Algorithms and Data Structures
Hash Map, Universal Hashing

Prof. Dr. Rolf Backofen
Bioinformatics Group / Department of Computer Science
Algorithms and Data Structures, November 2018
Structure

Associative Arrays
  Introduction
  Hash Map

Universal Hashing
  Introduction
  Probability Calculation
  Proof
  Examples
Reminder:

- An associative array is like a normal array, only that the indices are not $0, 1, 2, \ldots$, but different, e.g. telephone numbers.

Problem:

- Quickly find an element with a specific key.
- Naive solution: store pairs of key and value in a normal array.
- For $n$ keys searching requires $\Theta(n)$ time.
- With a hash map this just requires $\Theta(1)$ in the best case, regardless of how many elements are in the map!
Idea:
- Mapping the keys onto indices with a hash function
- Store the values at the calculated indices in a normal array

Example:
- Key set: \( x = \{3904433, 312692, 5148949\} \)
- Hash function: \( h(x) = x \mod 5 \), in the range \([0, \ldots, 4]\) 
- We need an array \( T \) with 5 elements.
  A “hash table” with 5 “buckets”
- The element with the key \( x \) is stored in \( T[h(x)] \)
Storage:

- `insert(3904433, "A")`: \( h(3904433) = 3 \Rightarrow T[3] = (3904433, "A") \)
- `insert(312692, "B")`: \( h(312692) = 2 \Rightarrow T[2] = (312692, "B") \)
- `insert(5148949, "C")`: \( h(5148949) = 4 \Rightarrow T[4] = (5148949, "C") \)

**Figure:** Hash table \( T \)
Searching:

- search(3904433): \( h(3904433) = 3 \Rightarrow T[3] \rightarrow (3904433, "A") \)

- search(123459): \( h(123459) = 4 \Rightarrow T[4] \)

\( \Rightarrow \) Value with key 123459 does not exist

- Search time for this example: \( \mathcal{O}(1) \)

Figure: Hash table T
Further inserting:

- insert(876543, "D"): $h(876543) = 3$  
  $\Rightarrow T[3] = (876543, "D") \Rightarrow$ Collision

- This happens more often than expected
  - **Birthday problem**: with 23 people we have the probability of 50% that 2 of them have birthday at the same day

*Figure*: Hash table $T$
**Associative Arrays**

**Hash Collisions**

**Problem:**
- Two keys are equal $h(x) = h(y)$ but not the values $x \neq y$

**Easiest Solution:**
- Represent each bucket as a list of key-value pairs
- Append new values to the end of the list

**Figure:** Hash table $T$

```
Table T:

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>312692</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>3904433</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>5148949</td>
<td>C</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>876543</td>
<td>D</td>
</tr>
</tbody>
</table>
```

November 2018 Prof. Dr. Rolf Backofen – Bioinformatics - University Freiburg - Germany
Best case:

- We have \( n \) keys which are equally distributed over \( m \) buckets.
- We have \( \approx \frac{n}{m} \) pairs per bucket.
- The runtime for searching is nearly \( \Theta(1) \) if not \( n \gg m \).
Worst case:

- All $n$ keys are mapped onto the same bucket
- The runtime is $\Theta(n)$ for searching

Worst case

$(m = 5, n = 10)$
Universal Hashing
Thought Experiment

Thought Experiment:

- A hash function is defined for a given key set
- Find a set of keys resulting in a degenerated hash table
  - The hash function stays fixed
  - For table size of 100: try $100 \times (99 + 1)$ different numbers
  - Worst case: all 100 key sets map to one bucket
- Now: find a solution to avoid that problem
Solution: universal hashing

- Out of a set of hash functions we randomly choose one
- The expected result of the hash function is an equal distribution over the buckets
- This hash function stays fixed for the lifetime of table
  Optional: copy table with new hash when degenerated
Universal Hashing

Definition

We call \( U \) the set (universe) of possible keys

- The size \( m \) of the hash table \( T \)
- Set of hash functions \( H = \{ h_1, h_2, \ldots, h_n \} \) with
  \[ h_i : U \to \{0, \ldots, m - 1\} \]

- Idea: runtime should be \( O(1 + \frac{|S|}{m}) \), where \( \frac{|S|}{m} \) is the table load

