Data Structures and Algorithms
Hashing

CS 225
G Carl Evans

April 17, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Slides by Brad Solomon

Learning Objectives

Motivate and formally define a hash table

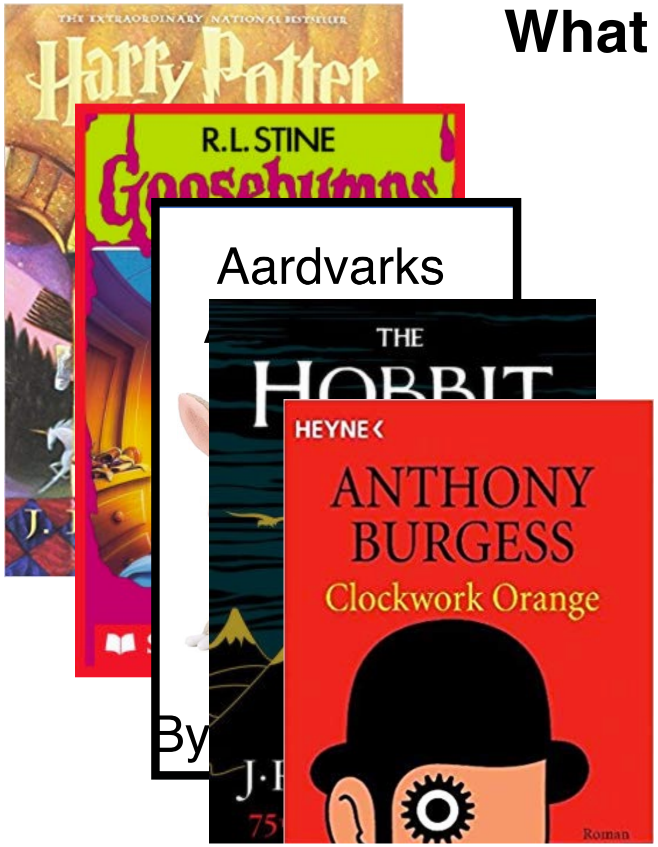
Discuss what a ‘good’ hash function looks like

Identify the key weakness of a hash table

Introduce strategies to “correct” this weakness

Data Structure Review

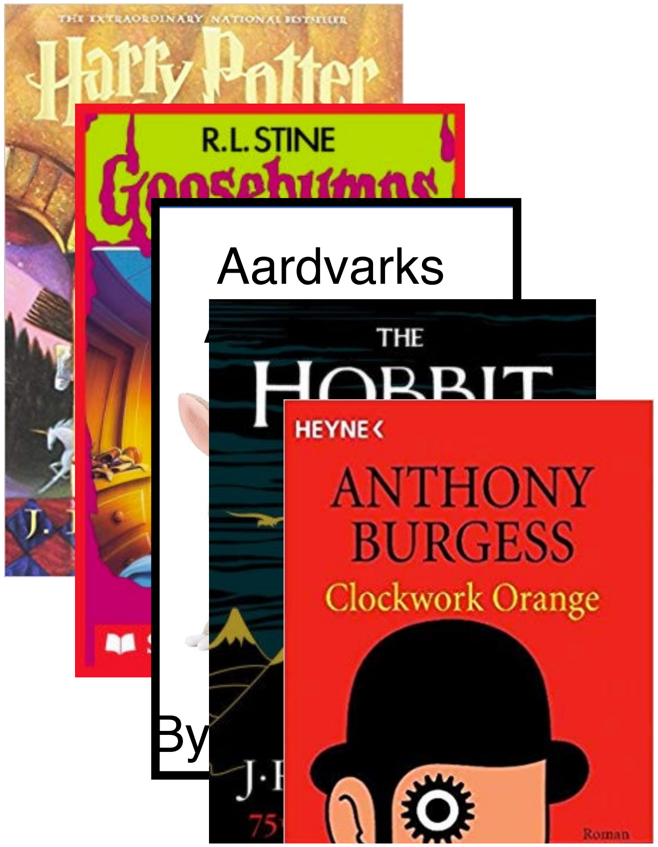
I have a collection of books and I want to store them in a dictionary!



What data structures can I use here?

