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

How do we build a Map?

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!

Associative Arrays

The Hash 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: $h(x) = x \bmod 5$, 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 $T[h(x)]$

Associative Arrays

The Hash Map

Storage:

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

Figure: Hash table T

Associative Arrays

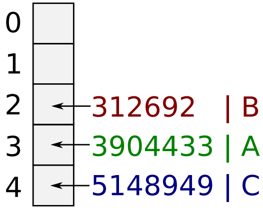
The Hash Map

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: $\mathcal{O}(1)$

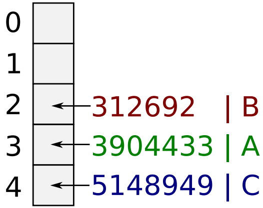
Figure: Hash table T



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

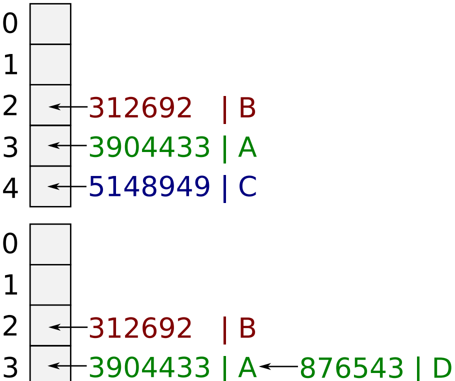
Problem:

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

Easiest Solution:

- Represent each bucket as list of key-value pairs
- Append new values to the end of the list

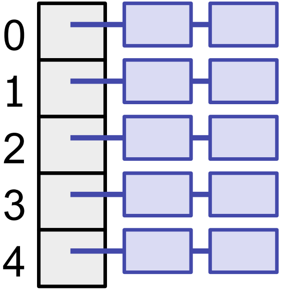
Figure: Hash table T



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 $\mathcal{O}(1)$ if **not** $n \gg m$

Best case
($m = 5$, $n = 10$)



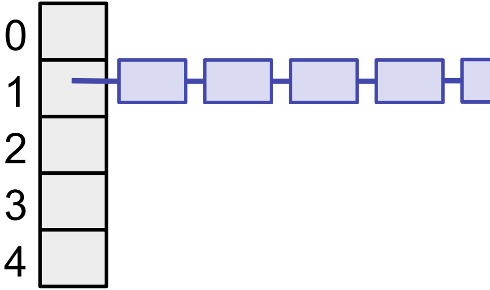
Associative Arrays

Expected Runtime

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

Universal Hashing

Idea

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

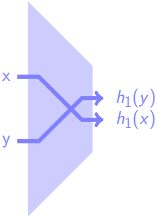


Figure: Hash func. 1

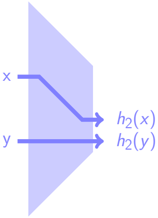


Figure: Hash func. 2

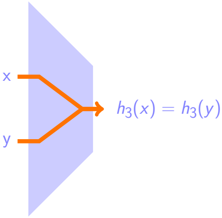


Figure: Hash func. coll.

Universal Hashing

Definition

Definition:

- We call $\mathbb{U}$ the set (universe) of possible keys
- The size m of the hash table T
- Set of hash functions $\mathbb{H} = \{h_1, h_2, \ldots, h_n\}$ with $h_i : \mathbb{U} \rightarrow \{0, \ldots, m-1\}$
- Idea: runtime should be $O(1 + \frac{|\mathbb{S}|}{m})$, where $\frac{|\mathbb{S}|}{m}$ is the table load

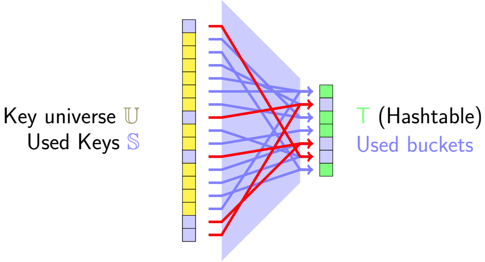


Figure: Hash function h_1

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

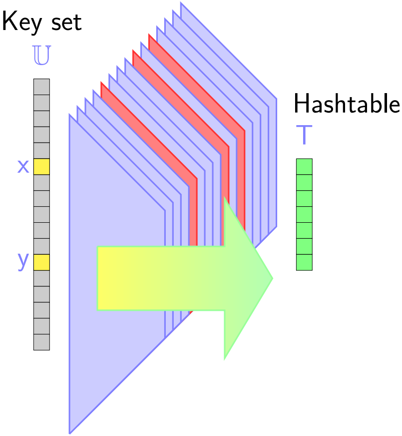


Figure: Set of hash functions $\mathbb{H}$

Universal Hashing

Definition

Definition: $\mathbb{H}$ is c-universal if $\forall x, y \in \mathbb{U} \mid x \neq y$:

$$\frac{\overbrace{|\{h \in \mathbb{H} \colon h(x) = h(y)\}|}^{\text{Number of hash functions that create collisions}}}{\underbrace{|\mathbb{H}|}_{\text{Number of hash functions}}} \leq c \cdot \frac{1}{m}, \qquad c \in \mathbb{R}$$