Figure: Hash function \( h_1 \)
Universal Hashing

Definition

- We choose two random keys $x, y \in \mathbb{U} \mid x \neq y$
- An average of 3 out of 15 functions produce collisions

Figure: Set of hash functions $H$
**Definition:** \( \mathcal{H} \) is \( c \)-universal if \( \forall x, y \in \mathbb{U} \mid x \neq y : \)

\[
\left| \left\{ h \in \mathcal{H} : h(x) = h(y) \right\} \right| \leq c \cdot \frac{1}{m}, \quad c \in \mathbb{R}
\]

Number of hash functions that create collisions

Number of hash functions

In other words, given an arbitrary but fixed pair \( x, y \). If \( h \in \mathcal{H} \) is chosen randomly then

\[
\text{Prob}(h(x) = h(y)) \leq c \cdot \frac{1}{m}
\]

**Note:** If the hash function assigns each key \( x \) and \( y \) randomly to buckets then:

\[
\text{Prob(Collision)} = \frac{1}{m} \iff c = 1
\]
Universal Hashing

Definition

- $\mathbb{U}$: key universe
- $\mathbb{S}$: used Keys
- $\mathbb{S}_i \subseteq \mathbb{S}$: keys mapping to Bucket $i$ (“synonyms”)
- Ideal would be $|\mathbb{S}_i| = \frac{|\mathbb{S}|}{m}$

Figure: Hash function $h \in \mathbb{H}$
Universal Hashing

Definition

- Let $\mathcal{H}$ be a $c$-universal class of hash functions
- Let $S$ be a set of keys and $h \in \mathcal{H}$ selected randomly
- Let $S_i$ be the key $x$ for which $h(x) = i$
- The expected average number of elements to search through per bucket is

$$
\mathbb{E}[|S_i|] \leq 1 + c \cdot \frac{|S|}{m}
$$

- Particularly: if $(m = \Omega(|S|))$ then $\mathbb{E}[|S_i|] = O(n)$
We just discuss the discrete case

- Probability space $\Omega$ with elementary (simple) events
- Events $e$ have probabilities ...

$$\sum_{e \in \Omega} P(e) = 1$$

The probability for a subset of events $E \subseteq \Omega$ is

$$P(E) = \sum_{e \in E} P(e) \mid e \in E$$

Table: throwing a dice

<table>
<thead>
<tr>
<th>$e$</th>
<th>$P(e)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\frac{1}{6}$</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{1}{6}$</td>
</tr>
<tr>
<td>3</td>
<td>$\frac{1}{6}$</td>
</tr>
<tr>
<td>4</td>
<td>$\frac{1}{6}$</td>
</tr>
<tr>
<td>5</td>
<td>$\frac{1}{6}$</td>
</tr>
<tr>
<td>6</td>
<td>$\frac{1}{6}$</td>
</tr>
</tbody>
</table>
Example:

- Rolling a dice twice ($\Omega = \{1, \ldots, 6\}^2$)
- Each event $e \in \Omega$ has the probability $P(e) = \frac{1}{36}$
- $E$ = if both results are even, then $P(E) =$

<table>
<thead>
<tr>
<th>$e$</th>
<th>$P(e)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1)</td>
<td>$\frac{1}{36}$</td>
</tr>
<tr>
<td>(1, 2)</td>
<td>$\frac{1}{36}$</td>
</tr>
<tr>
<td>(1, 3)</td>
<td>$\frac{1}{36}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(6, 5)</td>
<td>$\frac{1}{36}$</td>
</tr>
<tr>
<td>(6, 6)</td>
<td>$\frac{1}{36}$</td>
</tr>
</tbody>
</table>

Table: throwing a dice twice
Example:

- **Random variable**
  - Assigns a number to the result of an experiment
  - For example: \( X = \text{Sum of results for rolling twice} \)
  - \( X = 12 \) and \( X \geq 7 \) are regarded as events
  - Example 1: \( P(X = 2) = \)
  - Example 2: \( P(X = 4) = \)

**Table:** throwing a dice twice

<table>
<thead>
<tr>
<th>( e )</th>
<th>( P(e) )</th>
<th>( X )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1)</td>
<td>( \frac{1}{36} )</td>
<td>2</td>
</tr>
<tr>
<td>(1, 2)</td>
<td>( \frac{1}{36} )</td>
<td>3</td>
</tr>
<tr>
<td>(1, 3)</td>
<td>( \frac{1}{36} )</td>
<td>4</td>
</tr>
<tr>
<td>(6, 5)</td>
<td>( \frac{1}{36} )</td>
<td>11</td>
</tr>
<tr>
<td>(6, 6)</td>
<td>( \frac{1}{36} )</td>
<td>12</td>
</tr>
</tbody>
</table>
**Expected value** is defined as \( E(X) = \sum (k \cdot P(X = k)) \)