Data Structure Review

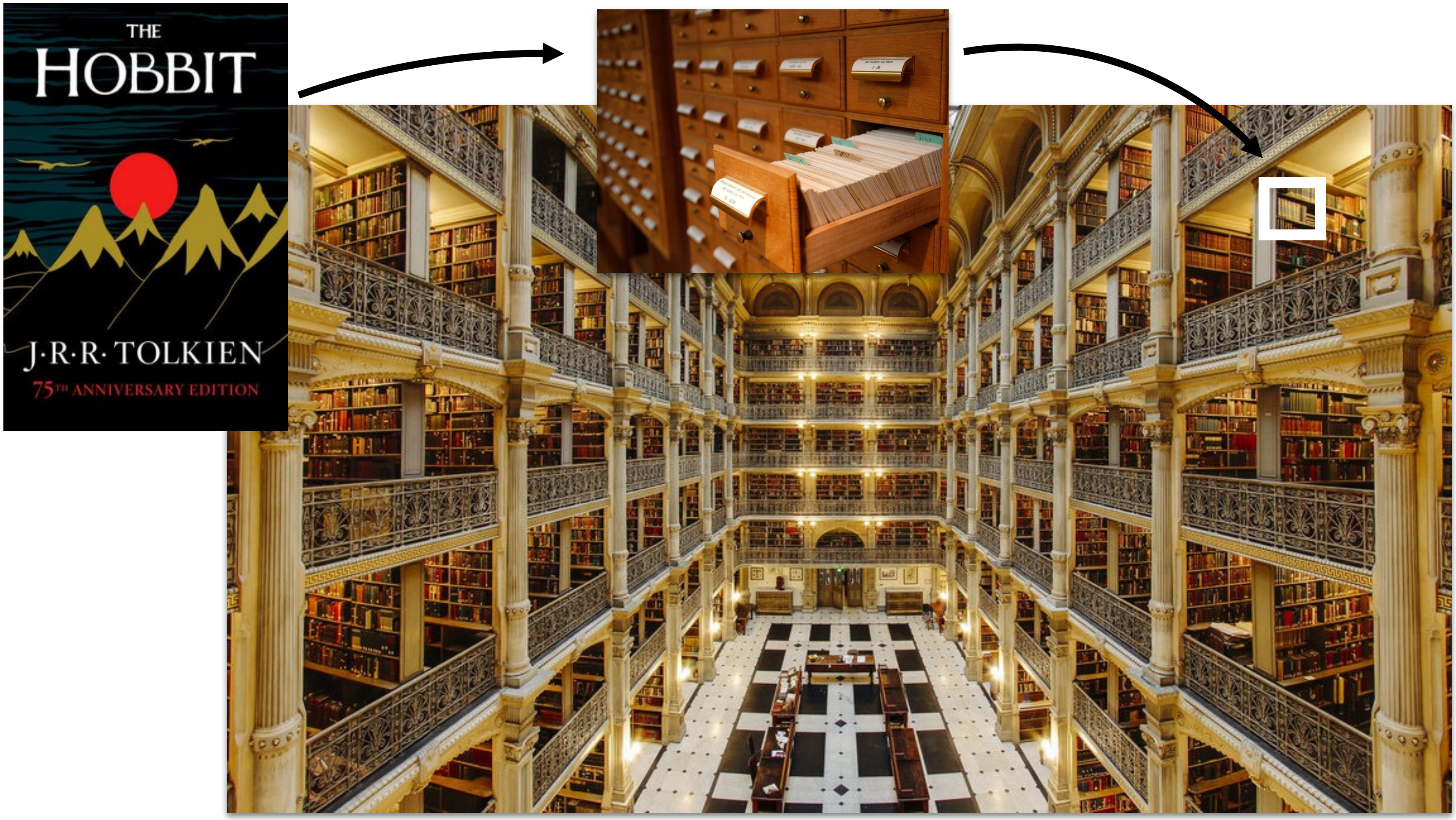
I have a collection of books and I want to store them in a dictionary!



	Sorted Array	BST	AVL Tree
Find			
Insert			
Remove			

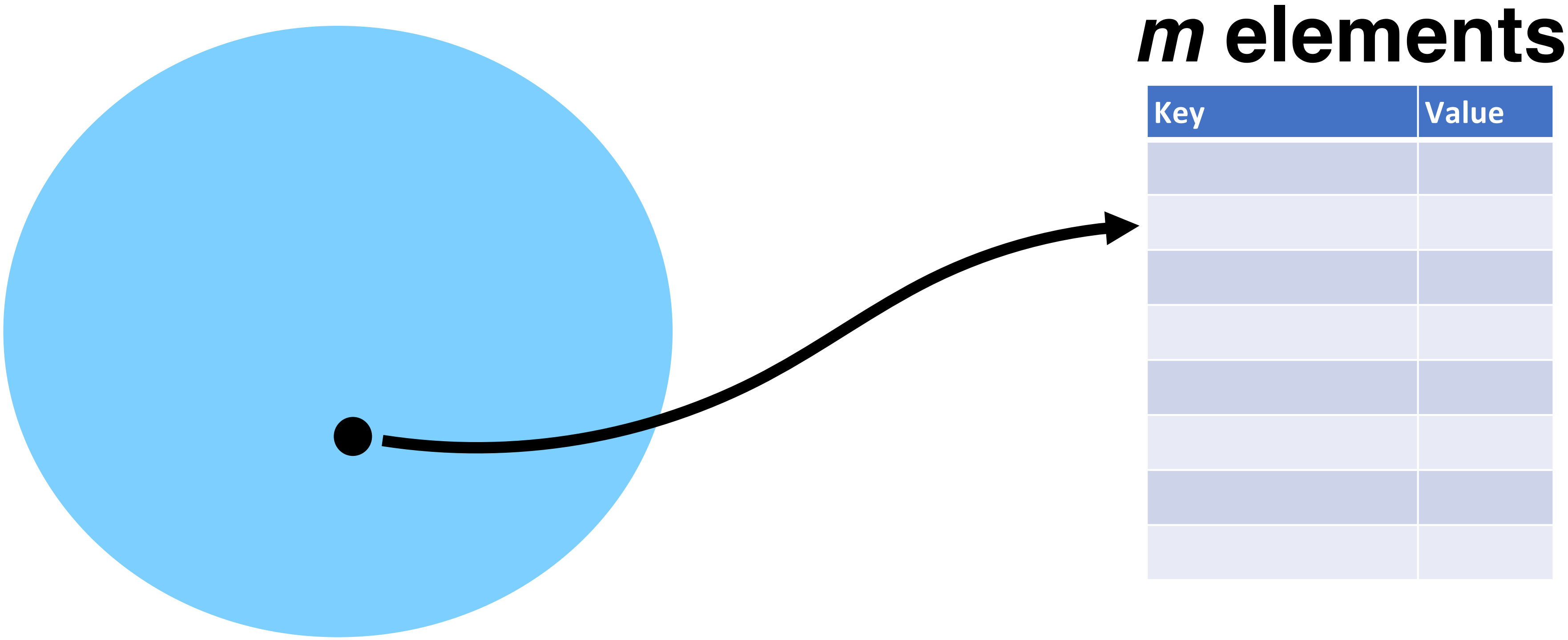
What if $O(log\ n)$ isn't good enough?

