# CS 473: Algorithms

Chandra Chekuri     Ruta Mehta

University of Illinois, Urbana-Champaign

Fall 2016

# Universal Hashing

Lecture 10

September 23, 2016

# Part I

# Hash Tables

# Dictionary Data Structure

1. $\mathcal{U}$: universe of keys with total order: numbers, strings, etc.
2. Data structure to store a subset $\mathbf{S} \subseteq \mathcal{U}$
3. **Operations:**
    1. **Search/look up**: given $\mathbf{x} \in \mathcal{U}$ is $\mathbf{x} \in \mathbf{S}$?
    2. **Insert**: given $\mathbf{x} \not\in \mathbf{S}$ add $\mathbf{x}$ to $\mathbf{S}$.
    3. **Delete**: given $\mathbf{x} \in \mathbf{S}$ delete $\mathbf{x}$ from $\mathbf{S}$
4. **Static** structure: $\mathbf{S}$ given in advance or changes very infrequently, main operations are lookups.
5. **Dynamic** structure: $\mathbf{S}$ changes rapidly so inserts and deletes as important as lookups.

Can we do everything in $\mathbf{O(1)}$ time?

# Hashing and Hash Tables

Hash Table data structure:

1. A (hash) table/array $\mathbf{T}$ of size $\mathbf{m}$ (the table **size**).
2. A hash function $\mathbf{h} : \mathcal{U} \rightarrow \{\mathbf{0}, \ldots, \mathbf{m} - \mathbf{1}\}$.
3. Item $\mathbf{x} \in \mathcal{U}$ hashes to slot $\mathbf{h}(\mathbf{x})$ in $\mathbf{T}$.

# Hashing and Hash Tables

Hash Table data structure:

1. A (hash) table/array $\mathbf{T}$ of size $\mathbf{m}$ (the table **size**).
2. A hash function $\mathbf{h} : \mathcal{U} \rightarrow \{\mathbf{0}, \ldots, \mathbf{m} - \mathbf{1}\}$.
3. Item $\mathbf{x} \in \mathcal{U}$ hashes to slot $\mathbf{h}(\mathbf{x})$ in $\mathbf{T}$.

Given $\mathbf{S} \subseteq \mathcal{U}$. How do we store $\mathbf{S}$ and how do we do lookups?

# Hashing and Hash Tables

Hash Table data structure:

1. A (hash) table/array $\mathbf{T}$ of size $\mathbf{m}$ (the table **size**).
2. A hash function $\mathbf{h} : \mathcal{U} \rightarrow \{\mathbf{0}, \ldots, \mathbf{m} - \mathbf{1}\}$.
3. Item $\mathbf{x} \in \mathcal{U}$ hashes to slot $\mathbf{h}(\mathbf{x})$ in $\mathbf{T}$.

Given $\mathbf{S} \subseteq \mathcal{U}$. How do we store $\mathbf{S}$ and how do we do lookups?

*Ideal situation:*

1. Each element $\mathbf{x} \in \mathbf{S}$ hashes to a distinct slot in $\mathbf{T}$. Store $\mathbf{x}$ in slot $\mathbf{h}(\mathbf{x})$
2. **Lookup**: Given $\mathbf{y} \in \mathcal{U}$ check if $\mathbf{T}[\mathbf{h}(\mathbf{y})] = \mathbf{y}$. $\mathbf{O}(\mathbf{1})$ time!

# Hashing and Hash Tables

Hash Table data structure:

1. A (hash) table/array $\mathbf{T}$ of size $\mathbf{m}$ (the table **size**).
2. A hash function $\mathbf{h} : \mathcal{U} \rightarrow \{0, \ldots, \mathbf{m} - 1\}$.
3. Item $\mathbf{x} \in \mathcal{U}$ hashes to slot $\mathbf{h(x)}$ in $\mathbf{T}$.

Given $\mathbf{S} \subseteq \mathcal{U}$. How do we store $\mathbf{S}$ and how do we do lookups?

*Ideal situation:*

1. Each element $\mathbf{x} \in \mathbf{S}$ hashes to a distinct slot in $\mathbf{T}$. Store $\mathbf{x}$ in slot $\mathbf{h(x)}$
2. **Lookup**: Given $\mathbf{y} \in \mathcal{U}$ check if $\mathbf{T[h(y)] = y}$. $\mathbf{O(1)}$ time!

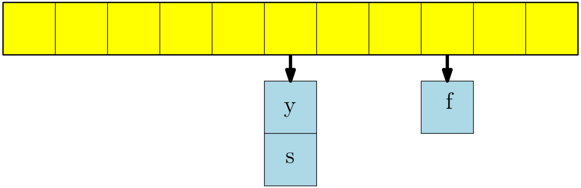
Collisions unavoidable if $|\mathbf{T}| < |\mathcal{U}|$. Several techniques to handle them.

# Handling Collisions: Chaining

**Collision:** $\mathbf{h(x) = h(y)}$ for some $\mathbf{x \neq y}$.

**Chaining/Open hashing** to handle collisions:

1. For each slot $\mathbf{i}$ store all items hashed to slot $\mathbf{i}$ in a linked list. $\mathbf{T[i]}$ points to the linked list
2. **Lookup**: to find if $\mathbf{y} \in \mathcal{U}$ is in $\mathbf{T}$, check the linked list at $\mathbf{T[h(y)]}$. Time proportion to size of linked list.




Does hashing give $\mathbf{O(1)}$ time per operation for dictionaries?

# Hash Functions

Parameters: $\mathbf{N} = |\mathcal{U}|$ (very large), $\mathbf{m} = |\mathbf{T}|$, $\mathbf{n} = |\mathbf{S}|$
Goal: $\mathbf{O(1)}$-time lookup, insertion, deletion.

## Single hash function

If $\mathbf{N} \geq \mathbf{m}^2$, then for any hash function $\mathbf{h} : \mathcal{U} \rightarrow \mathbf{T}$ there exists $\mathbf{i} < \mathbf{m}$ such that at least $\mathbf{N}/\mathbf{m} \geq \mathbf{m}$ elements of $\mathcal{U}$ get hashed to slot $\mathbf{i}$.

