly $\sum_{\mathbf{y} \in \mathbf{S}} \mathbf{C}_{xy}$

$$\implies E\left[\sum_{\mathbf{y} \in \mathbf{S}} \mathbf{C}_{xy}\right] = \sum_{\mathbf{y} \in \mathbf{S}} E[\mathbf{C}_{xy}] \leq \sum_{\mathbf{y} \in \mathbf{S}} \frac{1}{\mathbf{M}} = \mathbf{N}/\mathbf{M}$$

□

Main Corollary

Corollary

If \mathbf{H} is universal, then for any sequence of \mathbf{L} insert, lookup, and delete operations in which there are at most $\mathbf{O}(\mathbf{M})$ elements in the system at any time, the expected total cost of the whole sequence is only $\mathbf{O}(\mathbf{L})$ (assuming \mathbf{h} takes constant time to compute).

Main Corollary

Corollary

If H is universal, then for any sequence of L insert, lookup, and delete operations in which there are at most $O(M)$ elements in the system at any time, the expected total cost of the whole sequence is only $O(L)$ (assuming h takes constant time to compute).

Proof.

By theorem, each operation $O(1)$ in expectation. Total time is sum: linearity of expectations. □

Main Corollary

Corollary

If H is universal, then for any sequence of L insert, lookup, and delete operations in which there are at most $O(M)$ elements in the system at any time, the expected total cost of the whole sequence is only $O(L)$ (assuming h takes constant time to compute).

Proof.

By theorem, each operation $O(1)$ in expectation. Total time is sum: linearity of expectations. □

So universal distributions are great. Can we construct them?

Universal Hash Families

Definition

If \mathcal{H} is universal and is a uniform distribution over a set of functions $\{h_1, h_2, \dots\}$, then that set is called a *universal hash family*.

Often use \mathcal{H} to refer to both set of functions and uniform distribution over it.

Universal Hash Families

Definition

If \mathbf{H} is universal and is a uniform distribution over a set of functions $\{\mathbf{h}_1, \mathbf{h}_2, \dots\}$, then that set is called a *universal hash family*.

Often use \mathbf{H} to refer to both set of functions and uniform distribution over it.

Notation:

- ▶ $\mathbf{U} = \{0, 1\}^u$ (so $|\mathbf{U}| = 2^u$)
- ▶ $\mathbf{M} = 2^b$, so an index to \mathbf{A} is an element of $\{0, 1\}^b$

Universal Hash Families

Definition

If \mathbf{H} is universal and is a uniform distribution over a set of functions $\{\mathbf{h}_1, \mathbf{h}_2, \dots\}$, then that set is called a *universal hash family*.

Often use \mathbf{H} to refer to both set of functions and uniform distribution over it.

Notation:

- ▶ $\mathbf{U} = \{0, 1\}^u$ (so $|\mathbf{U}| = 2^u$)
- ▶ $\mathbf{M} = 2^b$, so an index to \mathbf{A} is an element of $\{0, 1\}^b$

Construction: $\mathbf{H} = \{0, 1\}^{b \times u}$, i.e., \mathbf{H} is all $b \times u$ binary matrices

- ▶ Each $\mathbf{h} \in \mathbf{H}$ is a (linear) function from \mathbf{U} to $[\mathbf{M}]$:
 $\mathbf{h}(\mathbf{x}) = \mathbf{h}\mathbf{x} \in \{0, 1\}^b$ (all operations mod 2)

$$\begin{array}{c} \mathbf{h} \quad \mathbf{x} \quad \mathbf{h}(\mathbf{x}) \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \end{array}$$

Universality

Theorem

***H** is a universal hash family: $\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M$ for all $x \neq y \in \{0, 1\}^u$.*

Universality

Theorem

H is a universal hash family: $\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M$ for all $x \neq y \in \{0, 1\}^u$.

Proof.

Matrix multiplication: $h(x) = hx = \sum_{i: x_i=1} h^i$ (where h^i is i 'th column of h).

Universality

Theorem

H is a universal hash family: $\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M$ for all $x \neq y \in \{0, 1\}^u$.

Proof.

Matrix multiplication: $h(x) = hx = \sum_{i: x_i=1} h^i$ (where h^i is i 'th column of h).

Since $x \neq y$, there is i s.t. $x_i \neq y_i$. WLOG, $x_i = 0$ and $y_i = 1$.