What if $O(log\ n)$ isn’t good enough?



Hash Function

Maps a **keyspace**, a (mathematical) description of the keys for a set of data, to a set of integers.



***m* elements**

Key	Value

Hash Function

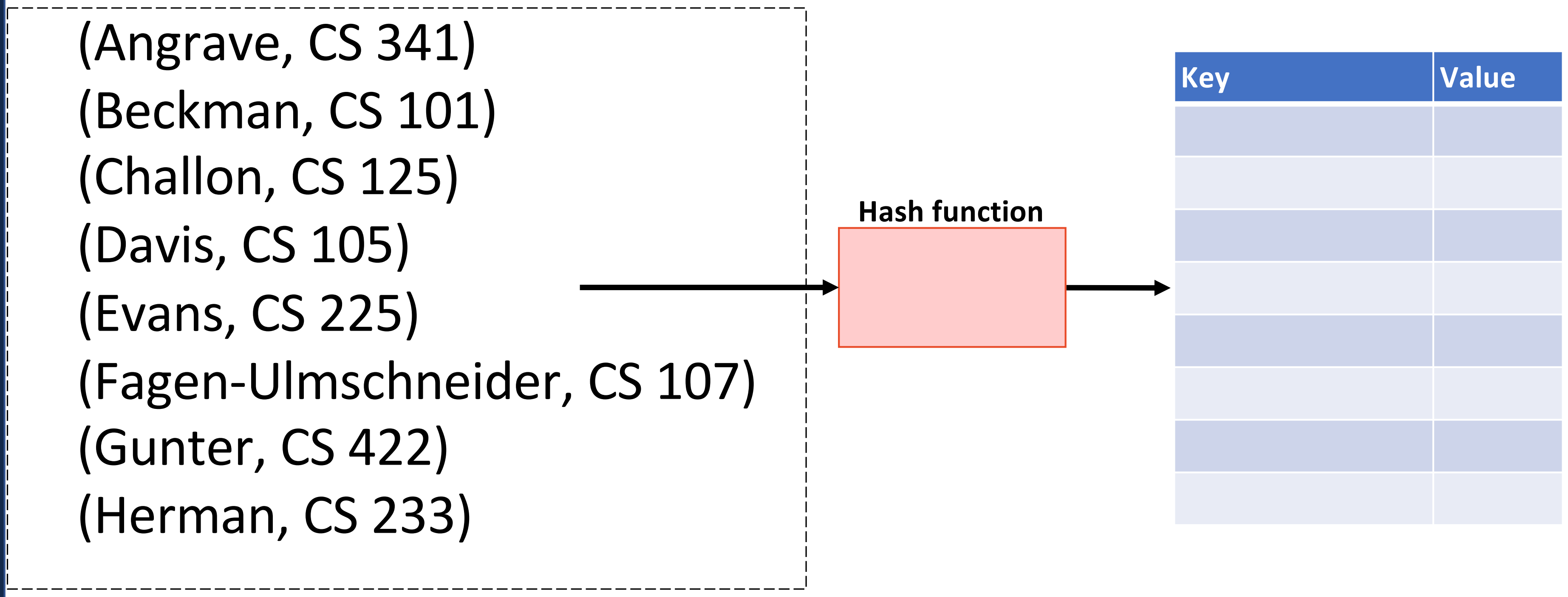
A hash function ***must*** be:

- **Deterministic**:

- **Efficient**:

