

# Advanced Algorithms – COMS31900

---

## Hashing part one

Chaining, true randomness and universal hashing

---

Raphaël Clifford

Slides by Benjamin Sach and Markus Jalsenius

# Dictionaries

In a **dictionary** data structure we store  $(\text{key}, \text{value})$ -pairs

such that for any  $\text{key}$  there is at most one pair  $(\text{key}, \text{value})$  in the dictionary.

Often we want to perform the following three operations:

$\text{add}(x, v)$  Add the pair  $(x, v)$ .

$\text{lookup}(x)$  Return  $v$  if  $(x, v)$  is in dictionary, or  $\text{NULL}$  otherwise.

$\text{delete}(x)$  Remove pair  $(x, v)$  (assuming  $(x, v)$  is in dictionary).

There are many data structures that will do this job, e.g.:

- ▶ Linked lists
- ▶ Red-black trees
- ▶ Binary search trees
- ▶ Skip lists
- ▶ (2,3,4)-trees
- ▶ van Emde Boas trees (later in this course)

*these data structures all support extra operations beyond the three above*

but none of them take  $\mathcal{O}(1)$  worst case time for all operations...

so *maybe* there is room for improvement?

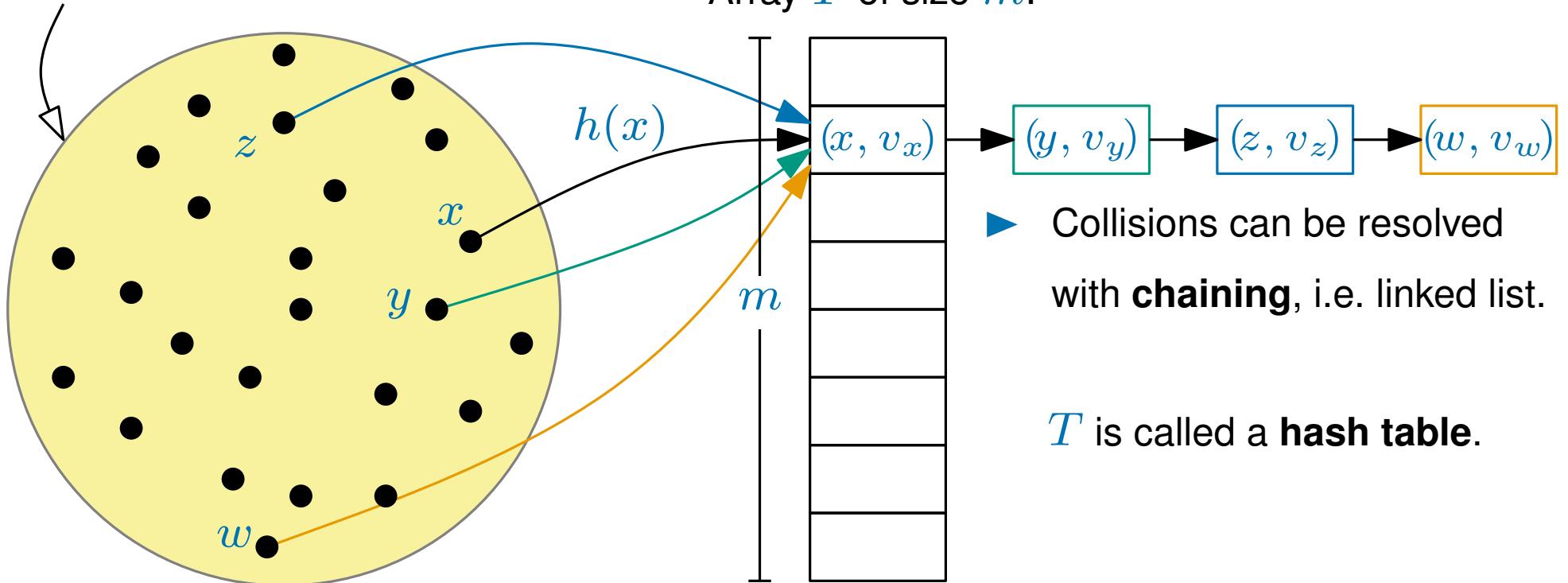
# Hash tables

We want to store  $n$  elements from the universe,  $U$  in a dictionary.

