# Algorithmns and Datastructures

## Hash Map, Universal Hashing

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Prof. Dr. Rolf Backofen

Bioinformatics Group / Department of Computer Science

Algorithmns and Datastructures, November 2016

# Structure

# Structure

- Feedback
  - Exercises
  - Lecture
- Associative Arrays
  - Introduction
  - Hash Map
- Universal Hashing
  - Introduction
  - Probability Calculation
  - Proof
  - Examples

# Feedback from the exercises

- Time effort from manageable to not manageable at all

# Feedback from the exercises

- Time effort from manageable to not manageable at all
- Complaints about Python, more programming examples needed
  → Acknowledged, but remember last year people had to deal with Java / C++

# Feedback from the exercises

- Time effort from manageable to not manageable at all
- Complaints about Python, more programming examples needed
  $\rightarrow$ Acknowledged, but remember last year people had to deal with Java / C++
- Exercise sheet instructions confusing / not very helpful
  $\rightarrow$ You are welcome to come up with your own implementation (e.g. use classes, other names). As long as you implement the required functionalities on your own (e.g. not relying on libraries) and test and document your code appropriately

# Feedback from the exercises

- Time effort from manageable to not manageable at all
- Complaints about Python, more programming examples needed
  $\rightarrow$ Acknowledged, but remember last year people had to deal with Java / C++
- Exercise sheet instructions confusing / not very helpful
  $\rightarrow$ You are welcome to come up with your own implementation (e.g. use classes, other names). As long as you implement the required functionalities on your own (e.g. not relying on libraries) and test and document your code appropriately
- For most using associative arrays (Python dictionary) tends to be a bit faster
  $\rightarrow$ Sorting the dictionary also takes time, depending on heterogeneity of the data (e.g. lots of locality names with

# Structure

**Reminder:**

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

**Reminder:**

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

**Problem:**

- Quickly find a element with a specific key

**Reminder:**

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

**Problem:**

- Quickly find a element with a specific key
- Naive solution: Store pairs of key and value in a normal field

UNI FREIBURG

**Reminder:**

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

**Problem:**

- Quickly find a element with a specific key
- Naive solution: Store pairs of key and value in a normal field
- For n keys searching requires $\Theta(n)$ time

**Reminder:**

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

**Problem:**

- Quickly find a element with a specific key
- Naive solution: Store pairs of key and value in a normal field
- For n keys searching requires $\Theta(n)$ time
- With a Hash Map this just requires $\Theta(1)$ in the best case, ... regardless how many elements are in the map!

# Structure

**Idea:**

- Mapping the keys onto indices with a hash function
- Store the values at the calculated indices in a normal array

**Example:**

- Key set: $x = \{3904433, 312692, 5148949\}$

**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 $[0, \ldots, 4]$

UNI FREIBURG

**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 "hashtable" with 5 "buckets"

**Idea:**

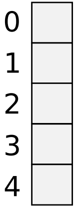
- Mapping the keys onto indices with a hash function
- Store the values at the calculated indices in a normal array

**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 "hashtable" with 5 "buckets"
- The element with the key x is stored in $T[h(x)]$

**Storage:**

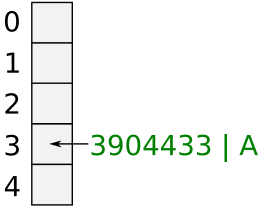
Figure: Hashtable T



**Storage:**

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

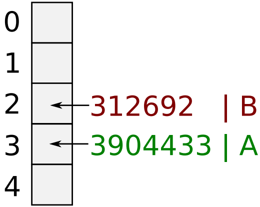
Figure: Hashtable T



**Storage:**

- insert(3904433,"A"): $h$(3904433) = 3 ⇒ T[3] = (3904433, "A")
- insert(312692, "B"): $h$(312692) = 2 ⇒ T[2] = (312692, "B")