Universality

$$\begin{bmatrix} 0 & 1 & 1 & ? & 1 & 0 \\ 1 & 0 & 1 & ? & 1 & 0 \\ 1 & 0 & 1 & ? & 1 & 0 \end{bmatrix}$$

$$h_x = 11001111$$
$$h'_y = 10111100$$
$$\hline 01110011$$

Theorem

H is a universal hash family: $\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M$ for all $x \neq y \in \{0, 1\}^u$.

Proof.

Matrix multiplication: $h(x) = hx = \sum_{i: x_i=1} h^i$ (where h^i is i 'th column of h).

Since $x \neq y$, there is i s.t. $x_i \neq y_i$. WLOG, $x_i = 0$ and $y_i = 1$.

Draw all entries of h except for h^i . Let $h' = h$ with h^i all 0's

- $h(x) = h'(x)$ already fixed.

Universality

Theorem

H is a universal hash family: $\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M$ for all $x \neq y \in \{0, 1\}^u$.

Proof.

Matrix multiplication: $h(x) = hx = \sum_{i: x_i=1} h^i$ (where h^i is i 'th column of h).

Since $x \neq y$, there is i s.t. $x_i \neq y_i$. WLOG, $x_i = 0$ and $y_i = 1$.

Draw all entries of h except for h^i . Let $h' = h$ with h^i all 0's

- ▶ $h(x) = h'(x)$ already fixed.
- ▶ If $h(y) = h(x)$, then h^i must equal $h(x) - h'(y)$

Universality

Theorem

H is a universal hash family: $\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M$ for all $x \neq y \in \{0, 1\}^u$.

Proof.

Matrix multiplication: $h(x) = hx = \sum_{i: x_i=1} h^i$ (where h^i is i 'th column of h).

Since $x \neq y$, there is i s.t. $x_i \neq y_i$. WLOG, $x_i = 0$ and $y_i = 1$.

Draw all entries of h except for h^i . Let $h' = h$ with h^i all 0's

- ▶ $h(x) = h'(x)$ already fixed.
- ▶ If $h(y) = h(x)$, then h^i must equal $h(x) - h'(y)$
- ▶ Happens with probability exactly $1/2^b = 1/M$



Perfect Hashing

Suppose you know S , never changes.

- ▶ Build table, then do lookups. Like a real dictionary!
- ▶ Care more about time to do lookup than time to build dictionary

Perfect Hashing

Suppose you know \mathbf{S} , never changes.

- ▶ Build table, then do lookups. Like a real dictionary!
- ▶ Care more about time to do lookup than time to build dictionary

Obvious approaches:

- ▶ Sorted array: lookups $O(\log N)$
- ▶ Balanced search tree: $O(\log N)$

Perfect Hashing

Suppose you know \mathbf{S} , never changes.

- ▶ Build table, then do lookups. Like a real dictionary!
- ▶ Care more about time to do lookup than time to build dictionary

Obvious approaches:

- ▶ Sorted array: lookups $O(\log N)$
- ▶ Balanced search tree: $O(\log N)$

Can we do better with hashing?

Perfect Hashing

Suppose you know \mathbf{S} , never changes.

- ▶ Build table, then do lookups. Like a real dictionary!
- ▶ Care more about time to do lookup than time to build dictionary

Obvious approaches:

- ▶ Sorted array: lookups $O(\log N)$
- ▶ Balanced search tree: $O(\log N)$

Can we do better with hashing? Yes, through universal hashing!

Method 1

Use table of size $M = N^2$.

Method 1

Use table of size $M = N^2$.

Theorem

Let H be universal with $M = N^2$. Then $\Pr_{h \sim H}[\text{no collisions in } S] \geq 1/2$.

Proof.

Fix $x, y \in S$ with $x \neq y$.

Method 1

Use table of size $M = N^2$.

Theorem

Let H be universal with $M = N^2$. Then $\Pr_{h \sim H}[\text{no collisions in } S] \geq 1/2$.

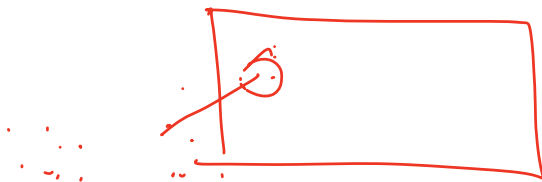
Proof.

Fix $x, y \in S$ with $x \neq y$.

$\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M = 1/N^2$ by universality.

Method 1

Use table of size $M = N^2$.



Theorem

Let H be universal with $M = N^2$. Then $\Pr_{h \sim H}[\text{no collisions in } S] \geq 1/2$.

Proof.

Fix $x, y \in S$ with $x \neq y$.