- Intuitive: the weighted average of possible values of \( X \), where the weights are the probabilities of the values

**Table: throwing a dice once**

<table>
<thead>
<tr>
<th>( X )</th>
<th>( P(X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \frac{1}{6} )</td>
</tr>
<tr>
<td>2</td>
<td>( \frac{1}{6} )</td>
</tr>
<tr>
<td>3</td>
<td>( \frac{1}{6} )</td>
</tr>
<tr>
<td>4</td>
<td>( \frac{1}{6} )</td>
</tr>
<tr>
<td>5</td>
<td>( \frac{1}{6} )</td>
</tr>
<tr>
<td>6</td>
<td>( \frac{1}{6} )</td>
</tr>
</tbody>
</table>

**Table: throwing a dice twice**

<table>
<thead>
<tr>
<th>( X )</th>
<th>( P(X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( \frac{1}{36} )</td>
</tr>
<tr>
<td>3</td>
<td>( \frac{2}{36} )</td>
</tr>
<tr>
<td>4</td>
<td>( \frac{3}{36} )</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>11</td>
<td>( \frac{2}{36} )</td>
</tr>
<tr>
<td>12</td>
<td>( \frac{1}{36} )</td>
</tr>
</tbody>
</table>

- Example "rolling once": \( E(X) = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + \cdots + 6 \cdot \frac{1}{6} = 3.5 \)
- Example "rolling twice": \( E(X) = 2 \cdot \frac{1}{36} + 3 \cdot \frac{2}{36} + \cdots + 12 \cdot \frac{1}{36} = 7 \)
Sum of expected values: for arbitrary discrete random variables $X_1, \ldots, X_n$ we can write:

$$E(X_1 + \cdots + X_n) = E(X_1) + \cdots + E(X_n)$$

Example: throwing two dice

- $X_1$: result of dice 1: $E(X_1) = 3.5$
- $X_2$: result of dice 2: $E(X_2) = 3.5$
- $X = X_1 + X_2$: total number
- Expected number when rolling two dices:

$$E(X) = E(X_1 + X_2) = E(X_1) + E(X_2) = 3.5 + 3.5 = 7$$
Corollary:

The probability of the event $E$ is $p = P(E)$. Let $X$ be the occurrences of the event $E$ and $n$ be the number of executions of the experiment. Then $E(X) = n \cdot P(E) = n \cdot p$

Example (Rolling the dice 60 times)

$$E(\text{occurrences of 6}) = \frac{1}{6} \cdot 60 = 10$$
Proof Corollary:

Indicator variable: $X_i$

$$X_i = \begin{cases} 
1, & \text{if event occurs} \\
0, & \text{else}
\end{cases}$$

$$\Rightarrow X = \sum_{i=1}^{n} X_i$$

$$\mathbb{E}(X) = \mathbb{E}\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} \mathbb{E}(X_i) \quad \text{def. } \mathbb{E}\text{-value} = \sum_{i=1}^{n} p = n \cdot p$$

Def. $\mathbb{E}$-value: $\mathbb{E}(X_i) = 0 \cdot P(X_i = 0) + 1 \cdot P(X_i = 1) = P(X_i = 1)$
Universal Hashing

Proof

Given:

- We pick two random keys \( x, y \in S \) \( x \neq y \) and a random hash function \( h \in \mathbb{H} \).
- We know the probability of a collision:

\[
P(h(x) = h(y)) \leq c \cdot \frac{1}{m}
\]

To proof:

\[
\mathbb{E}[|S_i|] \leq 1 + c \cdot \frac{|S|}{m} \quad \forall i
\]
We know:
\[ S_i = \{ x \in S : h(x) = i \} \]

If \( S_i = \emptyset \) \( \Rightarrow \) \( |S_i| = 0 \) otherwise, let \( x \in S_i \) be any key.