Figure: Hashtable T



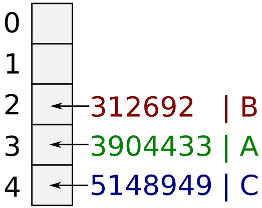
# Associative Arrays

## The Hash Map

**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: Hashtable T

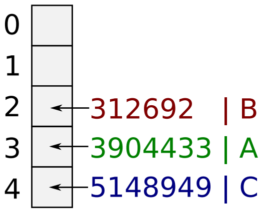


UNI FREIBURG

**Searching:**

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

Figure: Hashtable T



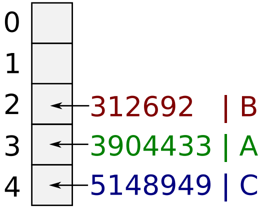
# Associative Arrays
## The Hash Map

**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

Figure: Hashtable T

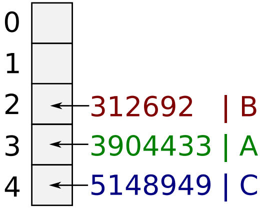


**Searching:**

- 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: $\mathcal{O}(1)$

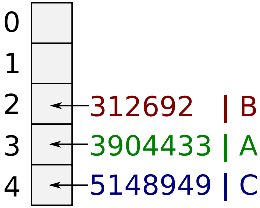
Figure: Hashtable T



**Further inserting:**

- insert(876543, "D"): $h(876543) = 3$

Figure: Hashtable T

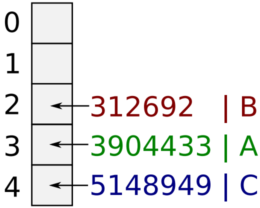


# Associative Arrays
## Hash Collisions

**Further inserting:**

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

Figure: Hashtable T

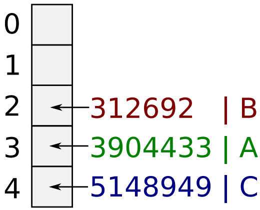


# Associative Arrays
## Hash Collisions

**Further inserting:**

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



**Problem:**

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

**Problem:**

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

**Easiest Solution:**

# Associative Arrays

## Hash Collisions

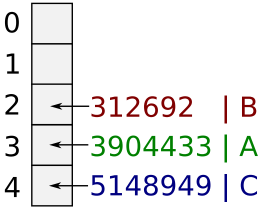
**Problem:**

- 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

Figure: Hashtable T



# Associative Arrays
## Hash Collisions

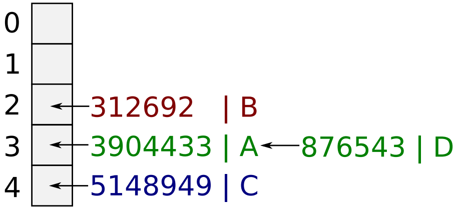
**Problem:**

- 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

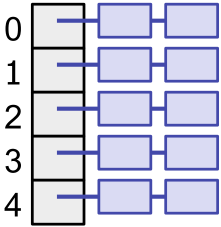
Figure: Hashtable T



Best case:

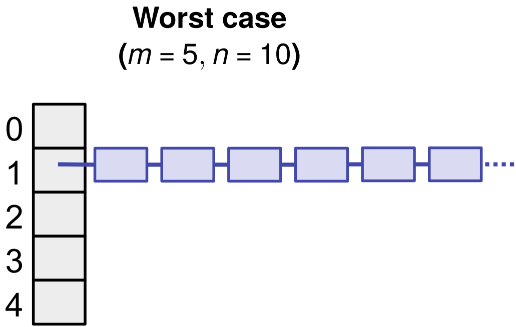
- We have $n$ keys which are equally distributed over $m$ buckets
- We have $\approx \frac{n}{m}$ pairs per bucket

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



**Worst case:**

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



# Structure

- I create a hash function

- I create a hash function
- Find a set of keys so that it results in a degenerated hash table

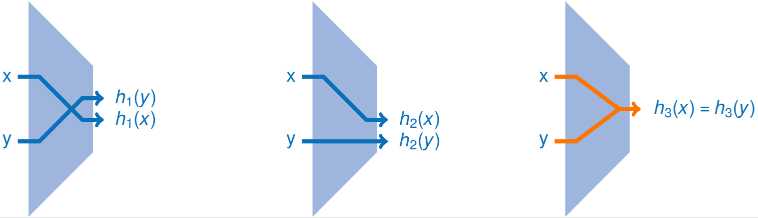
- I create a hash function
- Find a set of keys so that it results in a degenerated hash table
  - *you may use the hash function*