Typically  $u = |U|$  is much, much larger than  $n$ .

Universe  $U$  containing  $u$  keys.

Array  $T$  of size  $m$ .



- ▶ Collisions can be resolved with **chaining**, i.e. linked list.

$T$  is called a **hash table**.

A **hash function**  $h : U \rightarrow [m]$  maps a key to a position in  $T$ .

We write  $[m]$  to denote the set  $\{0, \dots, m - 1\}$ .

We want to avoid **collisions**, i.e.  $h(x) = h(y)$  for  $x \neq y$ .

## Time complexity

We cannot avoid collisions entirely since  $u \gg m$ ;

*some keys from the universe are bound to be mapped to the same position.*

(remember  $u$  is the size of the universe and  $m$  is the size of the table)

By building a hash table with chaining, we get the following time complexities:

Operation	Worst case time	Comment
$\text{add}(x, v)$	$O(1)$	Simply add item to the list link if necessary.
$\text{lookup}(x)$	$O(\text{length of chain containing } x)$	We might have to search through the whole list containing $x$ .
$\text{delete}(x)$	$O(\text{length of chain containing } x)$	Only $O(1)$ to perform the actual delete... but you have to find $x$ first

*So how long are these chains?*

# True randomness

## THEOREM

Consider any  $n$  fixed inputs to the hash table (which has size  $m$ ),  
i.e. any sequence of  $n$  add/lookup/delete operations.

Pick  $h$  uniformly at random from the set of all functions  $U \rightarrow [m]$ .

The expected run-time per operation is  $O(1 + \frac{n}{m})$ , or simply  $O(1)$  if  $m \geq n$ .

## PROOF

Let  $x, y$  be two distinct keys from  $U$ .  iff means if and only if.

Let indicator r.v.  $I_{x,y}$  be 1 iff  $h(x) = h(y)$ .

W

this is because  $h(x)$  and  $h(y)$  are chosen uniformly and independently from  $[m]$ .

Therefore,  $\mathbb{E}(I_{x,y}) = \Pr(I_{x,y} = 1) = \Pr(h(x) = h(y)) = \frac{1}{m}$ .

We have that,  $\mathbb{E}(I_{x,y}) = \frac{1}{m}$ .

# True randomness

## THEOREM

Consider any  $n$  fixed inputs to the hash table (which has size  $m$ ),  
i.e. any sequence of  $n$  add/lookup/delete operations.

Pick  $h$  uniformly at random from the set of all functions  $U \rightarrow [m]$ .

The expected run-time per operation is  $O(1 + \frac{n}{m})$ , or simply  $O(1)$  if  $m \geq n$ .

## PROOF

Let  $x, y$  be two distinct keys from  $U$ . iff means if and only if.

Let indicator r.v.  $I_{x,y}$  be 1 iff  $h(x) = h(y)$ . We have that,  $\mathbb{E}(I_{x,y}) = \frac{1}{m}$ .

Let  $N_x$  be the number of keys stored in  $T$  that are hashed to  $h(x)$

*so, in the worst case it takes  $N_x$  time to look up  $x$  in  $T$ .*

Observe that  $N_x = \sum_{y \in T} I_{x,y}$  the keys in  $T$

Finally, we have that  $\mathbb{E}(N_x) = \mathbb{E} \left( \sum_{y \in T} I_{x,y} \right)$  , linearity of expectation.

## Specifying the hash function

**Problem:** how do we specify an *arbitrary* (e.g. a truly random) hash function?

For each key in  $U$  we need to specify an arbitrary position in  $T$ ,  
this is a number in  $[m]$ , so requires  $\approx \log_2 m$  bits.

