Lecture 10: Universal and Perfect Hashing

Michael Dinitz

September 30, 2021
601.433/633 Introduction to Algorithms

Introduction

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

- Can improve operations to $\mathbf{O(1)}$, 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

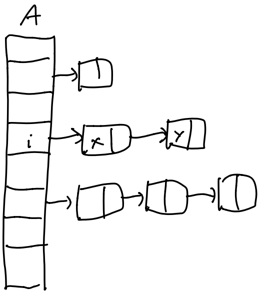
- Keys from universe $\mathbf{U}$ (think very large)
- Set $\mathbf{S} \subseteq \mathbf{U}$ of keys we actually care about (think relatively small). $|\mathbf{S}| = \mathbf{N}$.
- Hash table $\mathbf{A}$ (array) of size $\mathbf{M}$.
- Hash function $\mathbf{h : U \to [M]}$
  - $\mathbf{[M] = \{1, 2,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

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 $\mathbf{h} : \mathbf{U} \to [\mathbf{M}]$
  - $[\mathbf{M}] = \{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{M}\}$
- 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($\mathbf{x}$): Walk down the list at $\mathbf{A}[\mathbf{h}(\mathbf{x})]$ until we find $\mathbf{x}$ (or walk to the end of the list)

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

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

Dictionary Operations

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

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

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

Question: What should hash function be?

Dictionary Operations

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

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

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

Question: What should hash function be?

Properties we want:

Dictionary Operations

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

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

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

Question: What should hash function be?

Properties we want:

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

Dictionary Operations

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

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

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

Question: What should hash function be?

Properties we want:

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

Dictionary Operations

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

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

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

Question: What should hash function be?

Properties we want:

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

Bad News

Theorem

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

Bad News

Theorem

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

Proof.

Pigeonhole principle / contradiction / contrapositive. □

Bad News

Theorem

For any hash function $\mathbf{h}$, *if* $|\mathbf{U}| \geq (\mathbf{N}-\mathbf{1})\mathbf{M}+\mathbf{1}$, *then there exists a set* $\mathbf{S}$ *of* $\mathbf{N}$ *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 $\mathbf{h}$*, if* $|\mathbf{U}| \geq (\mathbf{N}-1)\mathbf{M}+1$*, then there exists a set* $\mathbf{S}$ *of* $\mathbf{N}$ *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 adversary isn't looking at your $\mathbf{h}$ when deciding on elements.

Bad News

Theorem

For any hash function $\mathbf{h}$, *if* $|\mathbf{U}| \geq (\mathbf{N}-1)\mathbf{M}+1$, *then there exists a set* $\mathbf{S}$ *of* $\mathbf{N}$ *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 adversary isn't looking at your $\mathbf{h}$ when deciding on elements.
- Option 2: Randomness! *Random function* $\mathbf{h}:\mathbf{U}\to[\mathbf{M}]$
  - For each $\mathbf{x}\in\mathbf{U}$, choose $\mathbf{y}\in[\mathbf{M}]$ uniformly at random and set $\mathbf{h}(\mathbf{x})=\mathbf{y}$.
  - Hopefully good behavior in expectation.

Bad News

Theorem

For any hash function $\mathbf{h}$, *if* $|\mathbf{U}| \geq (\mathbf{N}-1)\mathbf{M}+1$, *then there exists a set* $\mathbf{S}$ *of* $\mathbf{N}$ *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 adversary isn't looking at your $\mathbf{h}$ when deciding on elements.
- Option 2: Randomness! *Random function* $\mathbf{h} : \mathbf{U} \to [\mathbf{M}]$
  - For each $\mathbf{x} \in \mathbf{U}$, choose $\mathbf{y} \in [\mathbf{M}]$ uniformly at random and set $\mathbf{h}(\mathbf{x}) = \mathbf{y}$.
  - Hopefully good behavior in expectation.
  - Problem: How can we store/remember/create $\mathbf{h}$?

Universal Hashing

Definition

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

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

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 $\{\mathbf{h} : \mathbf{U} \to [\mathbf{M}]\}$ is *universal* if

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

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 $\{\mathbf{h} : \mathbf{U} \to [\mathbf{M}]\}$ 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 $\{\mathbf{h} : \mathbf{U} \to [\mathbf{M}]\}$ is *universal* if

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

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 $\mathbf{N}/\mathbf{M}$.