- I create a hash function
- Find a set of keys so that it results in a degenerated hash table
  - *you may use the hash function*
  - *for table size 100: try* $100 \times 99 + 1$ *different numbers*

- I create a hash function
- Find a set of keys so that it results in a degenerated hash table
  - *you may use the hash function*
  - *for table size 100: try* $100 \times 99 + 1$ *different numbers*
  - *worst case: still 100 must have same hash bucket*
- **Now:** Find a solution to avoid that problem

**Solution:**

- We use a set of hash functions

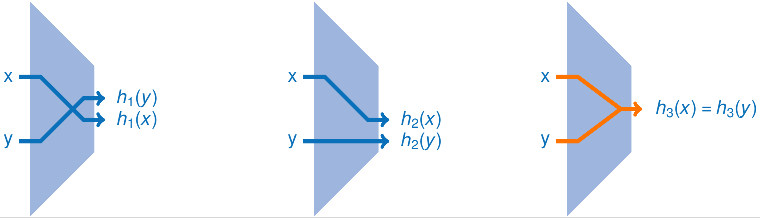


**Solution:**

- We use a set of hash functions
- We choose a random hash function so that the expected result is an equal distribution over the buckets

  *this is fixed for the lifetime of table*

  *optional: copy table with new hash when degenerated*

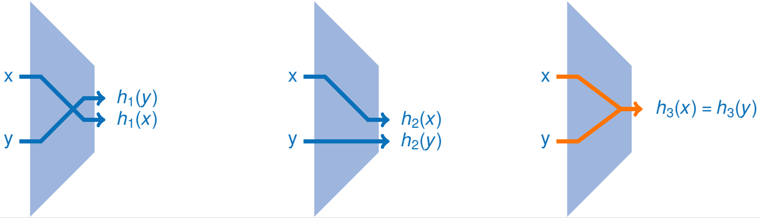


**Solution:**

- We use a set of hash functions
- We choose a random hash function so that the expected result is an equal distribution over the buckets

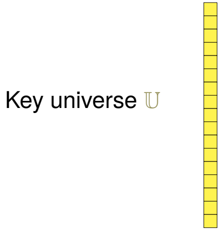
  *this is fixed for the lifetime of table*

  *optional: copy table with new hash when degenerated*
- This is called universal hashing



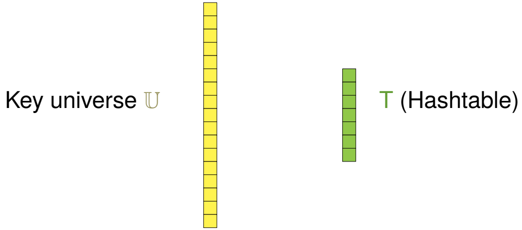
**Definition:**

- We call $\mathbb{U}$ the set (universum) of possible keys, and $\mathbb{S} \subseteq \mathbb{U}$ the set of used keys



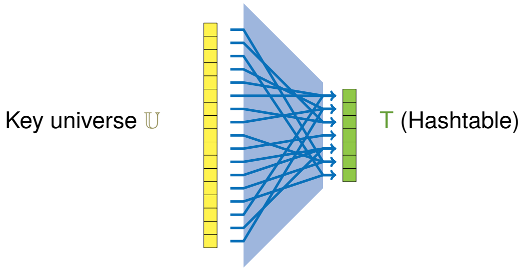
**Definition:**

- We call $\mathbb{U}$ the set (universum) of possible keys, and $\mathbb{S} \subseteq \mathbb{U}$ the set of used keys
- The size $m$ of the hash table $T$



**Definition:**

- We call $\mathbb{U}$ the set (universum) of possible keys, and $\mathbb{S} \subseteq \mathbb{U}$ the set of used keys
- The size $m$ of the hash table $T$
- Set of hash functions $\mathbb{H} = \{h_1, h_2, \ldots, h_n\}$ with $h_i : \mathbb{U} \to \{0, \ldots, m-1\}$