# Hash Functions

Parameters: $\mathbf{N} = |\mathcal{U}|$ (very large), $\mathbf{m} = |\mathbf{T}|$, $\mathbf{n} = |\mathbf{S}|$
Goal: $\mathbf{O(1)}$-time lookup, insertion, deletion.

## Single hash function

If $\mathbf{N} \geq \mathbf{m}^2$, then for any hash function $\mathbf{h} : \mathcal{U} \rightarrow \mathbf{T}$ there exists $\mathbf{i} < \mathbf{m}$ such that at least $\mathbf{N}/\mathbf{m} \geq \mathbf{m}$ elements of $\mathcal{U}$ get hashed to slot $\mathbf{i}$. Any $\mathbf{S}$ containing all of these is a **very very bad set for h**!

# Hash Functions

Parameters: $\mathbf{N} = |\mathcal{U}|$ (very large), $\mathbf{m} = |\mathbf{T}|$, $\mathbf{n} = |\mathbf{S}|$
Goal: $\mathbf{O(1)}$-time lookup, insertion, deletion.

## Single hash function

If $\mathbf{N} \geq \mathbf{m^2}$, then for any hash function $\mathbf{h} : \mathcal{U} \rightarrow \mathbf{T}$ there exists $\mathbf{i} < \mathbf{m}$ such that at least $\mathbf{N/m} \geq \mathbf{m}$ elements of $\mathcal{U}$ get hashed to slot $\mathbf{i}$. Any $\mathbf{S}$ containing all of these is a **very very bad set for h**!

Such a bad set may lead to $\mathbf{O(m)}$ lookup time!

# Hash Functions

Parameters: $\mathbf{N} = |\mathcal{U}|$ (very large), $\mathbf{m} = |\mathbf{T}|$, $\mathbf{n} = |\mathbf{S}|$
Goal: $\mathbf{O(1)}$-time lookup, insertion, deletion.

## Single hash function

If $\mathbf{N} \geq \mathbf{m}^2$, then for any hash function $\mathbf{h} : \mathcal{U} \rightarrow \mathbf{T}$ there exists $\mathbf{i} < \mathbf{m}$ such that at least $\mathbf{N}/\mathbf{m} \geq \mathbf{m}$ elements of $\mathcal{U}$ get hashed to slot $\mathbf{i}$. Any $\mathbf{S}$ containing all of these is a **very very bad set for h**!

Such a bad set may lead to $\mathbf{O(m)}$ lookup time!

## Lesson:

- Consider a family $\mathcal{H}$ of hash functions with *good properties* and choose $\mathbf{h}$ uniformly at random.
- Guarantees: small # collisions in expectation for a given $\mathbf{S}$.
- $\mathcal{H}$ should allow efficient sampling.

# Universal Hashing

**Question:** What are good properties of $\mathcal{H}$ in distributing data?

# Universal Hashing

**Question:** What are good properties of $\mathcal{H}$ in distributing data?

1. **Uniform:** Consider any element $\mathbf{x} \in \mathcal{U}$. Then if $\mathbf{h} \in \mathcal{H}$ is picked randomly then $\mathbf{x}$ should go into a random slot in $\mathbf{T}$. In other words $\Pr[\mathbf{h(x) = i}] = \mathbf{1/m}$ for every $\mathbf{0 \leq i < m}$.

# Universal Hashing

**Question:** What are good properties of $\mathcal{H}$ in distributing data?

1. **Uniform:** Consider any element $\mathbf{x} \in \mathcal{U}$. Then if $\mathbf{h} \in \mathcal{H}$ is picked randomly then $\mathbf{x}$ should go into a random slot in $\mathbf{T}$. In other words $\Pr[\mathbf{h}(\mathbf{x}) = \mathbf{i}] = \mathbf{1}/\mathbf{m}$ for every $\mathbf{0} \leq \mathbf{i} < \mathbf{m}$.
2. **Universal:** Consider any two distinct elements $\mathbf{x}, \mathbf{y} \in \mathcal{U}$. Then if $\mathbf{h} \in \mathcal{H}$ is picked randomly then the probability of a collision between $\mathbf{x}$ and $\mathbf{y}$ should be at most $\mathbf{1}/\mathbf{m}$. In other words $\Pr[\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] = \mathbf{1}/\mathbf{m}$ (cannot be smaller).

# Universal Hashing

**Question:** What are good properties of $\mathcal{H}$ in distributing data?

1. **Uniform:** Consider any element $\mathbf{x} \in \mathcal{U}$. Then if $\mathbf{h} \in \mathcal{H}$ is picked randomly then $\mathbf{x}$ should go into a random slot in $\mathbf{T}$. In other words $\Pr[\mathbf{h}(\mathbf{x}) = \mathbf{i}] = \mathbf{1}/\mathbf{m}$ for every $\mathbf{0} \leq \mathbf{i} < \mathbf{m}$.
2. **Universal:** Consider any two distinct elements $\mathbf{x}, \mathbf{y} \in \mathcal{U}$. Then if $\mathbf{h} \in \mathcal{H}$ is picked randomly then the probability of a collision between $\mathbf{x}$ and $\mathbf{y}$ should be at most $\mathbf{1}/\mathbf{m}$. In other words $\Pr[\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] = \mathbf{1}/\mathbf{m}$ (cannot be smaller).
3. Second property is stronger than the first and the crucial issue.

## Definition

A family of hash function $\mathcal{H}$ is (2-)**universal** if for all distinct $\mathbf{x}, \mathbf{y} \in \mathcal{U}$, $\Pr_{\mathbf{h} \sim \mathcal{H}}[\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] = \mathbf{1}/\mathbf{m}$ where $\mathbf{m}$ is the table size.

# Analyzing Universal Hashing

**Question:** Fixing set $\mathbf{S}$, what is the *expected* time to look up $\mathbf{x} \in \mathbf{S}$ when $\mathbf{h}$ is picked uniformly at random from $\mathcal{H}$?