- **Defined for a certain size table:**

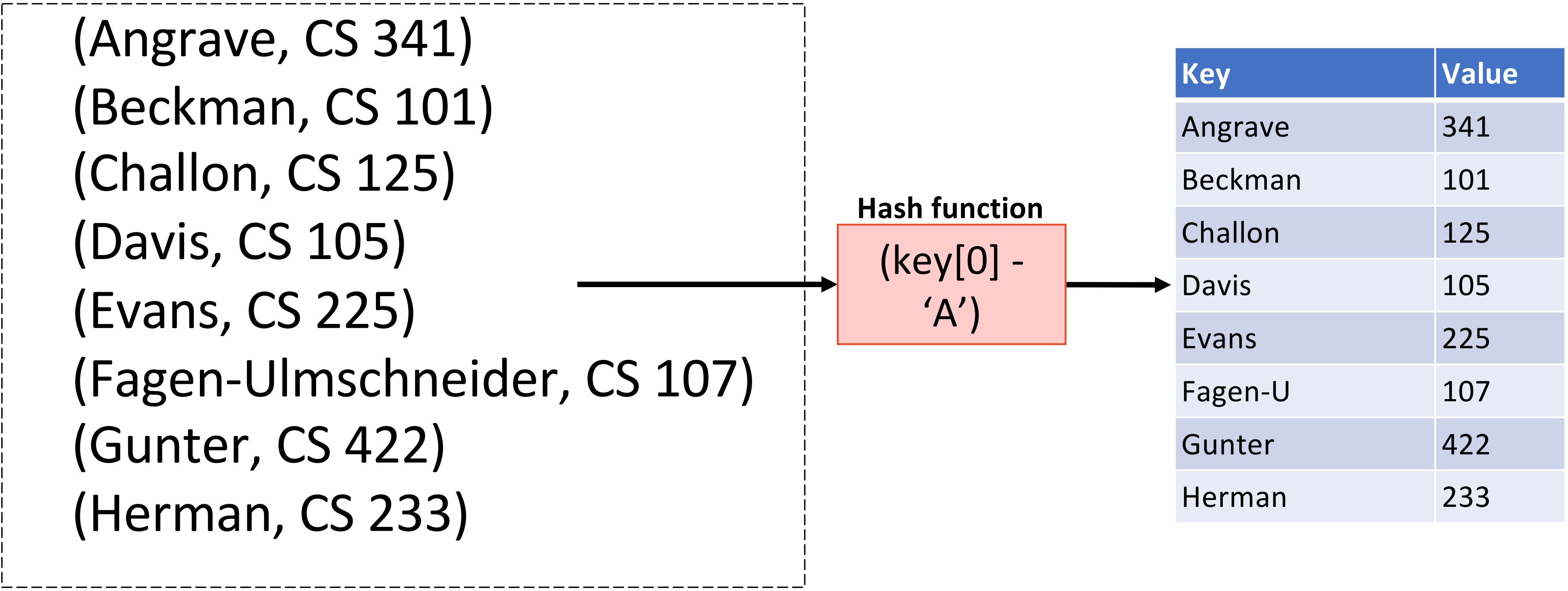
Hash Function



(Angrave, CS 341)
(Beckman, CS 101)
(Challon, CS 125)
(Davis, CS 105)
(Evans, CS 225)
(Fagen-Ulmschneider, CS 107)
(Gunter, CS 422)
(Herman, CS 233)

Key	Value

Hash Function



(Angrave, CS 341)
(Beckman, CS 101)
(Challon, CS 125)
(Davis, CS 105)
(Evans, CS 225)
(Fagen-Ulmschneider, CS 107)
(Gunter, CS 422)
(Herman, CS 233)

Hash function

(key[0] - ‘A’)

Key	Value
Angrave	341
Beckman	101
Challon	125
Davis	105
Evans	225
Fagen-U	107
Gunter	422
Herman	233

General Hash Function

An $O(1)$ deterministic operation that maps all keys in a universe U to a defined range of integers $[0, \ldots, m-1]$

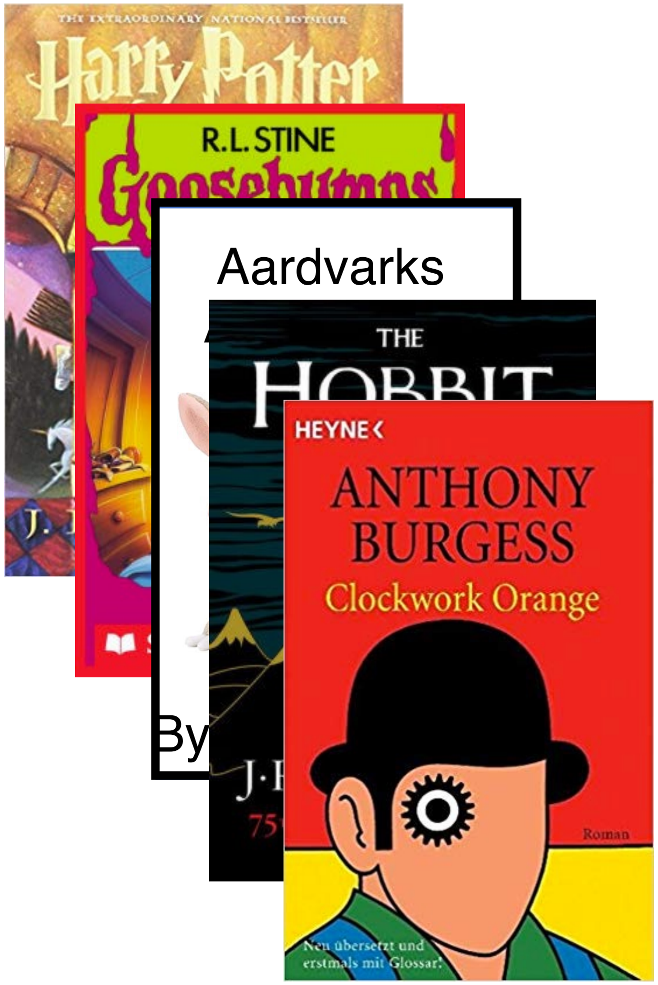
- A **hash**:

- A **compression**:

Choosing a good hash function is tricky…

- Don’t create your own (yet*)