So in total we need  $\approx u \log_2 m$  bits, which is a ridiculous amount of space!

*(in particular, it's much bigger than the table :s)*

Why not pick the hash function as we go?

Couldn't we generate  $h(x)$  when we first see  $x$ ?

Wouldn't we only use  $n \log_2 m$  bits? *(one per key we actually store)*

*The problem with this approach is recalling  $h(x)$  the next time we see  $x$*

Essentially we'd need to build a dictionary to solve the dictionary problem!

This has become rather cyclic... let's try something else!

## Specifying the hash function

**Problem:** how do we specify an *arbitrary* (e.g. a truly random) hash function?

For each key in  $U$  we need to specify an arbitrary position in  $T$ ,  
this is a number in  $[m]$ , so requires  $\approx \log_2 m$  bits.

So in total we need  $\approx u \log_2 m$  bits, which is a ridiculous amount of space!

*(in particular, it's much bigger than the table :s)*

Instead, we define a set, or *family of hash functions*:  $H = \{h_1, h_2, \dots\}$ .

As part of initialising the hash table,  
we choose the hash function  $h$  from  $H$  randomly.

How should we specify the hash functions in  $H$  and how do we pick one at random?

## Weakly universal hashing

► A set  $H$  of hash functions is **weakly universal** if for any two distinct keys  $x, y \in U$ ,

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$

where  $h$  is chosen uniformly at random from  $H$ .

### OBSERVE

The randomness here comes from the fact that  $h$  is picked randomly.

### THEOREM

Consider any  $n$  fixed inputs to the hash table (which has size  $m$ ),

i.e. any sequence of  $n$  add/lookup/delete operations.

Pick  $h$  uniformly at random from a weakly universal set  $H$  of hash functions.

The expected run-time per operation is  $O(1)$  if  $m \geq n$ .

### PROOF

The proof we used for true randomness works here too (which is nice)

## Constructing a weakly universal family of hash functions

- ▶ Suppose  $U = [u]$ , i.e. the keys in the universe are integers  $0$  to  $u-1$ .
- ▶ Let  $p$  be any prime bigger than  $u$ .
- ▶ For  $a, b \in [p]$ , let

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

$$H_{p,m} = \{h_{a,b} \mid a \in \{1, \dots, p-1\}, b \in \{0, \dots, p-1\}\}.$$

### THEOREM

$H_{p,m}$  is a weakly universal set of hash functions.

### PROOF

See CLRS, Theorem 11.5, (page 267 in 3rd edition).

### OBSERVE

- ▶  $ax + b$  is a linear transformation which “spreads the keys” over  $p$  values when taken modulo  $p$ . This does not cause any collisions.
- ▶ Only when taken modulo  $m$  do we get collisions.

# True randomness vs. weakly universal hashing

For both,

## **true randomness**

( $h$  is picked uniformly from the set of all possible hash functions)

## **and weakly universal hashing**

( $h$  is picked uniformly from a weakly universal set of hash functions)

we have seen that when  $m \geq n$ ,

the expected lookup time in the hash table is  $O(1)$ .

*Since constructing a weakly universal set of hash functions seems much easier than obtaining true randomness, this is all good news!*

*isn't it?*

What about the length of the *longest* chain? (the longest linked list)

If it is very long, some lookups could take a very long time...