1. $\ell(\mathbf{x})$ **:** the size of the list at $\mathbf{T[h(x)]}$. We want $\mathrm{E}[\ell(\mathbf{x})]$
2. For $\mathbf{y} \in \mathbf{S}$ let $\mathbf{D_y}$ be one if $\mathbf{h(y) = h(x)}$, else zero.
   $\ell(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbf{S}} \mathbf{D_y}$

# Analyzing Universal Hashing

**Question:** Fixing set $\mathbf{S}$, what is the *expected* time to look up $\mathbf{x} \in \mathbf{S}$ when $\mathbf{h}$ is picked uniformly at random from $\mathcal{H}$?

1. $\ell(\mathbf{x})$ **:** the size of the list at $\mathbf{T[h(x)]}$. We want $\mathrm{E}[\ell(\mathbf{x})]$
2. For $\mathbf{y} \in \mathbf{S}$ let $\mathbf{D_y}$ be one if $\mathbf{h(y) = h(x)}$, else zero. $\ell(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbf{S}} \mathbf{D_y}$

$$
\begin{aligned}
\mathbf{E}[\ell(\mathbf{x})] &= \sum_{\mathbf{y}\in\mathbf{S}} \mathrm{E}[\mathbf{D_y}] = \sum_{\mathbf{y}\in\mathbf{S}} \mathbf{Pr[h(x) = h(y)]} \\
&= \sum_{\mathbf{y}\in\mathbf{S}} \frac{1}{\mathbf{m}} \qquad \text{(since } \mathcal{H} \text{ is a universal hash family)} \\
&= |\mathbf{S}|/\mathbf{m} \leq \mathbf{1} \quad \text{if } |\mathbf{S}| \leq \mathbf{m}
\end{aligned}
$$

# Analyzing Universal Hashing

**Question:** What is the *expected* time to look up $\mathbf{x}$ in $\mathbf{T}$ using $\mathbf{h}$ assuming chaining used to resolve collisions?

**Answer:** $\mathbf{O(n/m)}$.

# Analyzing Universal Hashing

**Question:** What is the *expected* time to look up $\mathbf{x}$ in $\mathbf{T}$ using $\mathbf{h}$ assuming chaining used to resolve collisions?

**Answer:** $\mathbf{O(n/m)}$.

Comments:

1. $\mathbf{O(1)}$ expected time also holds for insertion.
2. Analysis assumes static set $\mathbf{S}$ but holds as long as $\mathbf{S}$ is a set formed with at most $\mathbf{O(m)}$ insertions and deletions.
3. **Worst-case**: look up time can be large! How large? $\mathbf{\Omega(\log n / \log\log n)}$

# Universal Hash Family

Universal: $\mathcal{H}$ such that $\mathbf{Pr[h(x) = h(y)] = 1/m}$.

## All functions

$\mathcal{H}$ **:** Set of all possible functions $\mathbf{h} : \mathcal{U} \to \{\mathbf{0}, \ldots, \mathbf{m-1}\}$.

- Universal.

# Universal Hash Family

Universal: $\mathcal{H}$ such that $\mathbf{Pr[h(x) = h(y)] = 1/m}$.

## All functions

$\mathcal{H}$ **:** Set of all possible functions $\mathbf{h : \mathcal{U} \rightarrow \{0, \ldots, m-1\}}$.

- Universal.
- $|\mathcal{H}| = \mathbf{m}^{|\mathcal{U}|}$
- representing $\mathbf{h}$ requires $|\mathcal{U}| \log \mathbf{m}$ – Not $\mathbf{O(1)}$!

# Universal Hash Family

Universal: $\mathcal{H}$ such that $\mathbf{Pr[h(x) = h(y)] = 1/m}$.

## All functions

$\mathcal{H}$ **:** Set of all possible functions $\mathbf{h : \mathcal{U} \to \{0, \ldots, m-1\}}$.

- Universal.
- $|\mathcal{H}| = \mathbf{m}^{|\mathcal{U}|}$
- representing $\mathbf{h}$ requires $|\mathcal{U}| \log \mathbf{m}$ – Not $\mathbf{O(1)}$!

We need *compactly representable* universal family.

# Compact Universal Hash Family

Parameters: $\mathbf{N} = |\mathcal{U}|$, $\mathbf{m} = |\mathbf{T}|$, $\mathbf{n} = |\mathbf{S}|$

1. Choose a **prime** number $\mathbf{p} \geq \mathbf{N}$. $\mathbb{Z}_\mathbf{p} = \{0, 1, \ldots, \mathbf{p} - 1\}$ is a field.
2. For $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_\mathbf{p}$, $\mathbf{a} \neq 0$, define the hash function $\mathbf{h}_{\mathbf{a},\mathbf{b}}$ as $\mathbf{h}_{\mathbf{a},\mathbf{b}}(\mathbf{x}) = ((\mathbf{ax} + \mathbf{b}) \mod \mathbf{p}) \mod \mathbf{m}$.
3. Let $\mathcal{H} = \{\mathbf{h}_{\mathbf{a},\mathbf{b}} \mid \mathbf{a}, \mathbf{b} \in \mathbb{Z}_\mathbf{p}, \mathbf{a} \neq 0\}$. Note that $|\mathcal{H}| = \mathbf{p}(\mathbf{p} - 1)$.

# Compact Universal Hash Family

Parameters: $\mathbf{N} = |\mathcal{U}|$, $\mathbf{m} = |\mathbf{T}|$, $\mathbf{n} = |\mathbf{S}|$

1. Choose a **prime** number $\mathbf{p} \geq \mathbf{N}$. $\mathbb{Z}_\mathbf{p} = \{0, 1, \ldots, \mathbf{p} - 1\}$ is a field.
2. For $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_\mathbf{p}$, $\mathbf{a} \neq 0$, define the hash function $\mathbf{h}_{\mathbf{a},\mathbf{b}}$ as $\mathbf{h}_{\mathbf{a},\mathbf{b}}(\mathbf{x}) = ((\mathbf{ax} + \mathbf{b}) \mod \mathbf{p}) \mod \mathbf{m}$.
3. Let $\mathcal{H} = \{\mathbf{h}_{\mathbf{a},\mathbf{b}} \mid \mathbf{a}, \mathbf{b} \in \mathbb{Z}_\mathbf{p}, \mathbf{a} \neq 0\}$. Note that $|\mathcal{H}| = \mathbf{p}(\mathbf{p} - 1)$.