So Lookup($\mathbf{x}$) and Delete($\mathbf{x}$) have expected time $\mathbf{O}(\mathbf{N}/\mathbf{M})$.
$\implies$ If $\mathbf{M} = \mathbf{\Omega}(\mathbf{N})$, operations in $\mathbf{O}(\mathbf{1})$ 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.

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

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

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* $\mathbf{N/M}$*.*

Proof.

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

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

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

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

□

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 $\mathbf{O(M)}$ elements in the system at any time, the expected total cost of the whole sequence is only $\mathbf{O(L)}$ (assuming $\mathbf{h}$ takes constant time to compute).

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* $\mathbf{O(M)}$ *elements in the system at any time, the expected total cost of the whole sequence is only* $\mathbf{O(L)}$ *(assuming* $\mathbf{h}$ *takes constant time to compute).*

Proof.

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

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 $\mathbf{O(M)}$ elements in the system at any time, the expected total cost of the whole sequence is only $\mathbf{O(L)}$ (assuming $\mathbf{h}$ takes constant time to compute).

Proof.

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

So universal distributions are great. Can we construct them?

Universal Hash Families

Definition

If $\mathbf{H}$ is universal and is a uniform distribution over a set of functions $\{\mathbf{h_1}, \mathbf{h_2}, \ldots\}$, 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.

Universal Hash Families

Definition

If $\mathbf{H}$ is universal and is a uniform distribution over a set of functions $\{\mathbf{h_1}, \mathbf{h_2}, \ldots\}$, 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:

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

Universal Hash Families

Definition

If $\mathbf{H}$ is universal and is a uniform distribution over a set of functions $\{\mathbf{h_1}, \mathbf{h_2}, \ldots\}$, 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:

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

$$\begin{array}{ccccc} h & & x & & h(x) \\ \boxed{\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{matrix}} & & \boxed{\begin{matrix} 1 \\ 0 \\ 1 \\ 0 \end{matrix}} & = & \boxed{\begin{matrix} 1 \\ 1 \\ 0 \end{matrix}} \end{array}$$

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

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

Universality

Theorem

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

Universality

Theorem

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

Proof.

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

Universality

Theorem

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

Proof.

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

Since $\mathbf{x} \neq \mathbf{y}$, there is $\mathbf{i}$ s.t. $\mathbf{x}_\mathbf{i} \neq \mathbf{y}_\mathbf{i}$. WLOG, $\mathbf{x}_\mathbf{i} = \mathbf{0}$ and $\mathbf{y}_\mathbf{i} = \mathbf{1}$.

Universality

Theorem

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

Proof.

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

Since $\mathbf{x} \neq \mathbf{y}$, there is $\mathbf{i}$ s.t. $\mathbf{x}_\mathbf{i} \neq \mathbf{y}_\mathbf{i}$. WLOG, $\mathbf{x}_\mathbf{i} = \mathbf{0}$ and $\mathbf{y}_\mathbf{i} = \mathbf{1}$.

Draw all entries of $\mathbf{h}$ *except* for $\mathbf{h}^\mathbf{i}$. Let $\mathbf{h}' = \mathbf{h}$ with $\mathbf{h}^\mathbf{i}$ all $\mathbf{0}$'s

- $\mathbf{h}(\mathbf{x}) = \mathbf{h}'(\mathbf{x})$ already fixed.

Universality

Theorem

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

Proof.

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

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

Draw all entries of $\mathbf{h}$ *except* for $\mathbf{h^i}$. Let $\mathbf{h}' = \mathbf{h}$ with $\mathbf{h^i}$ all $\mathbf{0}$'s

- $\mathbf{h}(\mathbf{x}) = \mathbf{h}'(\mathbf{x})$ already fixed.
- If $\mathbf{h}(\mathbf{y}) = \mathbf{h}(\mathbf{x})$, then $\mathbf{h^i}$ must equal $\mathbf{h}(\mathbf{x}) - \mathbf{h}'(\mathbf{y})$

Universality

Theorem

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

Proof.

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

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

Draw all entries of $\mathbf{h}$ *except* for $\mathbf{h^i}$. Let $\mathbf{h}' = \mathbf{h}$ with $\mathbf{h^i}$ all $\mathbf{0}$'s

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

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 $\mathbf{O(\log N)}$
- Balanced search tree: $\mathbf{O(\log N)}$

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 $\mathbf{O(\log N)}$
- Balanced search tree: $\mathbf{O(\log N)}$

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 $\mathbf{O(\log N)}$
- Balanced search tree: $\mathbf{O(\log N)}$

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

Method 1

Use table of size $\mathbf{M} = \mathbf{N}^2$.