# Longest chain – true randomness

## LEMMA

If  $h$  is selected uniformly at random from all functions  $U \rightarrow [m]$  then,  
over  $m$  fixed inputs,

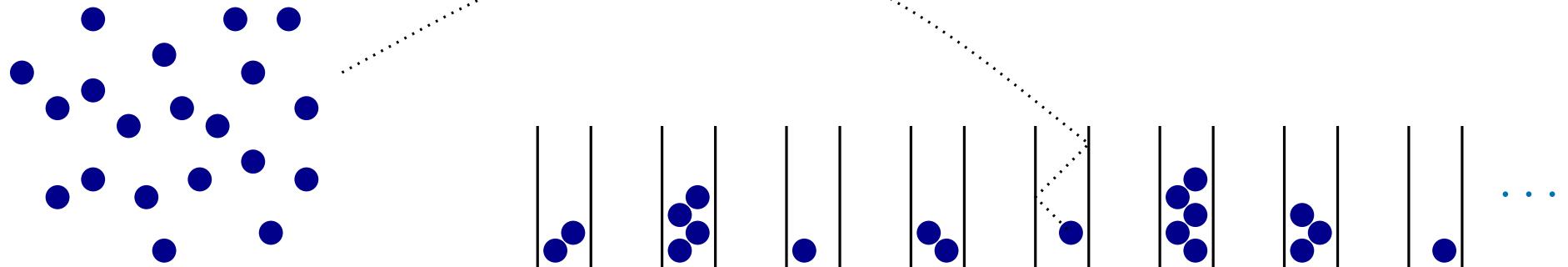
$$\Pr(\text{any chain has length } \geq 3 \log m) \leq \frac{1}{m}.$$

## OBSERVE

In this lemma we insert  $m$  keys, i.e.  $n = m$ .

## PROOF

The problem is equivalent to showing that if we randomly throw  $m$  balls into  $m$  bins, the probability of having a bin with at least  $3 \log m$  balls is at most  $\frac{1}{m}$ .



# Longest chain – true randomness

## PROOF

*continued...*

Let  $X_1$  be the number of balls in the first bin.

Choose any  $k$  of the  $m$  balls (we'll pick  $k$  in a bit)

the probability that all of these  $k$  balls go into the first bin is  $\frac{1}{m^k}$ .

So, the union bound gives us

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$

Number of subsets of size  $k$ .



## THEOREM

Let  $V_1, \dots, V_q$  be  $q$  events. Then

P

## Longest chain – true randomness

### PROOF

continued...

Let  $X_1$  be the number of balls in the first bin.

Choose any  $k$  of the  $m$  balls (we'll pick  $k$  in a bit)

the probability that all of these  $k$  balls go into the first bin is  $\frac{1}{m^k}$ .

So, the union bound gives us

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$

Number of subsets of size  $k$ .

(

$$\text{--- } k \text{ ---}$$

$$\leq \leq$$

## Longest chain – true randomness

### PROOF

continued...

Let  $X_1$  be the number of balls in the first bin.

Choose any  $k$  of the  $m$  balls (we'll pick  $k$  in a bit)

the probability that all of these  $k$  balls go into the first bin is  $\frac{1}{m^k}$ .

So, the union bound gives us

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$

Number of subsets of size  $k$ .

By using the union bound *again*, we have that

$$\Pr(\text{at least one bin receives at least } k \text{ balls}) \leq m \cdot \Pr(X_1 \geq k) \leq \frac{m}{k!}.$$

- Now we set  $k = 3 \log m$  and observe that  $\frac{m}{k!} \leq \frac{1}{m}$  for  $m \geq 2$ , and we are done.

# Longest chain – true randomness

## PROOF

continued...

Let  $X_1$  be the number of balls in the first bin.

Choose any  $k$  of the  $m$  balls (we'll pick  $k$  in a bit)

Why is  $\frac{m}{k!} \leq \frac{1}{m}$ ? (when  $k = 3 \log m$ )

So, th

$$k! = k \times (k-1) \times (k-2) \dots \times 2 \times 1$$

$$k! > 2 \times 2 \times 2 \times \dots \times 2 \times 1 = 2^{k-1}$$

By us

Let  $k = 3 \log m \dots$

$$k! > 2^{(3 \log m - 1)} \geq 2^{2 \log m} = (2^{\log m})^2 = m^2$$

Pr

$$\text{so } \frac{m}{k!} \leq \frac{m}{m^2} = \frac{1}{m}$$

$$\frac{m}{k!}.$$

- Now we set  $\kappa = 3 \log m$  and observe that  $\frac{1}{k!} \leq \frac{1}{m}$  for  $m \geq 2$ , and we are done.