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

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

Note: If the hash function assigns each key x and y randomly to buckets then:
$Prob(Collision) = \frac{1}{m} \Leftrightarrow c = 1$

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 $|\mathbb{S}_i| = \frac{|\mathbb{S}|}{m}$

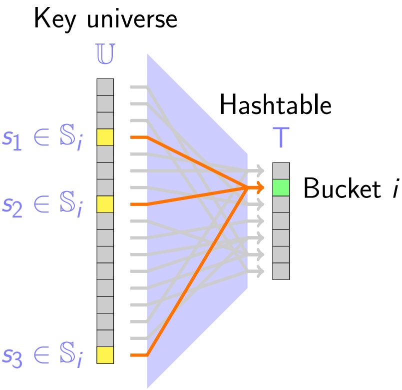


Figure: Hash function $h \in \mathbb{H}$

Universal Hashing

Definition

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

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

- Particulary: if $(m = \Omega(|\mathbb{S}|))$ then $\mathbb{E}\left[|\mathbb{S}_i|\right] = \mathcal{O}(n)$

Universal Hashing

Probability Calculation

- We just discuss the discrete case
- Probability space Ω 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

e	$P(e)$
1	$1/6$
2	$1/6$
3	$1/6$
4	$1/6$
5	$1/6$
6	$1/6$

Universal Hashing

Probability Calculation

Example:

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

Table: throwing a dice twice

e	$P(e)$
$(1, 1)$	${}^1/_{36}$
$(1, 2)$	${}^1/_{36}$
$(1, 3)$	${}^1/_{36}$
$\ldots$	$\ldots$
$(6, 5)$	${}^1/_{36}$
$(6, 6)$	${}^1/_{36}$

Example:

- Random variable
  - Assigns a number to the result of an experiment
  - For example: $X =$ 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

e	$P(e)$	X
$(1,1)$	$^1/_{36}$	2
$(1,2)$	$^1/_{36}$	3
$(1,3)$	$^1/_{36}$	4
$\dots$	$\dots$	$\dots$
$(6,5)$	$^1/_{36}$	11
$(6,6)$	$^1/_{36}$	12

Universal Hashing

Probability Calculation

Expected value is defined as $\mathbb{E}(X) = \sum (k \cdot P(X = k))$

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

Table: throwing a dice once

X	$P(X)$
1	$^1/_6$
2	$^1/_6$
3	$^1/_6$
4	$^1/_6$
5	$^1/_6$
6	$^1/_6$

Table: throwing a dice twice

X	$P(X)$
2	$^1/_{36}$
3	$^2/_{36}$
4	$^3/_{36}$
...	...
11	$^2/_{36}$
12	$^1/_{36}$

- Example "rolling once": $\mathbb{E}(X) = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + \cdots + 6 \cdot \frac{1}{6} = 3.5$
- Example "rolling twice": $\mathbb{E}(X) = 2 \cdot \frac{1}{36} + 3 \cdot \frac{2}{36} + \cdots + 12 \cdot \frac{1}{36} = 7$

Sum of expected values: for arbitrary discrete random variables $X_1, \ldots, X_n$ we can write:

$$\mathbb{E}\left(X_1 + \cdots + X_n\right) = \mathbb{E}(X_1) + \cdots + \mathbb{E}(X_n)$$

Example: throwing two dice

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

$$\mathbb{E}(X) = \mathbb{E}(X_1 + X_2) = \mathbb{E}(X_1) + \mathbb{E}(X_2) = 3.5 + 3.5 = 7$$

Universal Hashing

Probability Calculation

Corollary:

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

Example (Rolling the dice 60 times)

$$\mathbb{E}\,(\text{occurences of } 6) = \frac{1}{6} \cdot 60 = 10$$

Proof Corollary:

Indicator variable: X_i

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

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

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

$\square$

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 $x, y \in \mathbb{S} \mid x \neq y$ 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:

$$\mathbb{E}\left[|\mathbb{S}_i|\right] \leq 1 + c \cdot \frac{|\mathbb{S}|}{m} \qquad \forall\, i$$