**Theorem**

*$\mathcal{H}$ is a universal hash family.*

# Compact Universal Hash Family

Parameters: $\mathbf{N} = |\mathcal{U}|$, $\mathbf{m} = |\mathbf{T}|$, $\mathbf{n} = |\mathbf{S}|$

1. Choose a **prime** number $\mathbf{p} \geq \mathbf{N}$. $\mathbb{Z}_\mathbf{p} = \{0, 1, \ldots, \mathbf{p} - 1\}$ is a field.
2. For $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_\mathbf{p}$, $\mathbf{a} \neq 0$, define the hash function $\mathbf{h_{a,b}}$ as $\mathbf{h_{a,b}(x) = ((ax + b) \mod p) \mod m}$.
3. Let $\mathcal{H} = \{\mathbf{h_{a,b}} \mid \mathbf{a}, \mathbf{b} \in \mathbb{Z}_\mathbf{p}, \mathbf{a} \neq 0\}$. Note that $|\mathcal{H}| = \mathbf{p}(\mathbf{p} - 1)$.

**Theorem**

*$\mathcal{H}$ is a universal hash family.*

Comments:

1. Hash family is of small size, easy to sample from.
2. Easy to store a hash function ($\mathbf{a}, \mathbf{b}$ have to be stored) and evaluate it.

# Some math required...

## Lemma (LemmaUnique)

*Let* $\mathbf{p}$ *be a prime number,*
$\mathbf{x}$*: an integer number in* $\{1, \ldots, \mathbf{p}-1\}$*.*
$\implies$ *There exists a unique* $\mathbf{y}$ *s.t.* $\mathbf{xy} = 1 \mod \mathbf{p}$*.*

In other words: For every element there is a unique inverse.
$\implies$ $\mathbb{Z}_{\mathbf{p}} = \{0, 1, \ldots, \mathbf{p}-1\}$ when working modulo $\mathbf{p}$ is a *field*.

# Proof of LemmaUnique

## Claim

*Let* $\mathbf{p}$ *be a prime number. For any* $\mathbf{x, y, z} \in \{\mathbf{1, \ldots, p-1}\}$ *s.t.* $\mathbf{y \neq z}$*, we have that* $\mathbf{xy \mod p \neq xz \mod p}$*.*

## Proof.

Assume for the sake of contradiction $\mathbf{xy \mod p = xz \mod p}$. Then

$$\begin{aligned}
\mathbf{x(y-z)} &= \mathbf{0 \mod p} \\
\implies & \quad \mathbf{p} \text{ divides } \mathbf{x(y-z)} \\
\implies & \quad \mathbf{p} \text{ divides } \mathbf{y-z} \\
\implies & \quad \mathbf{y - z = 0} \\
\implies & \quad \mathbf{y = z.}
\end{aligned}$$

# Proof of LemmaUnique

## Lemma (LemmaUnique)

*Let* $\mathbf{p}$ *be a prime number,*
$\mathbf{x}$*: an integer number in* $\{1, \ldots, \mathbf{p}-1\}$*.*
$\implies$ *There exists a unique* $\mathbf{y}$ *s.t.* $\mathbf{xy} = 1 \mod \mathbf{p}$*.*

## Proof.

By the above claim if $\mathbf{xy} = 1 \mod \mathbf{p}$ and $\mathbf{xz} = 1 \mod \mathbf{p}$ then $\mathbf{y} = \mathbf{z}$. Hence uniqueness follows.

# Proof of LemmaUnique

## Lemma (LemmaUnique)

*Let* $\mathbf{p}$ *be a prime number,*
$\mathbf{x}$*: an integer number in* $\{1, \ldots, p-1\}$*.*
$\implies$ *There exists a unique* $\mathbf{y}$ *s.t.* $xy = 1 \mod p$*.*

## Proof.

By the above claim if $xy = 1 \mod p$ and $xz = 1 \mod p$ then $y = z$. Hence uniqueness follows.

**Existence.** For any $x \in \{1, \ldots, p-1\}$ we have that $\{x * 1 \mod p, x * 2 \mod p, \ldots, x * (p-1) \mod p\} = \{1, 2, \ldots, p-1\}$.
$\implies$ There exists a number $y \in \{1, \ldots, p-1\}$ such that $xy = 1 \mod p$.

# Proof of the Theorem: Outline

$\mathbf{h_{a,b}(x) = ((ax + b) \mod p) \mod m)}$.

## Theorem

$\mathcal{H} = \{\mathbf{h_{a,b}} \mid \mathbf{a, b} \in \mathbb{Z}_\mathbf{p}, \mathbf{a} \neq \mathbf{0}\}$ *is universal.*

## Proof.

Fix $\mathbf{x, y} \in \mathcal{U}$. We need to show that
$\Pr_{\mathbf{h_{a,b}} \sim \mathcal{H}}[\mathbf{h_{a,b}(x) = h_{a,b}(y)}] \leq \mathbf{1/m}$. Note that $|\mathcal{H}| = \mathbf{p(p-1)}$.

# Proof of the Theorem: Outline

$h_{a,b}(x) = ((ax + b) \mod p) \mod m)$.