# Longest chain – true randomness

## LEMMA

If  $h$  is selected uniformly at random from all functions  $U \rightarrow [m]$  then,  
over  $m$  fixed inputs,

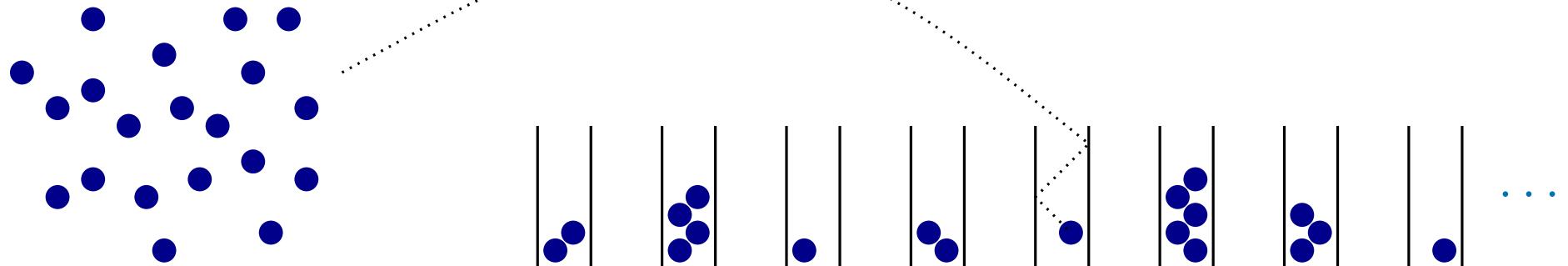
$$\Pr(\text{any chain has length } \geq 3 \log m) \leq \frac{1}{m}.$$

## OBSERVE

In this lemma we insert  $m$  keys, i.e.  $n = m$ .

## PROOF

The problem is equivalent to showing that if we randomly throw  $m$  balls into  $m$  bins, the probability of having a bin with at least  $3 \log m$  balls is at most  $\frac{1}{m}$ .



## Longest chain – weakly universal hashing

The conclusion from previous slides is that with true randomness,  
the longest chain is very short (at most  $3 \log m$ ) with high probability.

### LEMMA

If  $h$  is picked uniformly at random from a weakly universal set of hash functions then, over  $m$  fixed inputs,

$$\Pr \left( \text{any chain has length } \geq 1 + \sqrt{2m} \right) \leq \frac{1}{2}.$$

### OBSERVE

This rubbish upper bound of  $\frac{1}{2}$  does not necessarily rule out the possibility that the *tightest* upper bound is indeed very small. However, the upper bound of  $\frac{1}{2}$  is in fact tight!

## Longest chain – weakly universal hashing

### PROOF

- ▶ For any two keys  $x, y$ , let indicator r.v.  $I_{x,y}$  be 1 iff  $h(x) = h(y)$ .
- ▶ Let r.v.  $C$  be the total number of collisions:  $C = \sum_{x,y \in T, x < y} I_{x,y}$ .
- ▶ Using linearity of expectation and  $\mathbb{E}(I_{x,y}) = \frac{1}{m}$  ( $h$  is weakly universal),

$$\mathbb{E}(C) = \mathbb{E}\left(\sum_{x,y \in T, x < y} I_{x,y}\right) = \sum_{x,y \in T, x < y} \mathbb{E}(I_{x,y}) = \binom{m}{2} \cdot \frac{1}{m} \leq \frac{m}{2}.$$

- ▶ by Markov's inequality,  $\Pr(C \geq m) \leq \frac{\mathbb{E}(C)}{m} \leq \frac{1}{2}$ .
- ▶ Let r.v.  $L$  be the length of the longest chain. Then  $C \geq \binom{L}{2}$ .