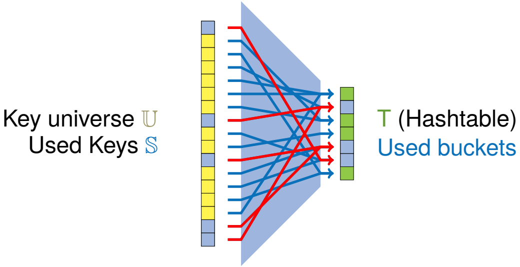


# Universal Hashing

## Definition

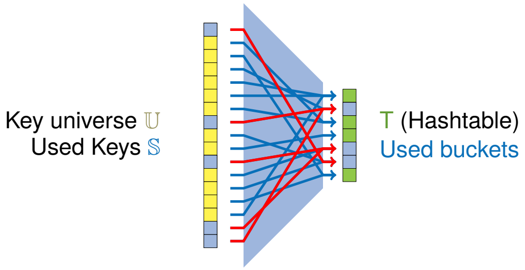
**Definition:**

- We call $\mathbb{U}$ the set (universum) of possible keys, and $\mathbb{S} \subseteq \mathbb{U}$ the set of used keys
- The size $m$ of the hash table $T$
- Set of hash functions $\mathbb{H} = \{h_1, h_2, \ldots, h_n\}$ with $h_i : \mathbb{U} \to \{0, \ldots, m-1\}$



**Definition:**

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



- We choose two random keys $x, y \in \mathbb{U} \mid x \neq y$

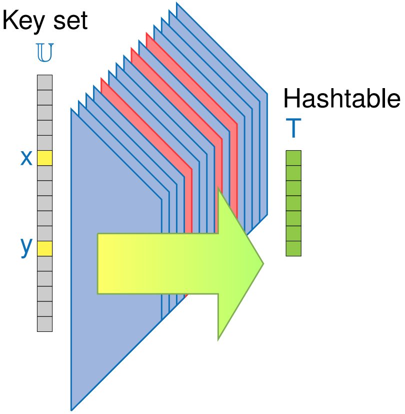



Figure: Set of hash functions $\mathbb{H}$

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

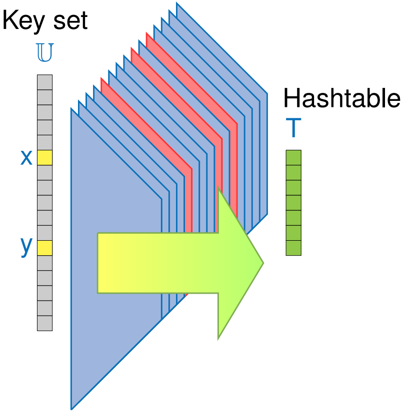



Figure: Set of hash functions $\mathbb{H}$

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

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

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

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

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

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

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

- With other words, given a arbitrary but fixed pair x,y. If $h \in \mathbb{H}$ is chosen randomly then $Prob(h(x) = h(y)) \leq c \cdot \frac{1}{m}$

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

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

- With other words, given a 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:

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

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

- With other words, given a 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:

$Prob(Collision) = \frac{1}{m} \Leftrightarrow c = 1$

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

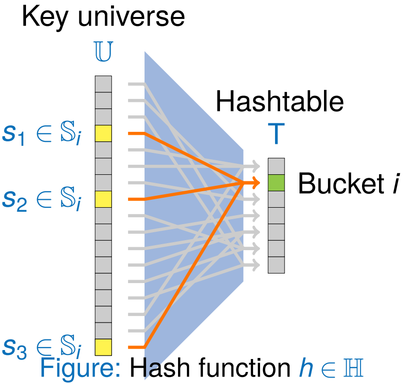



Figure: Hash function $h \in \mathbb{H}$

- Let $\mathbb{H}$ be a *c*-universal class of hash functions

- 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 $\mathbb{H}$ 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 $h(x) = i$

UNI FREIBURG

- 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

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

- 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

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

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

# Structure

- We just discuss the discrete case.