**Theorem**

$\mathcal{H} = \{h_{a,b} \mid a, b \in \mathbb{Z}_p, a \neq 0\}$ *is universal.*

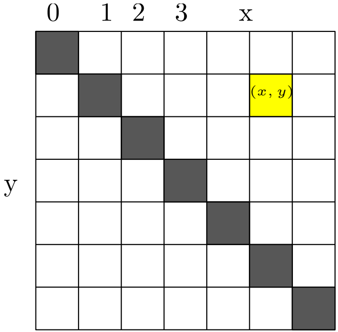
**Proof.**

Fix $x, y \in \mathcal{U}$. We need to show that $\Pr_{h_{a,b} \sim \mathcal{H}}[h_{a,b}(x) = h_{a,b}(y)] \leq 1/m$. Note that $|\mathcal{H}| = p(p-1)$.

1. Let $(a, b)$ (equivalently $h_{a,b}$) be *bad* for $x, y$ if $h_{a,b}(x) = h_{a,b}(y)$.
2. **Claim:** Number of bad $(a, b)$ is at most $p(p-1)/m$.
3. Total number of hash functions is $p(p-1)$ and hence probability of a collision is $\leq 1/m$. □

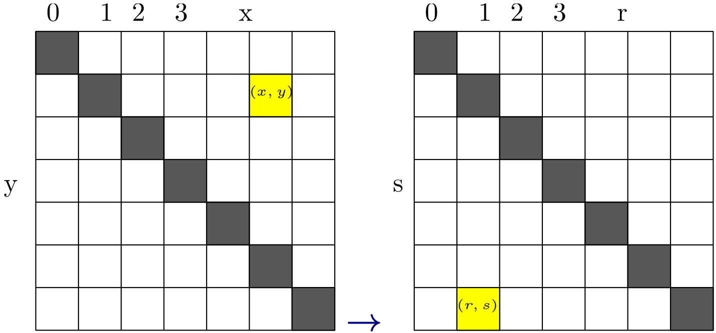
# Intuition for the Claim

$\mathbf{g_{a,b}(x) = (ax + b) \bmod p}$, $\mathbf{h_{a,b}(x) = (g_{a,b}(x)) \bmod m}$
First map $\mathbf{x \neq y}$ to $\mathbf{r = g_{a,b}(x)}$ and $\mathbf{s = g_{a,b}(y)}$. $\mathbf{r \neq s}$ (LemmaUnique)



# Intuition for the Claim

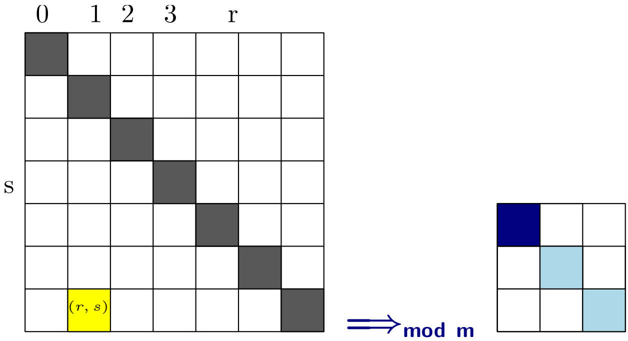
$g_{a,b}(x) = (ax + b) \bmod p$, $h_{a,b}(x) = (g_{a,b}(x)) \bmod m$
First map $x \neq y$ to $r = g_{a,b}(x)$ and $s = g_{a,b}(y)$. $r \neq s$ (LemmaUnique)




As $(a, b)$ varies, $(r, s)$ takes all possible $p(p-1)$ values. Since $(a, b)$ is picked u.a.r., every value of $(r, s)$ has equal probability.

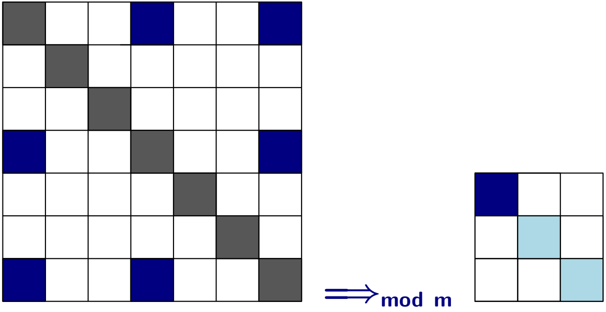
# Intuition for the Claim

$\mathbf{g_{a,b}(x) = (ax + b) \bmod p, \;\; h_{a,b}(x) = (g_{a,b}(x)) \bmod m}$



# Intuition for the Claim

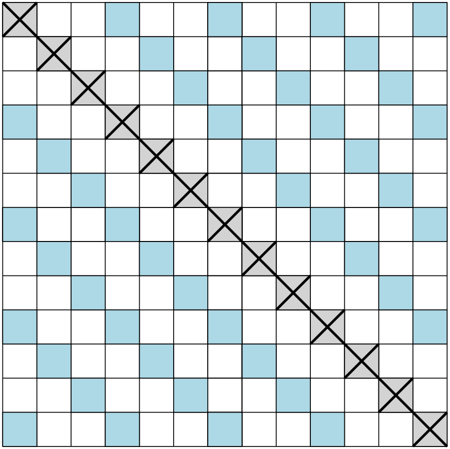
$$g_{a,b}(x) = (ax + b) \bmod p, \quad h_{a,b}(x) = (g_{a,b}(x)) \bmod m$$



# Intuition for the Claim

$g_{a,b}(x) = (ax + b) \bmod p$, $h_{a,b}(x) = (g_{a,b}(x)) \bmod m$

1. First part of mapping maps $(x, y)$ to a random location $(g_{a,b}(x), g_{a,b}(y))$ in the "matrix".
2. $(g_{a,b}(x), g_{a,b}(y))$ is not on main diagonal.
3. All blue locations are "bad" – map by **mod m** to a location of collusion.
4. But... at most $1/m$ fraction of allowable locations in the matrix are bad.



# We need

## to show at most 1/m fraction of bad $h_{a,b}$

$$h_{a,b}(x) = (((ax + b) \bmod p) \bmod m)$$

2 lemmas ...

Fix $x \neq y \in \mathbb{Z}_p$, and let $r = (ax + b) \bmod p$ and $s = (ay + b) \bmod p$.