Hash Function

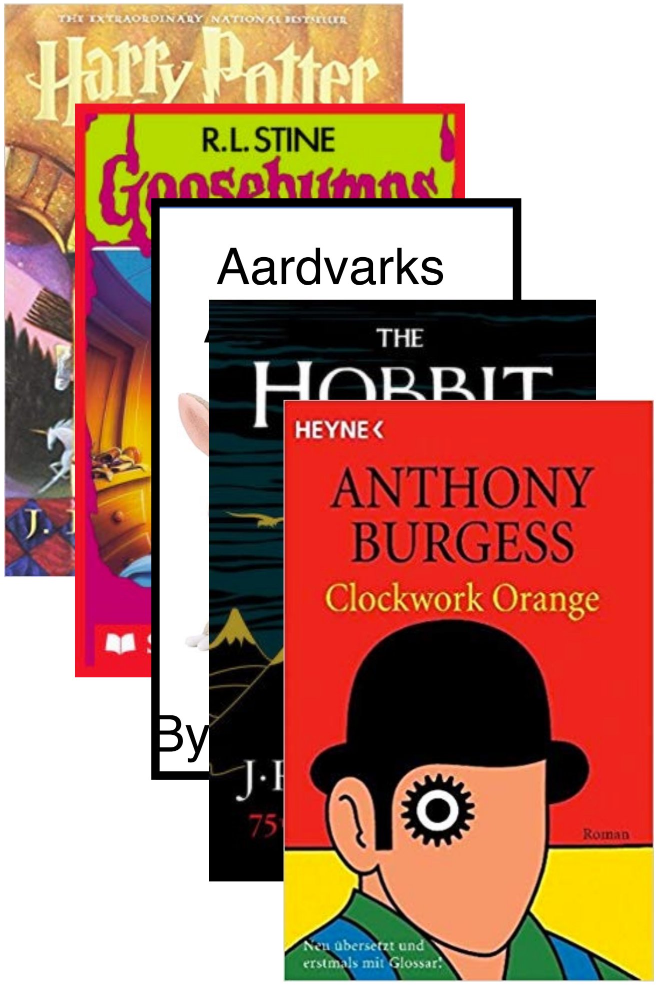


$$h(k) = (k.firstName[0] + k.lastName[0])\ \%\ m$$

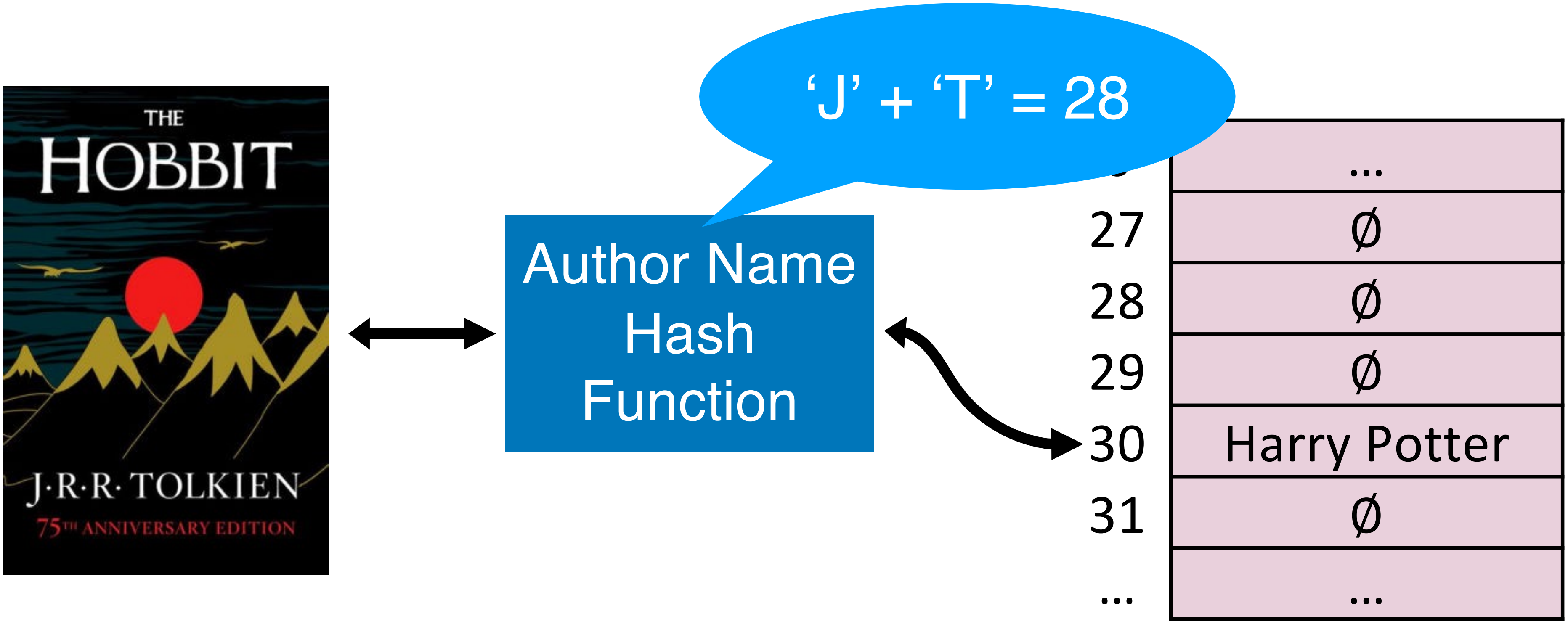
$$h(k) = (rand() * k.numPages)\ \%\ m$$

$$h(k) = (k.order\_1st\_read\_by\_me)\ \%\ m$$

Hash Function



Hash Function

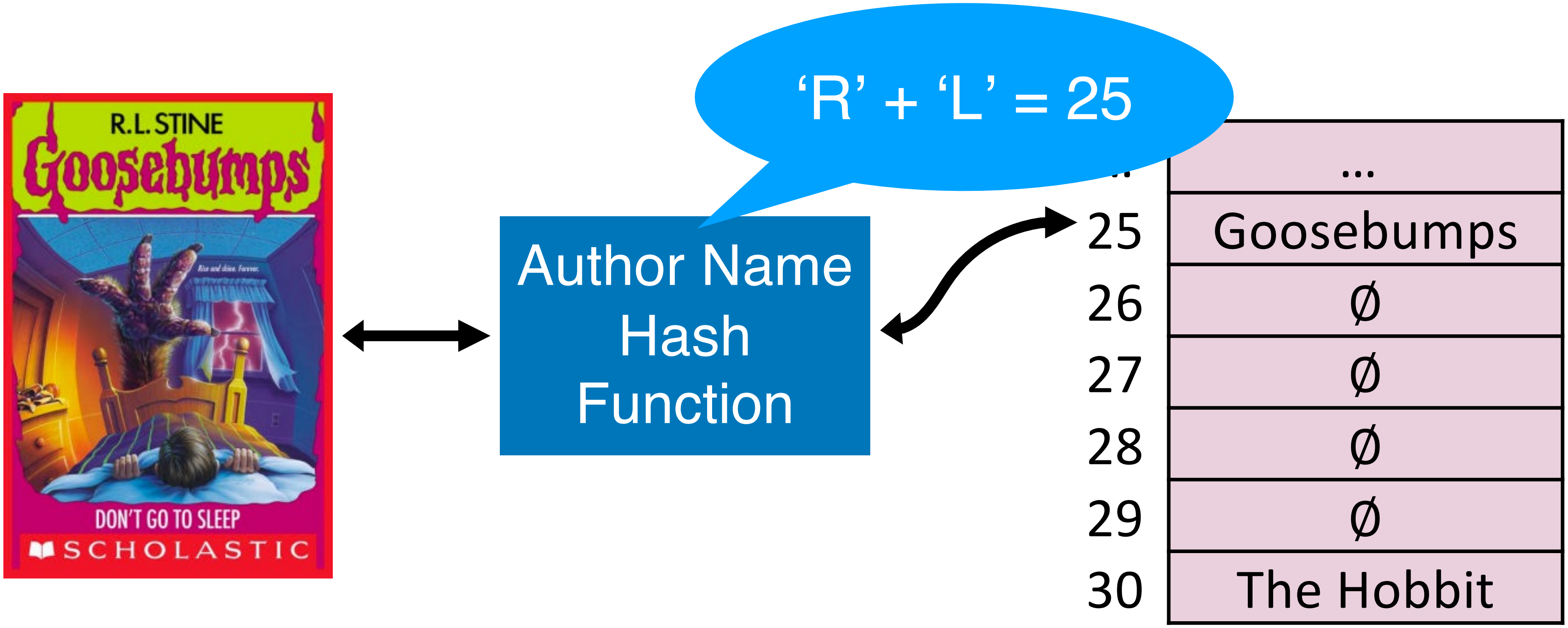