This is because a chain of length  $L$  causes  $\binom{L}{2}$  collisions!

## Longest chain – weakly universal hashing

### PROOF

- ▶ For any two keys  $x, y$ , let indicator r.v.  $I_{x,y}$  be 1 iff  $h(x) = h(y)$ .
- ▶ Let r.v.  $C$  be the total number of collisions:  $C = \sum_{x,y \in T, x < y} I_{x,y}$ .
- ▶ Using linearity of expectation and  $\mathbb{E}(I_{x,y}) = \frac{1}{m}$  ( $h$  is weakly universal),

$$\mathbb{E}(C) = \mathbb{E}\left(\sum_{x,y \in T, x < y} I_{x,y}\right) = \sum_{x,y \in T, x < y} \mathbb{E}(I_{x,y}) = \binom{m}{2} \cdot \frac{1}{m} \leq \frac{m}{2}.$$

- ▶ by Markov's inequality,  $\Pr(C \geq m) \leq \frac{\mathbb{E}(C)}{m} \leq \frac{1}{2}$ .
- ▶ Let r.v.  $L$  be the length of the longest chain. Then  $C \geq \binom{L}{2}$ .
- ▶ Now,  $\Pr\left(\frac{(L-1)^2}{2} \geq m\right) \leq \Pr\left(\binom{L}{2} \geq m\right) \leq \Pr(C \geq m) \leq \frac{1}{2}$ .

↑

this is because  $\binom{L}{2} = \frac{L!}{2!(L-2)!} = \frac{L \cdot (L-1)}{2} \geq \frac{(L-1)^2}{2}$

## Longest chain – weakly universal hashing

### PROOF

- ▶ For any two keys  $x, y$ , let indicator r.v.  $I_{x,y}$  be 1 iff  $h(x) = h(y)$ .
- ▶ Let r.v.  $C$  be the total number of collisions:  $C = \sum_{x,y \in T, x < y} I_{x,y}$ .
- ▶ Using linearity of expectation and  $\mathbb{E}(I_{x,y}) = \frac{1}{m}$  ( $h$  is weakly universal),

$$\mathbb{E}(C) = \mathbb{E}\left(\sum_{x,y \in T, x < y} I_{x,y}\right) = \sum_{x,y \in T, x < y} \mathbb{E}(I_{x,y}) = \binom{m}{2} \cdot \frac{1}{m} \leq \frac{m}{2}.$$

- ▶ by Markov's inequality,  $\Pr(C \geq m) \leq \frac{\mathbb{E}(C)}{m} \leq \frac{1}{2}$ .
- ▶ Let r.v.  $L$  be the length of the longest chain. Then  $C \geq \binom{L}{2}$ .
- ▶ Now,  $\Pr\left(\frac{(L-1)^2}{2} \geq m\right) \leq \Pr\left(\binom{L}{2} \geq m\right) \leq \Pr(C \geq m) \leq \frac{1}{2}$ .

By rearranging, we have that  $\Pr\left(L \geq 1 + \sqrt{2m}\right) \leq \frac{1}{2}$ , and we are done.

## Conclusions

For both,

**true randomness** ( $h$  is picked uniformly from the set of all possible hash functions)

and **weakly universal hashing**

( $h$  is picked uniformly from a weakly universal set of hash functions)

we have seen that when  $m \geq n$ ,

the expected lookup time in a hash table with chaining is  $O(1)$ .

### LEMMA

If  $h$  is selected uniformly at random from all functions  $U \rightarrow [m]$  then,

$$\Pr(\text{any chain has length } \geq 3 \log m) \leq \frac{1}{m}.$$

### LEMMA

If  $h$  is picked uniformly at random from a weakly universal set of hash functions,

P

(both Lemmas hold for  $m$  any fixed inputs)