# We need

to show at most 1/m fraction of bad $h_{a,b}$

$$h_{a,b}(x) = (((ax + b) \bmod p) \bmod m)$$

2 lemmas ...

Fix $x \neq y \in \mathbb{Z}_p$, and let $r = (ax + b) \bmod p$ and $s = (ay + b) \bmod p$.

1. 1-to-1 correspondence between $p(p-1)$ pairs of $(a, b)$ (equivalently $h_{a,b}$) and $p(p-1)$ pairs of $(r, s)$.

# We need

## to show at most 1/m fraction of bad $h_{a,b}$

$$h_{a,b}(x) = (((ax + b) \bmod p) \bmod m)$$

2 lemmas ...

Fix $x \neq y \in \mathbb{Z}_p$, and let $r = (ax + b) \bmod p$ and $s = (ay + b) \bmod p$.

1. 1-to-1 correspondence between $p(p-1)$ pairs of $(a, b)$ (equivalently $h_{a,b}$) and $p(p-1)$ pairs of $(r, s)$.
2. Out of all possible $p(p-1)$ pairs of $(r, s)$, at most $p(p-1)/m$ fraction satisfies $r \bmod m = s \bmod m$.

# Some Lemmas

## Lemma

*If* $\mathbf{x} \neq \mathbf{y}$ *then for any* $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_\mathbf{p}$ *such that* $\mathbf{a} \neq \mathbf{0}$*, we have*

$$\mathbf{ax + b \mod p \neq ay + b \mod p}.$$

## Proof.

If $\mathbf{ax + b \mod p = ay + b \mod p}$ then $\mathbf{a(x - y) \mod p = 0}$ and $\mathbf{a \neq 0}$ and $\mathbf{(x - y) \neq 0}$. However, $\mathbf{a}$ and $\mathbf{(x - y)}$ cannot divide $\mathbf{p}$ since $\mathbf{p}$ is prime and $\mathbf{a < p}$ and $\mathbf{(x - y) < p}$. □

# Some Lemmas

**Lemma**

*If* $x \neq y$ *then for each* $(r, s)$ *such that* $r \neq s$ *and* $0 \leq r, s \leq p - 1$ *there is exactly* **one** $a, b$ *such that*

$$ax + b \mod p = r \text{ and } ay + b \mod p = s.$$

**Proof.**

Solve the two equations:

$$ax + b = r \mod p \quad \text{and} \quad ay + b = s \mod p$$

We get $a = \frac{r-s}{x-y} \mod p$ and $b = r - ax \mod p$. □

One-to-one correspondence between $(a, b)$ and $(r, s)$

# Understanding the hashing

Once we fix $\mathbf{a}$ and $\mathbf{b}$, and we are given a value $\mathbf{x}$, we compute the hash value of $\mathbf{x}$ in two stages:

1. **Compute**: $\mathbf{r} \leftarrow (\mathbf{ax} + \mathbf{b}) \mod \mathbf{p}$.
2. **Fold**: $\mathbf{r}' \leftarrow \mathbf{r} \mod \mathbf{m}$

## Collision...

Given two distinct values $\mathbf{x}$ and $\mathbf{y}$ they might collide only because of folding.

## Lemma

*# not equal pairs* $(\mathbf{r}, \mathbf{s})$ *of* $\mathbb{Z}_\mathbf{p} \times \mathbb{Z}_\mathbf{p}$ *that are folded to the same number is* $\mathbf{p}(\mathbf{p} - 1)/\mathbf{m}$.

# Folding numbers

## Lemma

*# pairs* $(\mathbf{r}, \mathbf{s}) \in \mathbb{Z}_p \times \mathbb{Z}_p$ *such that* $\mathbf{r} \neq \mathbf{s}$ *and* $\mathbf{r} \mod \mathbf{m} = \mathbf{s} \mod \mathbf{m}$ *(folded to the same number) is* $\mathbf{p}(\mathbf{p}-1)/\mathbf{m}$.

## Proof.

Consider a pair $(\mathbf{r}, \mathbf{s}) \in \{0, 1, \ldots, \mathbf{p}-1\}^2$ s.t. $\mathbf{r} \neq \mathbf{s}$. Fix $\mathbf{r}$:

1. $\mathbf{a} = \mathbf{r} \mod \mathbf{m}$.
2. There are $\lceil \mathbf{p}/\mathbf{m} \rceil$ values of $\mathbf{s}$ that fold into $\mathbf{a}$. That is

$$\mathbf{r} \mod \mathbf{m} = \mathbf{s} \mod \mathbf{m}.$$

3. One of them is when $\mathbf{r} = \mathbf{s}$.
4. $\implies$ # of colliding pairs $(\lceil \mathbf{p}/\mathbf{m} \rceil - 1)\mathbf{p} \leq (\mathbf{p}-1)\mathbf{p}/\mathbf{m}$

$\square$

# Proof of Claim

## # of bad pairs is $\mathbf{p(p-1)/m}$

### Proof.

Let $\mathbf{a, b} \in \mathbb{Z}_\mathbf{p}$ such that $\mathbf{a \neq 0}$ and $\mathbf{h_{a,b}(x) = h_{a,b}(y)}$.

1. Let $\mathbf{r = ax + b \mod p}$ and $\mathbf{s = ay + b \mod p}$.
2. Collision if and only if $\mathbf{r \mod m = s \mod m}$.
3. (Folding error): Number of pairs $\mathbf{(r, s)}$ such that $\mathbf{r \neq s}$ and $\mathbf{0 \leq r, s \leq p - 1}$ and $\mathbf{r \mod m = s \mod m}$ is $\mathbf{p(p-1)/m}$.
4. From previous lemma there is one-to-one correspondence between $\mathbf{(a, b)}$ and $\mathbf{(r, s)}$. Hence total number of bad $\mathbf{(a, b)}$ pairs is $\mathbf{p(p-1)/m}$.