Universal Hashing

Proof

We know:

$$\mathbb{S}_i = \{x \in \mathbb{S} : h(x) = i\}$$

If $\mathbb{S}_i = \emptyset \Rightarrow |\mathbb{S}_i| = 0$ otherwise, let $x \in \mathbb{S}_i$ be any key

We define an indicator variable:

$$I_y = \begin{cases} 1, & \text{if } h(y) = i \\ 0, & \text{else} \end{cases} \qquad y \in \mathbb{S} \setminus \{x\}$$

$$\Rightarrow \qquad |\mathbb{S}_i| = 1 + \sum_{y \in \mathbb{S} \setminus x} I_y$$

$$\Rightarrow \qquad \mathbb{E}\left(|\mathbb{S}_i|\right) = \mathbb{E}\left(1 + \sum_{y \in \mathbb{S} \setminus x} I_y\right) = 1 + \sum_{y \in \mathbb{S} \setminus x} \mathbb{E}(I_y)$$

Universal Hashing

Proof

Auxiliary calculation:

$$\begin{aligned} \mathbb{E}[I_y] &= P(I_y = 1) \\ &= P(h(y) = i) \\ &= P(h(y) = h(x)) \\ &\leq c \cdot \frac{1}{m} \end{aligned}$$

Hence:

$$\begin{aligned} \mathbb{E}\left[|\mathbb{S}_i|\right] = 1 + \sum_{y \in \mathbb{S} \setminus x} \mathbb{E}\left[I_y\right] &\leq 1 + \sum_{y \in \mathbb{S} \setminus x} c \cdot \frac{1}{m} \\ &= 1 + (|\mathbb{S}| - 1) \cdot c \cdot \frac{1}{m} \\ &\leq 1 + |\mathbb{S}| \cdot c \cdot \frac{1}{m} \\ &= 1 + c \cdot \frac{|\mathbb{S}|}{m} \qquad \square \end{aligned}$$

Negative example:

- The set of all h for which $h_a(x) = (a \cdot x) \mod m$, for a $a \in \mathbb{U}$
- It is not c-universal.
- If universal:

$$\forall x, y \quad x \neq y\colon \; \frac{|\{h \in \mathbb{H} : h(x) = h(y)\}|}{|\mathbb{H}|} \leq c \cdot \frac{1}{m}$$

- Which x, y lead to a relative collision count bigger than $\frac{c}{m}$?

Universal Hashing

Examples

Positive example:

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

$$h_{a,b}(x) = ((a \cdot x + b) \mod p) \mod m,$$
$$\text{where } 1 \leq a < p,\ 0 \leq b < p$$

- This is $\approx$ 1-universal, see Exercise 4.11 in Mehlhorn/Sanders
- E.g.: $U = \{0, ..., 99\}, \quad p = 101, \quad a = 47, b = 5$
- 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:

$$h_a(x) = a \bullet x \mod m, \qquad a \in \mathbb{U}$$

- **We define:**

$$a = \sum_{0,\ldots,k-1} a_i \cdot m^i, \qquad k = \mathrm{ceil}(\log_m |\mathbb{U}|)$$

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

- **Intuitive**: scalar product with base m

$$a \bullet x = \sum_{0,\ldots,k-1} a_i \cdot x_i$$

Universal Hashing

Examples

Example ($\mathbb{U} = \{0, \dots, 999\},\ m = 10,\ a = 348$)

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

$$\begin{aligned} h_{348}(x) &= (a_2 \cdot x_2 + a_1 \cdot x_1 + a_0 \cdot x_0) \mod m \\ &= (3\,x_2 + 4\,x_1 + 8\,x_0) \mod 10 \end{aligned}$$

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

$$\begin{aligned} h_{348}(127) &= (3 \cdot x_2 + 4 \cdot x_1 + 8 \cdot x_0) \mod 10 \\ &= (3 \cdot 1 + 4 \cdot 2 + 8 \cdot 7) \mod 10 \\ &= 7 \end{aligned}$$

Further Literature

- **Course literature**

[CRL01] Thomas H. Cormen, Ronald L. Rivest, and Charles E. Leiserson.
Introduction to Algorithms.
MIT Press, Cambridge, Mass, 2001.

[MS08] Kurt Mehlhorn and Peter Sanders.
Algorithms and data structures, 2008.
https://people.mpi-inf.mpg.de/~mehlhorn/ftp/Mehlhorn-Sanders-Toolbox.pdf.

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