‘J’ + ‘T’ = 28

Author Name Hash Function

	…
27	Ø
28	Ø
29	Ø
30	Harry Potter
31	Ø
…	…

Hash Function

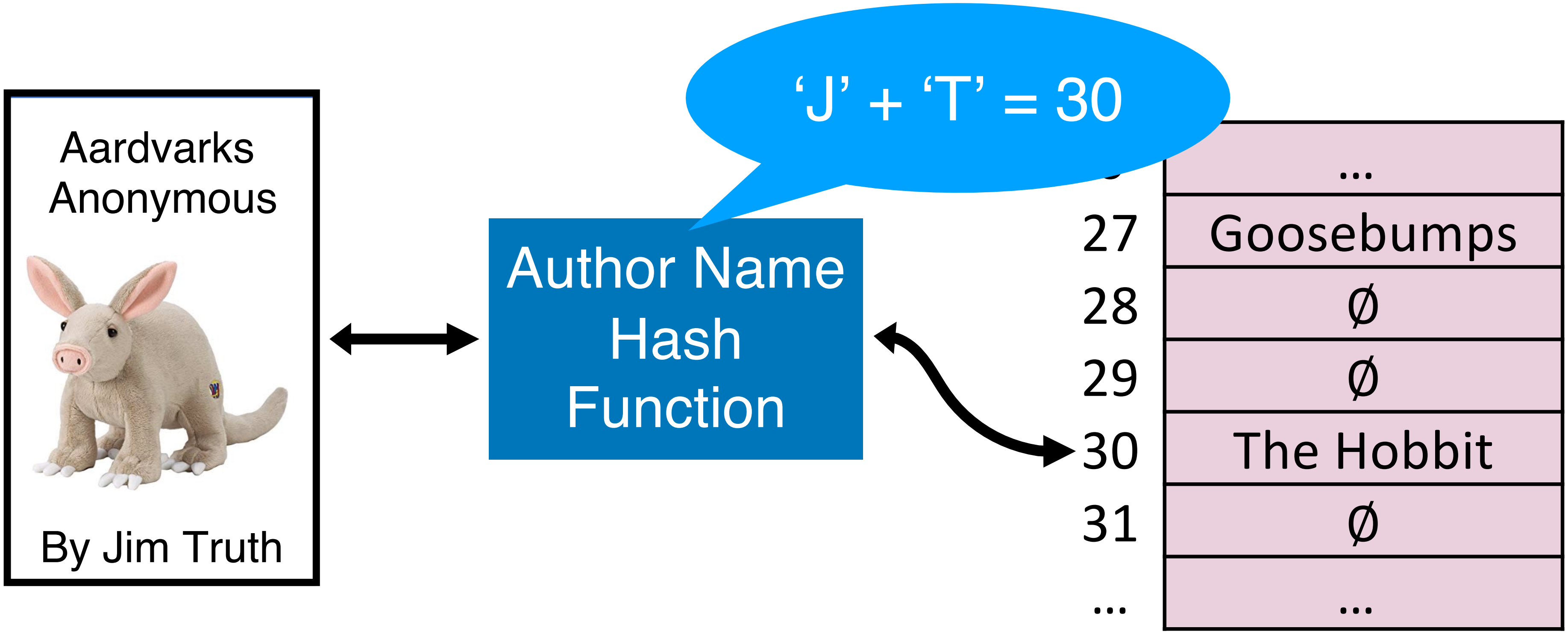


‘R’ + ‘L’ = 25

Author Name Hash Function

…	…
25	Goosebumps
26	∅
27	∅
28	∅
29	∅
30	The Hobbit

Hash Function



Aardvarks Anonymous