□

# Proof of Claim

## # of bad pairs is $p(p-1)/m$

### Proof.

Let $a, b \in \mathbb{Z}_p$ such that $a \neq 0$ and $h_{a,b}(x) = h_{a,b}(y)$.

1. Let $r = ax + b \mod p$ and $s = ay + b \mod p$.
2. Collision if and only if $r \mod m = s \mod m$.
3. (Folding error): Number of pairs $(r, s)$ such that $r \neq s$ and $0 \leq r, s \leq p-1$ and $r \mod m = s \mod m$ is $p(p-1)/m$.
4. From previous lemma there is one-to-one correspondence between $(a, b)$ and $(r, s)$. Hence total number of bad $(a, b)$ pairs is $p(p-1)/m$.

□

Prob of $x$ and $y$ to collide: $\frac{\#\text{ bad }(a,b)\text{ pairs}}{\#(a,b)\text{ pairs}} = \frac{p(p-1)/m}{p(p-1)} = \frac{1}{m}$.

# Look up Time

Say $|S| = |T| = m$.
For $0 \leq i \leq m - 1$, $\ell(i)$ : number of elements hashed to slot $i$ in $T$.

## Expected look up time

Since for $x \neq y$, $\Pr\left[h_{a,b}(x) = h_{a,b(y)}\right] = 1/m$, we get $E[\ell(i)] = |S|/m = 1$.

# Look up Time

Say $|S| = |T| = m$.
For $0 \le i \le m-1$, $\ell(i)$ : number of elements hashed to slot $i$ in $T$.

## Expected look up time

Since for $x \neq y$, $\Pr\left[h_{a,b}(x) = h_{a,b(y)}\right] = 1/m$, we get $E[\ell(i)] = |S|/m = 1$.

## Expected worst case look up time

Like in *Balls & Bins*, $E\left[\max_{i=0}^{m-1} \ell(i)\right] \ge O(\ln n / \ln \ln n)$.

# Look up Time

Say $|S| = |T| = m$.
For $0 \leq i \leq m-1$, $\ell(i)$ : number of elements hashed to slot $i$ in $T$.

## Expected look up time

Since for $x \neq y$, $\Pr\left[h_{a,b}(x) = h_{a,b(y)}\right] = 1/m$, we get $E[\ell(i)] = |S|/m = 1$.

## Expected worst case look up time

Like in *Balls & Bins*, $E\left[\max_{i=0}^{m-1} \ell(i)\right] \geq O(\ln n / \ln \ln n)$.

## What if $|T| = m^2$ (# Bins is $m^2$)

**Claim:** If $|T| = m^2$, then $E\left[\max_{i=0}^{m-1} \ell(i)\right] = O(1)$.

# Perfect Hashing

## Two levels of hash tables

**Question:** Can we make look up time $\mathbf{O(1)}$ in worst case?

### Perfect Hashing for Static Data

- Do hashing once.
- If $\mathbf{Y_i} = |\ell(\mathbf{i})| > \mathbf{10}$ then hash elements of $\ell(\mathbf{i})$ to a table of $\mathbf{Y_i^2}$ size.

# Perfect Hashing

## Two levels of hash tables

**Question:** Can we make look up time $\mathbf{O(1)}$ in worst case?

### Perfect Hashing for Static Data

- Do hashing once.
- If $\mathbf{Y_i} = |\ell(\mathbf{i})| > \mathbf{10}$ then hash elements of $\ell(\mathbf{i})$ to a table of $\mathbf{Y_i^2}$ size.

### Lemma

*Worst case expected look up time is* $\mathbf{O(1)}$.

# Perfect Hashing

## Two levels of hash tables

**Question:** Can we make look up time $\mathbf{O(1)}$ in worst case?

### Perfect Hashing for Static Data

- Do hashing once.
- If $\mathbf{Y_i} = |\ell(\mathbf{i})| > \mathbf{10}$ then hash elements of $\ell(\mathbf{i})$ to a table of $\mathbf{Y_i^2}$ size.

### Lemma

*Worst case expected look up time is* $\mathbf{O(1)}$.

### Lemma

*If* $|\mathbf{S}| = \mathbf{O(m)}$ *then space usage of perfect hashing is* $\mathbf{O(m)}$.

# Intuition: Throwing $\mathbf{m}$ Balls in to $\mathbf{m^2}$ Bins

- $\mathbf{Pr[i}$th ball lands in $\mathbf{j}$th bin$\mathbf{]}$

# Intuition: Throwing $\mathbf{m}$ Balls in to $\mathbf{m^2}$ Bins

- $\mathbf{Pr[i\text{th ball lands in } j\text{th bin}] = 1/m^2}$
- For a fixed bin $\mathbf{j}$, $\mathbf{Y_j} = \#$ balls in bin $\mathbf{j}$.

# Intuition: Throwing $\mathbf{m}$ Balls in to $\mathbf{m^2}$ Bins

- $\Pr[\mathbf{i}\text{th ball lands in } \mathbf{j}\text{th bin}] = 1/m^2$
- For a fixed bin $\mathbf{j}$, $\mathbf{Y_j} = \#$ balls in bin $\mathbf{j}$. $\mathbf{E[Y_j] = 1/m}$.

# Intuition: Throwing $\mathbf{m}$ Balls in to $\mathbf{m^2}$ Bins

- $\Pr[i\text{th ball lands in } j\text{th bin}] = 1/m^2$
- For a fixed bin $j$, $Y_j = \#$ balls in bin $j$. $E[Y_j] = 1/m$.
- For $c \geq 3$, let $\delta = cm - 1$. $\Pr[Y_j > c]$

$$
\begin{aligned}
\Pr[Y_j > cm/m] &= \Pr[Y_j > (1+\delta)\, E[Y_j]] \\
\text{(Chernoff)} &< \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu} \\
&= \left(\frac{e^{(cm-1)}}{(cm)^{cm}}\right)^{1/m} \leq (e/c)^c(1/m^c) \\
&\leq 1/m^3
\end{aligned}
$$