- We just discuss the discrete case.
- Probability space $\Omega$ with elementary (simple) events

- We just discuss the discrete case.
- Probability space $\Omega$ with elementary (simple) events
- Events have probabilities ..
  Condition

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

- We just discuss the discrete case.
- Probability space $\Omega$ with elementary (simple) events
- Events have probabilities ..
  Condition

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

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

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

- We just discuss the discrete case.
- Probability space $\Omega$ with elementary (simple) events
- Events have probabilities ..
  Condition

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

- 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 | $1/6$ |

**Example:**

**Example:**

- Rolling a dice twice ($\Omega = \{1, \ldots, 6\}^2$)

**Example:**

- Rolling a dice twice ($\Omega = \{1, \ldots, 6\}^2$)
- Each event $e \in \Omega$ has the probability $P(e) = {}^1\!/_{36}$

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:**

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

**Example:**

- Random variable
  - Assigns a number to the result of an experiment

**Example:**

- Random variable
  - Assigns a number to the result of an experiment
  - For example: $X$ = Sum of eye numbers for rolling twice

UNI FREIBURG

**Example:**

- Random variable
  - Assigns a number to the result of an experiment
  - For example: $X$ = Sum of eye numbers for rolling twice
  - $X = 12$ and $X \geq 7$ are then just events

Table: Throwing a dice twice

| $e$ | $P(e)$ | $X$ |
|---|---|---|
| $(1,1)$ | $1/36$ | 2 |
| $(1,2)$ | $1/36$ | 3 |
| $(1,3)$ | $1/36$ | 4 |
| ... | ... | ... |
| $(6,5)$ | $1/36$ | 11 |
| $(6,6)$ | $1/36$ | 12 |

**Example:**

- Random variable
  - Assigns a number to the result of an experiment
  - For example: $X$ = Sum of eye numbers for rolling twice
  - $X = 12$ and $X \geq 7$ are then just events
  - Example 1: $P(X = 2) =$

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 |

**Example:**

- Random variable
  - Assigns a number to the result of an experiment
  - For example: $X$ = Sum of eye numbers for rolling twice
  - $X = 12$ and $X \geq 7$ are then just 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 |
| ... | ... | ... |
| $(6,5)$ | $^1/_{36}$ | 11 |
| $(6,6)$ | $^1/_{36}$ | 12 |

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

**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

**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 | $^1/_{36}$ |

**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:

**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: $\mathbb{E}(X) = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + \cdots + 6 \cdot \frac{1}{6} = 3.5$

**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:

**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: $\mathbb{E}(X) = 2 \cdot \frac{1}{36} + 3 \cdot \frac{2}{36} + \cdots + 12 \cdot \frac{1}{36} = 7$

UNI FREIBURG

**Sum of expected values:** For independent (discrete) result variables $X_1, \ldots, X_n$ we can write:

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

**Sum of expected values:** For independent (discrete) result variables $X_1, \ldots, X_n$ we can write:

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

**Example:** Throwing two dice

**Sum of expected values:** For independent (discrete) result variables $X_1, \ldots, X_n$ we can write:

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

**Example:** Throwing two dice

- $X_1$: Expected number of eyes dice 1: $\mathbb{E}(X_1) = 3.5$

**Sum of expected values:** For independent (discrete) result variables $X_1, \ldots, X_n$ we can write:

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

**Example:** Throwing two dice

- $X_1$: Expected number of eyes dice 1: $\mathbb{E}(X_1) = 3.5$
- $X_2$: Expected number of eyes dice 2: $\mathbb{E}(X_2) = 3.5$

**Sum of expected values:** For independent (discrete) result variables $X_1, \ldots, X_n$ we can write:

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

**Example:** Throwing two dice

- $X_1$: Expected number of eyes dice 1: $\mathbb{E}(X_1) = 3.5$
- $X_2$: Expected number of eyes dice 2: $\mathbb{E}(X_2) = 3.5$
- $X = X_1 + X_2$: Expected total number of eyes:

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

## 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 $\mathbb{E}(X) = n \cdot P(E) = n \cdot p$

UNI FREIBURG

## 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: $X_i$

## Proof Corollary:

Indicator variable: $X_i$

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

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

UNI FREIBURG

## Proof Corollary:

Indicator variable: $X_i$

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

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

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

$\square$

UNI FREIBURG

## 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: $\mathbb{E}(X_i) = 0 \cdot P(X_i = 0) + 1 \cdot P(X_i = 1) = P(X_i = 1)$

# Structure

**Given:**

- We pick two random keys $x, y \in \mathbb{S} \mid x \neq y$ and a random hash function $h \in \mathbb{H}$

UNI FREIBURG

**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:

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

**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:**

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

**We know:**

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

**We know:**

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

if $\mathbb{S}_i = \emptyset \Rightarrow |\mathbb{S}_i| = 0$

**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 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:

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

**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:

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

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

**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:

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

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

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

**Auxiliary calculation:**

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

**Auxiliary calculation:**

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

**Auxiliary calculation:**

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

$$\Rightarrow 1 + \sum_{y \in \mathbb{S} \setminus x} \mathbb{E}(I_y) \leq 1 + \sum_{y \in \mathbb{S} \setminus x} c \cdot \frac{1}{m}$$

**Auxiliary calculation:**

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

$$\Rightarrow 1 + \sum_{y \in \mathbb{S} \setminus x} \mathbb{E}(I_y) \leq 1 + \sum_{y \in \mathbb{S} \setminus x} c \cdot \frac{1}{m}$$

$$= 1 + (|\mathbb{S}| - 1) \cdot c \cdot \frac{1}{m}$$

**Auxiliary calculation:**

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

$$\begin{aligned}\Rightarrow 1 + \sum_{y \in \mathbb{S} \setminus x} \mathbb{E}(I_y) &\leq 1 + \sum_{y \in \mathbb{S} \setminus x} c \cdot \frac{1}{m}\\ &= 1 + (|\mathbb{S}| - 1) \cdot c \cdot \frac{1}{m}\\ &\leq 1 + |\mathbb{S}| \cdot c \cdot \frac{1}{m} = 1 + c \cdot \frac{|\mathbb{S}|}{m}\end{aligned}$$

**Auxiliary calculation:**

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

$$\Rightarrow 1 + \sum_{y \in \mathbb{S} \setminus x} \mathbb{E}(I_y) \leq 1 + \sum_{y \in \mathbb{S} \setminus x} c \cdot \frac{1}{m}$$

$$= 1 + (|\mathbb{S}| - 1) \cdot c \cdot \frac{1}{m}$$

$$\leq 1 + |\mathbb{S}| \cdot c \cdot \frac{1}{m} = 1 + c \cdot \frac{|\mathbb{S}|}{m}$$

$$\mathbb{E}\left[|\mathbb{S}_i|\right] = 1 + \sum_{y \in \mathbb{S} \setminus \{x\}} \leq 1 + c \cdot \frac{|\mathbb{S}|}{m} \qquad \square$$

# Structure

**Negative example:**

**Negative example:**

- The set of all $h$ for which $h_a(x) = (a \cdot x) \mod m$, for a $a \in \mathbb{U}$

**Negative example:**

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

**Negative example:**

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

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

**Negative example:**

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

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

- 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 $p \geq |\mathbb{U}|$

**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:

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

**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

**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.: $U = \{0, ..., 99\}, \quad p = 101, \quad a = 47, 5$

**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.: $U = \{0, ..., 99\}, \quad p = 101, \quad a = 47, 5$
- Then $h(x) = ((47 \cdot x + 5) \mod 101) \mod m$

**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.: $U = \{0, ..., 99\}, \quad p = 101, \quad a = 47, 5$
- Then $h(x) = ((47 \cdot x + 5) \mod 101) \mod m$
- Easy to implement but hard to proof

UNI FREIBURG

**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.: $U = \{0, ..., 99\}, \quad p = 101, \quad a = 47, 5$
- 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

**Positive example:**

- The set of hash functions is $c$-universal:

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

**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:**

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

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

**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 $m$

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

UNI FREIBURG

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

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

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

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

- **General for this Lecture**


[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