$\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M = 1/N^2$ by universality.

↖ which bound!

$$\begin{aligned} \Pr_{h \sim H}[\exists \text{ collision in } S] &\leq \sum_{\substack{x, y \in S \\ x \neq y}} \Pr_{h \sim H}[h(x) = h(y)] \leq \sum_{\substack{x, y \in S \\ x \neq y}} \frac{1}{N^2} \\ &= \binom{N}{2} \frac{1}{N^2} = \frac{N(N-1)}{2} \frac{1}{N^2} \leq \frac{1}{2} \end{aligned}$$



Method 1

Use table of size $M = N^2$.

Theorem

Let H be universal with $M = N^2$. Then $\Pr_{h \sim H}[\text{no collisions in } S] \geq 1/2$.

Proof.

Fix $x, y \in S$ with $x \neq y$.

$\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M = 1/N^2$ by universality.

$$\begin{aligned}\Pr_{h \sim H}[\exists \text{ collision in } S] &\leq \sum_{\substack{x, y \in S \\ x \neq y}} \Pr_{h \sim H}[h(x) = h(y)] \leq \sum_{\substack{x, y \in S \\ x \neq y}} \frac{1}{N^2} \\ &= \binom{N}{2} \frac{1}{N^2} = \frac{N(N-1)}{2} \frac{1}{N^2} \leq \frac{1}{2}\end{aligned}$$

□

So keep sampling $h \sim H$ until get one with no collisions!

Method 2

$M = N^2$ is pretty big!

- ▶ Only storing N things, and know them ahead of time
- ▶ Want space $O(N)$
- ▶ Open question for a long time!

Method 2

$M = N^2$ is pretty big!

- ▶ Only storing N things, and know them ahead of time
- ▶ Want space $O(N)$
- ▶ Open question for a long time!

Starting approach: set $M = N$, use a universal hash family H . Draw $h \sim H$.

- ▶ Will have collisions. Need to do something other than chaining.

Method 2

$M = N^2$ is pretty big!

- ▶ Only storing N things, and know them ahead of time
- ▶ Want space $O(N)$
- ▶ Open question for a long time!

Starting approach: set $M = N$, use a universal hash family H . Draw $h \sim H$.

- ▶ Will have collisions. Need to do something other than chaining.

For each $i \in [M]$, let $S_i = \{x \in S : h(x) = i\}$ and let $n_i = |S_i|$

Method 2

$M = N^2$ is pretty big!

- ▶ Only storing N things, and know them ahead of time
- ▶ Want space $O(N)$
- ▶ Open question for a long time!

Starting approach: set $M = N$, use a universal hash family H . Draw $h \sim H$.

- ▶ Will have collisions. Need to do something other than chaining.

For each $i \in [M]$, let $S_i = \{x \in S : h(x) = i\}$ and let $n_i = |S_i|$

- ▶ Use another hash table for S_i !
- ▶ Use Method 1: $O(n_i^2)$ -size ~~perfect~~ hashing of S_i .
 - ▶ Let $h_i : U \rightarrow [n_i^2]$ be hash function for S_i , and A_i be table (pointer from $A[i]$)

Method 2

$M = N^2$ is pretty big!

- ▶ Only storing N things, and know them ahead of time
- ▶ Want space $O(N)$
- ▶ Open question for a long time!

Starting approach: set $M = N$, use a universal hash family H . Draw $h \sim H$.

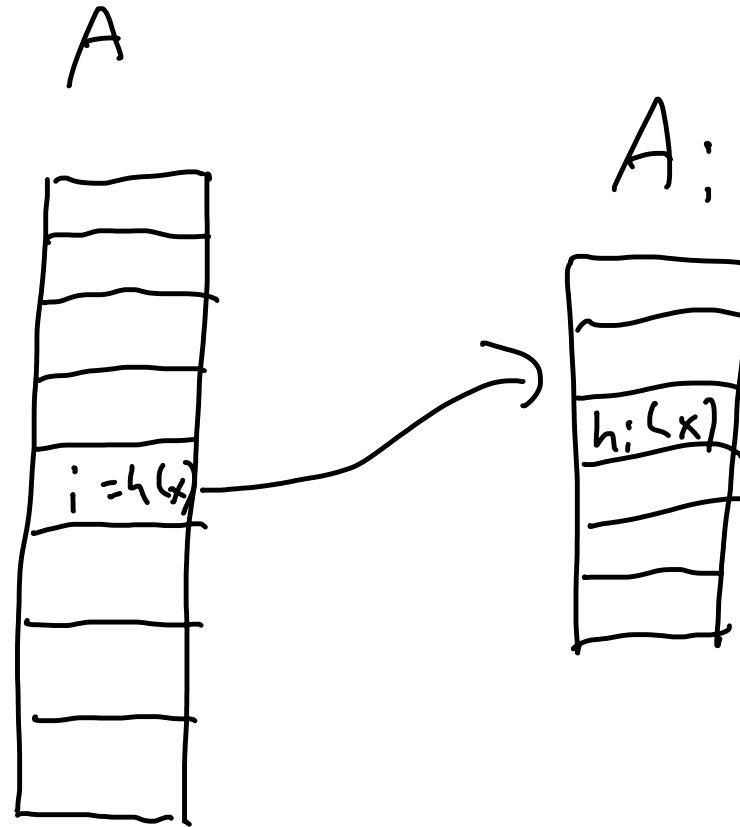
- ▶ Will have collisions. Need to do something other than chaining.