# Intuition: Throwing $\mathbf{m}$ Balls in to $\mathbf{m^2}$ Bins

- $\Pr[i\text{th ball lands in } j\text{th bin}] = 1/m^2$
- For a fixed bin $j$, $Y_j = \#$ balls in bin $j$. $E[Y_j] = 1/m$.
- For $c \geq 3$, let $\delta = cm - 1$. $\Pr[Y_j > c]$

$$\begin{aligned}
\Pr[Y_j > cm/m] &= \Pr[Y_j > (1+\delta)\, E[Y_j]] \\
\text{(Chernoff)} &< \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu} \\
&= \left(\frac{e^{(cm-1)}}{(cm)^{cm}}\right)^{1/m} \leq (e/c)^c (1/m^c) \\
&\leq 1/m^3
\end{aligned}$$

- $\Pr\left[\max_{j=1}^{m^2} Y_j > c\right] \leq 1/m$ (Union bound).
- $\Pr\left[\max_{j=1}^{m^2} Y_j \leq c\right] \geq 1 - 1/m$ – (w.h.p.)
- $E[\max_j Y_j] \leq c + 1 = O(1)$.

# Rehashing, amortization and...

... making the hash table dynamic

So far we assumed fixed $S$ of size $\simeq m$.

**Question:** What happens as items are inserted and deleted?

1. If $|S|$ grows to more than $cm$ for some constant $c$ then hash table performance clearly degrades.
2. If $|S|$ stays around $\simeq m$ but incurs many insertions and deletions then the initial random hash function is no longer random enough!

# Rehashing, amortization and...

... making the hash table dynamic

So far we assumed fixed $S$ of size $\simeq m$.

**Question:** What happens as items are inserted and deleted?

1. If $|S|$ grows to more than $cm$ for some constant $c$ then hash table performance clearly degrades.
2. If $|S|$ stays around $\simeq m$ but incurs many insertions and deletions then the initial random hash function is no longer random enough!

**Solution:** Rebuild hash table periodically!

1. Choose a new table size based on current number of elements in table.
2. Choose a new random hash function and rehash the elements.
3. Discard old table and hash function.

**Question:** When to rebuild? How expensive?

# Rebuilding the hash table

1. Start with table size $\mathbf{m}$ where $\mathbf{m}$ is some estimate of $|\mathbf{S}|$ (can be some large constant).
2. If $|\mathbf{S}|$ grows to more than twice current table size, build new hash table (choose a new random hash function) with double the current number of elements. Can also use similar trick if table size falls below quarter the size.
3. If $|\mathbf{S}|$ stays roughly the same but more than $\mathbf{c}|\mathbf{S}|$ operations on table for some chosen constant $\mathbf{c}$ (say $\mathbf{10}$), rebuild.

The **amortize** cost of rebuilding to previously performed operations. Rebuilding ensures $\mathbf{O(1)}$ expected analysis holds even when $\mathbf{S}$ changes. Hence $\mathbf{O(1)}$ expected look up/insert/delete time *dynamic* data dictionary data structure!

# Bloom Filters

**Hashing:**

1. To insert $\mathbf{x}$ in dictionary store $\mathbf{x}$ in table in location $\mathbf{h(x)}$
2. To lookup $\mathbf{y}$ in dictionary check contents of location $\mathbf{h(y)}$

# Bloom Filters

**Hashing:**

1. To insert **x** in dictionary store **x** in table in location **h(x)**
2. To lookup **y** in dictionary check contents of location **h(y)**

**Bloom Filter:** tradeoff space for false positives

1. Storing items in dictionary expensive in terms of memory, especially if items are unwieldy objects such a long strings, images, etc with *non-uniform* sizes.
2. To insert **x** in dictionary set *bit* to **1** in location **h(x)** (initially all bits are set to **0**)
3. To lookup **y** if bit in location **h(y)** is **1** say yes, else no.

# Bloom Filters

# Bloom Filters

**Bloom Filter:** tradeoff space for false positives

1. To insert $\mathbf{x}$ in dictionary set *bit* to $\mathbf{1}$ in location $\mathbf{h(x)}$ (initially all bits are set to $\mathbf{0}$)
2. To lookup $\mathbf{y}$ if bit in location $\mathbf{h(y)}$ is $\mathbf{1}$ say yes, else no
3. No false negatives but false positives possible due to collisions

Reducing false positives:

1. Pick $\mathbf{k}$ hash functions $\mathbf{h_1, h_2, \ldots, h_k}$ *independently*
2. To insert $\mathbf{x}$ for $\mathbf{1 \leq i \leq k}$ set bit in location $\mathbf{h_i(x)}$ in table $\mathbf{i}$ to $\mathbf{1}$
3. To lookup $\mathbf{y}$ compute $\mathbf{h_i(y)}$ for $\mathbf{1 \leq i \leq k}$ and say yes only if each bit in the corresponding location is $\mathbf{1}$, otherwise say no. If probability of false positive for one hash function is $\alpha < 1$ then with $\mathbf{k}$ independent hash function it is $\alpha^{\mathbf{k}}$.

# Take away points

1. Hashing is a powerful and important technique for dictionaries. Many practical applications.
2. Randomization fundamental to understanding hashing.
3. Good and efficient hashing possible in theory and practice with proper definitions (universal, perfect, etc).
4. Related ideas of creating a compact fingerprint/sketch for objects is very powerful in theory and practice.

# Practical Issues

Hashing used typically for integers, vectors, strings etc.

- Universal hashing is defined for integers. To implement for other objects need to map objects in some fashion to integers (via representation)
- Practical methods for various important cases such as vectors, strings are studied extensively. See http://en.wikipedia.org/wiki/Universal_hashing for some pointers.
- Details on Cuckoo hashing and its advantage over chaining http://en.wikipedia.org/wiki/Cuckoo_hashing.
- Recent important paper bridging theory and practice of hashing. "The power of simple tabulation hashing" by Mikkel Thorup and Mihai Patrascu, 2011. See http://en.wikipedia.org/wiki/Tabulation_hashing
- Cryptographic hash functions have a different motivation and