By Jim Truth

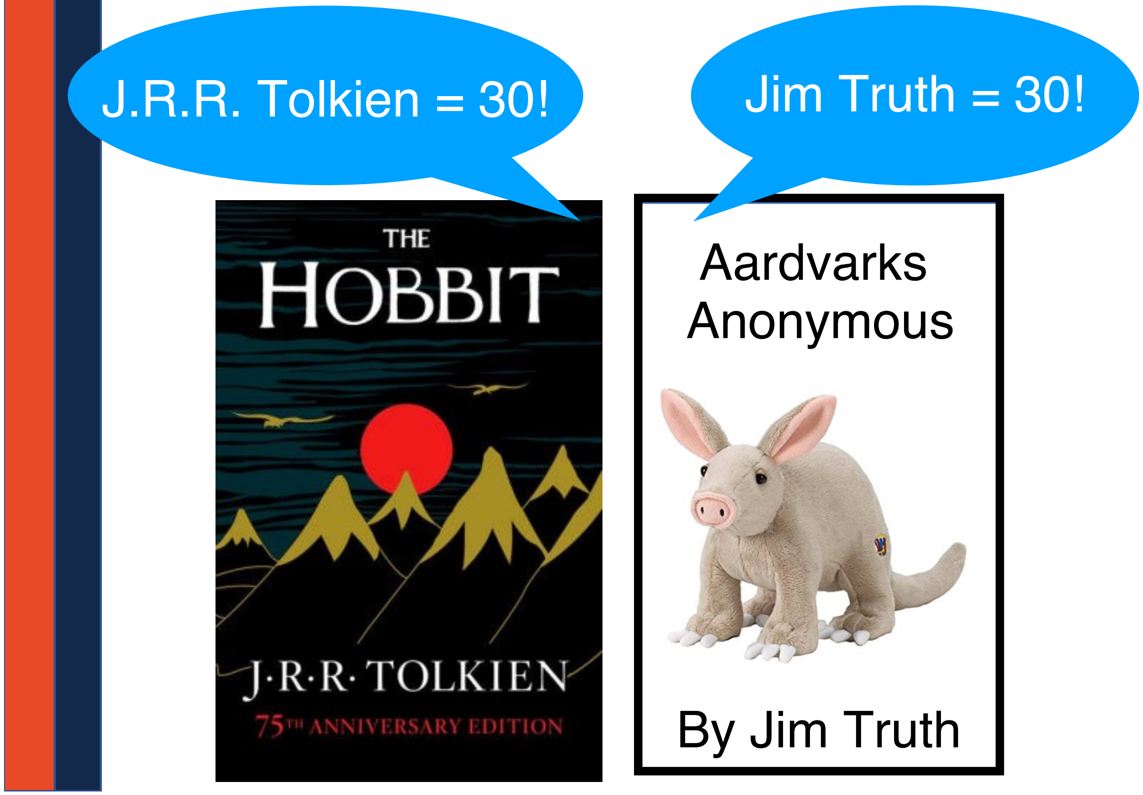
Author Name Hash Function

‘J’ + ‘T’ = 30

	…
27	Goosebumps
28	Ø
29	Ø
30	The Hobbit
31	Ø
…	…

Hash Collision

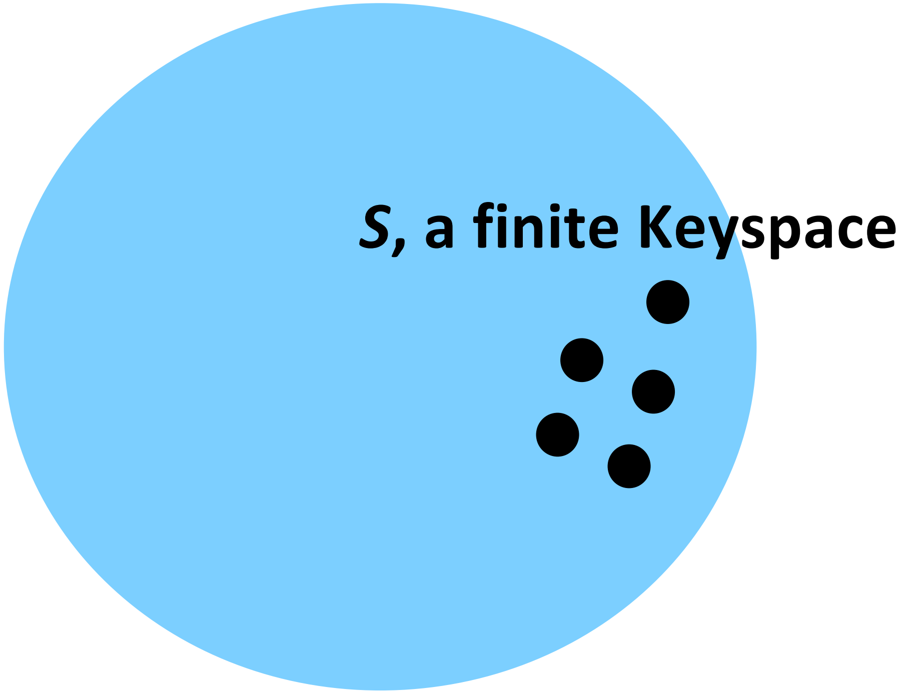
A ***hash collision*** occurs when multiple unique keys hash to the same value