For each $i \in [M]$, let $S_i = \{x \in S : h(x) = i\}$ and let $n_i = |S_i|$

- ▶ Use another hash table for S_i !
- ▶ Use Method 1: $O(n_i^2)$ -size perfect hashing of S_i .
 - ▶ Let $h_i : U \rightarrow [n_i^2]$ be hash function for S_i , and A_i be table (pointer from $A[i]$)

Lookup(x): Look in $A_{h(x)}[h_{h(x)}(x)]$

Picture



Analysis

Lookup time: by analysis of Method 1, no collisions in second level.

⇒ Lookup time **$O(1)$**

Analysis

Lookup time: by analysis of Method 1, no collisions in second level.

⇒ Lookup time $O(1)$

Size: $O(N + \sum_{i=1}^N n_i^2)$

Analysis

Lookup time: by analysis of Method 1, no collisions in second level.

⇒ Lookup time $O(1)$

Size: $O(N + \sum_{i=1}^N n_i^2)$

Theorem

Let H be universal onto a table of size N . Then

$$\Pr_{h \sim H} \left[\sum_{i=1}^N n_i^2 > 4N \right] < 1/2.$$

So like with method 1: keep drawing $h \sim H$ until $\sum_{i=1}^N n_i^2 \leq 4N$

Analysis

Lookup time: by analysis of Method 1, no collisions in second level.

⇒ Lookup time $O(1)$

Size: $O(N + \sum_{i=1}^N n_i^2)$

Theorem

Let H be universal onto a table of size N . Then

$$\Pr_{h \sim H} \left[\sum_{i=1}^N n_i^2 > 4N \right] < 1/2.$$

So like with method 1: keep drawing $h \sim H$ until $\sum_{i=1}^N n_i^2 \leq 4N$

Prove that $E \left[\sum_{i=1}^N n_i^2 \right] \leq 2N$.

- ▶ Implies theorem by Markov's inequality
 - ▶ $\Pr[X > 2E[X]] \leq 1/2$ for nonnegative random variables X .

Proof

Observation: $\sum_{i=1}^N n_i^2$ is exactly number of *ordered* pairs that collide, including self-collisions

- ▶ Example: If $S_i = \{a, b, c\}$ then $n_i^2 = 9$. Ordered colliding pairs:
 $(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)$

Proof

Observation: $\sum_{i=1}^N n_i^2$ is exactly number of *ordered* pairs that collide, including self-collisions

- ▶ Example: If $S_i = \{a, b, c\}$ then $n_i^2 = 9$. Ordered colliding pairs:
 $(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)$

Let $C_{xy} = \begin{cases} 1 & \text{if } h(x) = h(y) \\ 0 & \text{otherwise} \end{cases}$

Proof

Observation: $\sum_{i=1}^N n_i^2$ is exactly number of *ordered* pairs that collide, including self-collisions

- ▶ Example: If $S_i = \{a, b, c\}$ then $n_i^2 = 9$. Ordered colliding pairs:
 $(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)$

$$\text{Let } C_{xy} = \begin{cases} 1 & \text{if } h(x) = h(y) \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} E \left[\sum_{i=1}^N n_i^2 \right] &= E \left[\sum_{x \in S} \sum_{y \in S} C_{xy} \right] \\ &= N + \sum_{x \in S} \sum_{y \in S: y \neq x} E[C_{xy}] && \text{(linearity of expectations)} \\ &\leq N + \frac{N(N-1)}{M} && \text{(definition of universal)} \\ &< 2N && \text{(since } M = N) \end{aligned}$$