Method 1

Use table of size $\mathbf{M} = \mathbf{N}^2$.

Theorem

Let $\mathbf{H}$ *be universal with* $\mathbf{M} = \mathbf{N}^2$*. Then* $\mathbf{Pr}_{\mathbf{h}\sim\mathbf{H}}[\textit{no collisions in } \mathbf{S}] \geq \mathbf{1/2}$*.*

Proof.

Fix $\mathbf{x}, \mathbf{y} \in \mathbf{S}$ with $\mathbf{x} \neq \mathbf{y}$.

Method 1

Use table of size $\mathbf{M} = \mathbf{N}^2$.

Theorem

Let $\mathbf{H}$ *be universal with* $\mathbf{M} = \mathbf{N}^2$. *Then* $\mathbf{Pr}_{\mathbf{h} \sim \mathbf{H}}[\textit{no collisions in } \mathbf{S}] \geq \mathbf{1/2}$.

Proof.

Fix $\mathbf{x}, \mathbf{y} \in \mathbf{S}$ with $\mathbf{x} \neq \mathbf{y}$.

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

Method 1

Use table of size $\mathbf{M = N^2}$.

Theorem

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

Proof.

Fix $\mathbf{x, y \in S}$ with $\mathbf{x \neq y}$.

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

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

$$= \binom{N}{2}\frac{1}{N^2} = \frac{N(N-1)}{2}\frac{1}{N^2} \leq \frac{1}{2}$$

□

Method 1

Use table of size $\mathbf{M} = \mathbf{N}^2$.

Theorem

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

Proof.

Fix $\mathbf{x}, \mathbf{y} \in \mathbf{S}$ with $\mathbf{x} \neq \mathbf{y}$.
$\mathbf{Pr}_{\mathbf{h}\sim\mathbf{H}}[\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] \leq \mathbf{1/M} = \mathbf{1/N^2}$ 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 $\mathbf{h} \sim \mathbf{H}$ until get one with no collisions!

Method 2

$\mathbf{M} = \mathbf{N}^2$ is pretty big!

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

Method 2

$\mathbf{M} = \mathbf{N^2}$ is pretty big!

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

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

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

Method 2

$\mathbf{M} = \mathbf{N}^2$ is pretty big!

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

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

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

Let $\mathbf{S_i} = \{\mathbf{x} \in \mathbf{S} : \mathbf{h(x)} = \mathbf{i}\}$ and let $\mathbf{n_i} = |\mathbf{S_i}|$

Method 2

$\mathbf{M} = \mathbf{N}^2$ is pretty big!

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

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

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

Let $\mathbf{S_i} = \{\mathbf{x} \in \mathbf{S} : \mathbf{h(x)} = \mathbf{i}\}$ and let $\mathbf{n_i} = |\mathbf{S_i}|$

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

Method 2

$\mathbf{M} = \mathbf{N}^2$ is pretty big!

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

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

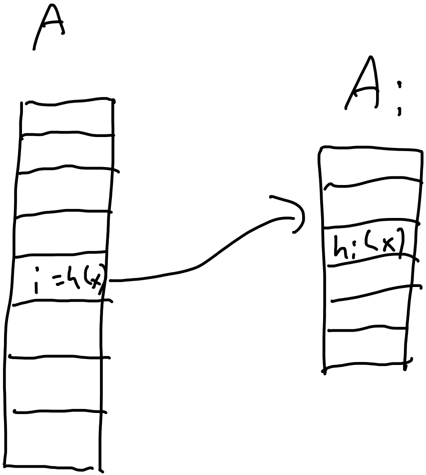
- Will have collisions. Need to do something other than chaining.

Let $\mathbf{S_i} = \{\mathbf{x} \in \mathbf{S} : \mathbf{h(x)} = \mathbf{i}\}$ and let $\mathbf{n_i} = |\mathbf{S_i}|$

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

Lookup($\mathbf{x}$): Look in linked list at $\mathbf{A_{h(x)}[h_{h(x)}(x)]}$

Picture



Analysis

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

$\Longrightarrow$ Lookup time $\mathbf{O(1)}$

Analysis

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

$\implies$ Lookup time $\mathbf{O(1)}$

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

Analysis

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

$\Longrightarrow$ Lookup time $\mathbf{O(1)}$

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

Theorem

Let $\mathbf{H}$ *be universal onto a table of size* $\mathbf{N}$*. Then*

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

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

Analysis

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

$\Longrightarrow$ 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: $(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)$

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

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