We define an indicator variable:
\[ I_y = \begin{cases} 1, & \text{if } h(y) = i \\ 0, & \text{else} \end{cases} \quad y \in S \setminus \{x\} \]

\[ \Rightarrow \quad |S_i| = 1 + \sum_{y \in S \setminus x} I_y \]

\[ \Rightarrow \quad \mathbb{E}(|S_i|) = \mathbb{E}(1 + \sum_{y \in S \setminus x} I_y) = 1 + \sum_{y \in S \setminus x} \mathbb{E}(I_y) \]
Universal Hashing

Proof

Auxiliary calculation:

\[ E[I_y] = P(I_y = 1) \]
\[ = P(h(y) = i) \]
\[ = P(h(y) = h(x)) \]
\[ \leq c \cdot \frac{1}{m} \]

Hence:

\[ E[|S_i|] = 1 + \sum_{y \in S \setminus x} E[I_y] \leq 1 + \sum_{y \in S \setminus x} c \cdot \frac{1}{m} \]
\[ = 1 + (|S| - 1) \cdot c \cdot \frac{1}{m} \]
\[ \leq 1 + |S| \cdot c \cdot \frac{1}{m} \]
\[ = 1 + c \cdot \frac{|S|}{m} \]

□
Negative example:
- The set of all $h$ for which $h_a(x) = (a \cdot x) \mod m$, for $a \in \mathbb{U}$
- It is not $c$-universal.
- If universal:
  \[
  \forall x, y \quad x \neq y : \quad \frac{|\{h \in \mathbb{H} : h(x) = h(y)\}|}{|\mathbb{H}|} \leq c \cdot \frac{1}{m}
  \]
- Which $x, y$ lead to a relative collision count bigger than $\frac{c}{m}$?
Positive example:

- Let $p$ be a big prime number, $p > m$ and $p \geq |\mathbb{U}|$
- Let $H$ be the set of all $h$ for which:

\[
    h_{a,b}(x) = ((a \cdot x + b) \mod p) \mod m,
    \]

where $1 \leq a < p$, $0 \leq b < p$

- This is $\approx 1$-universal, see Exercise 4.11 in Mehlhorn/Sanders
- E.g.: $U = \{0, ..., 99\}$, $p = 101$, $a = 47$, $b = 5$
- Then $h(x) = ((47 \cdot x + 5) \mod 101) \mod m$
- Easy to implement but hard to proof
- Exercise: show empirically that it is 2-universal
Universal Hashing

Examples

Positive example:

- The set of hash functions is \( c \)-universal:

\[
h_a(x) = a \cdot x \mod m, \quad a \in \mathbb{U}
\]

- We define:

\[
a = \sum_{0,\ldots,k-1} a_i \cdot m^i, \quad k = \text{ceil}(\log_m |\mathbb{U}|)
\]

\[
x = \sum_{0,\ldots,k-1} x_i \cdot m^i
\]

- Intuitive: scalar product with base \( m \)

\[
a \cdot x = \sum_{0,\ldots,k-1} a_i \cdot x_i
\]
Example \((\mathbb{U} = \{0, \ldots, 999\}, \ m = 10, \ a = 348)\)

With \(a = 348\): \(a_2 = 3, \ a_1 = 4, \ a_0 = 8\)

\[
h_{348}(x) = (a_2 \cdot x_2 + a_1 \cdot x_1 + a_0 \cdot x_0) \mod m
= (3x_2 + 4x_1 + 8x_0) \mod 10
\]

With \(x = 127\): \(x_2 = 1, \ x_1 = 2, \ x_0 = 7\)

\[
h_{348}(127) = (3 \cdot x_2 + 4 \cdot x_1 + 8 \cdot x_0) \mod 10
= (3 \cdot 1 + 4 \cdot 2 + 8 \cdot 7) \mod 10
= 7
\]
Course literature

Introduction to Algorithms. 

[MS08] Kurt Mehlhorn and Peter Sanders. 
Algorithms and data structures, 2008. 
Further Literature

- **Hash Map - Theory**
  
  [Wik] Hash table
  https://en.wikipedia.org/wiki/Hash_table

- **Hash Map - Implementations / API**
  
  [Cpp] C++ - hash_map
  http://www.sgi.com/tech/stl/hash_map.html

  [Jav] Java - HashMap
  https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html

  [Pyt] Python - Dictionaries (Hash table)
  https://en.wikipedia.org/wiki/Hash_table