…	…
27	Ø
28	Ø
29	Ø
30	???
31	Ø
…	…

Perfect Hashing

If $m \geq S$, we can write a *perfect* hash with no collisions

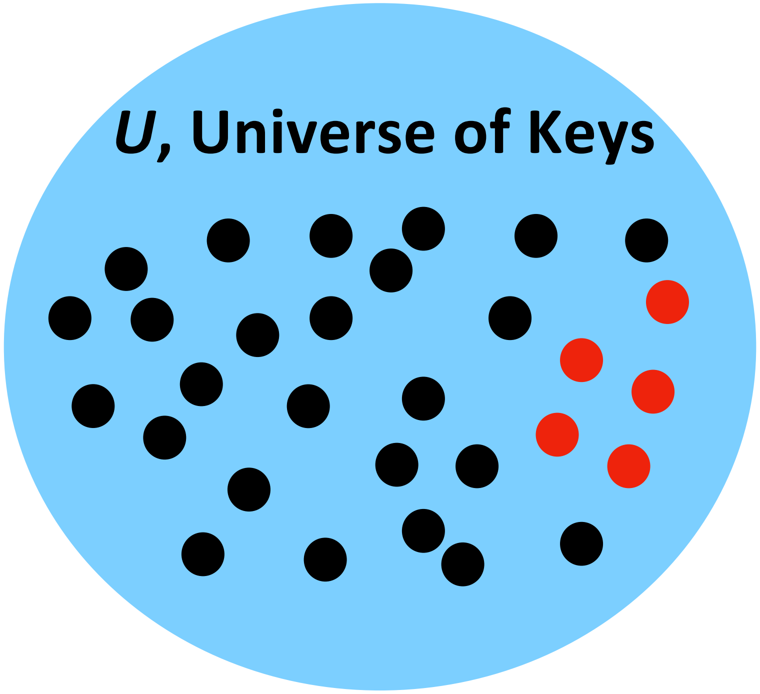


***m* elements**

Key	Value

General Purpose Hashing

In CS 225, we want our hash functions to work *in general*.

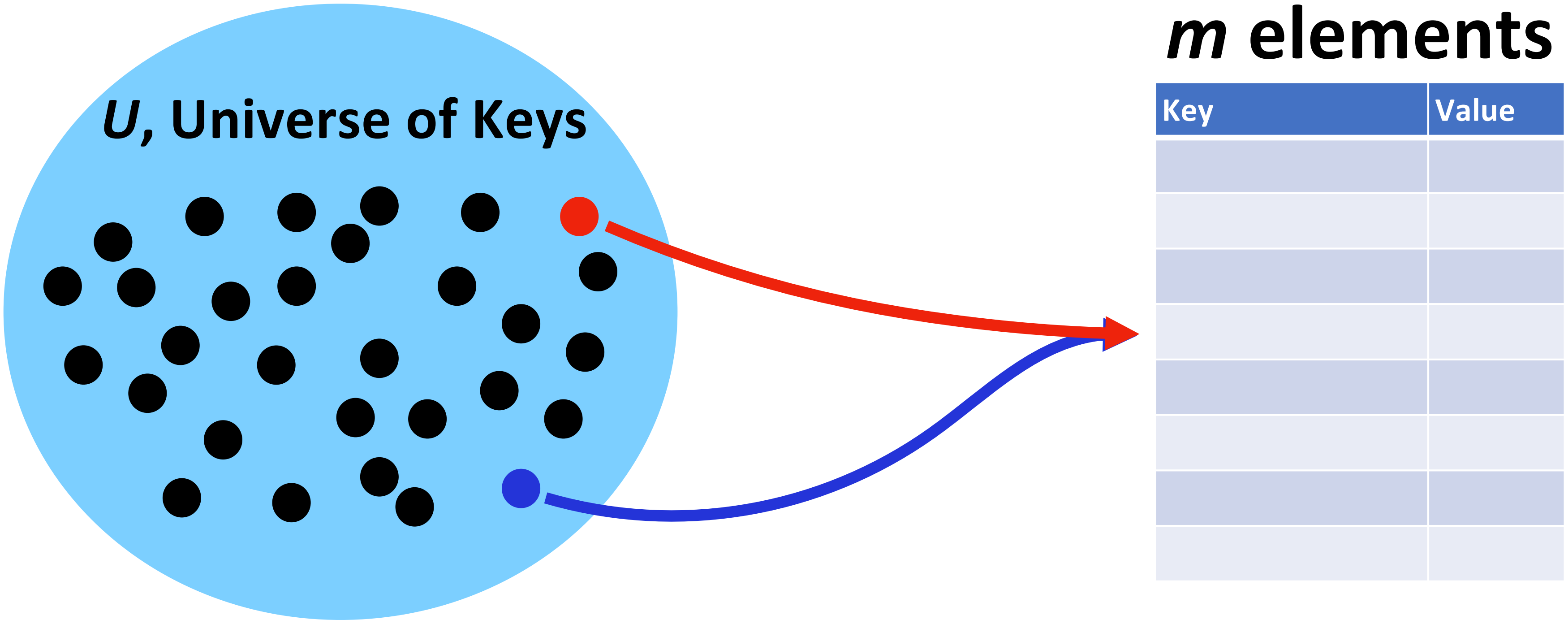


***m* elements**

Key	Value

General Purpose Hashing

If $m < U$, there must be at least one hash collision.



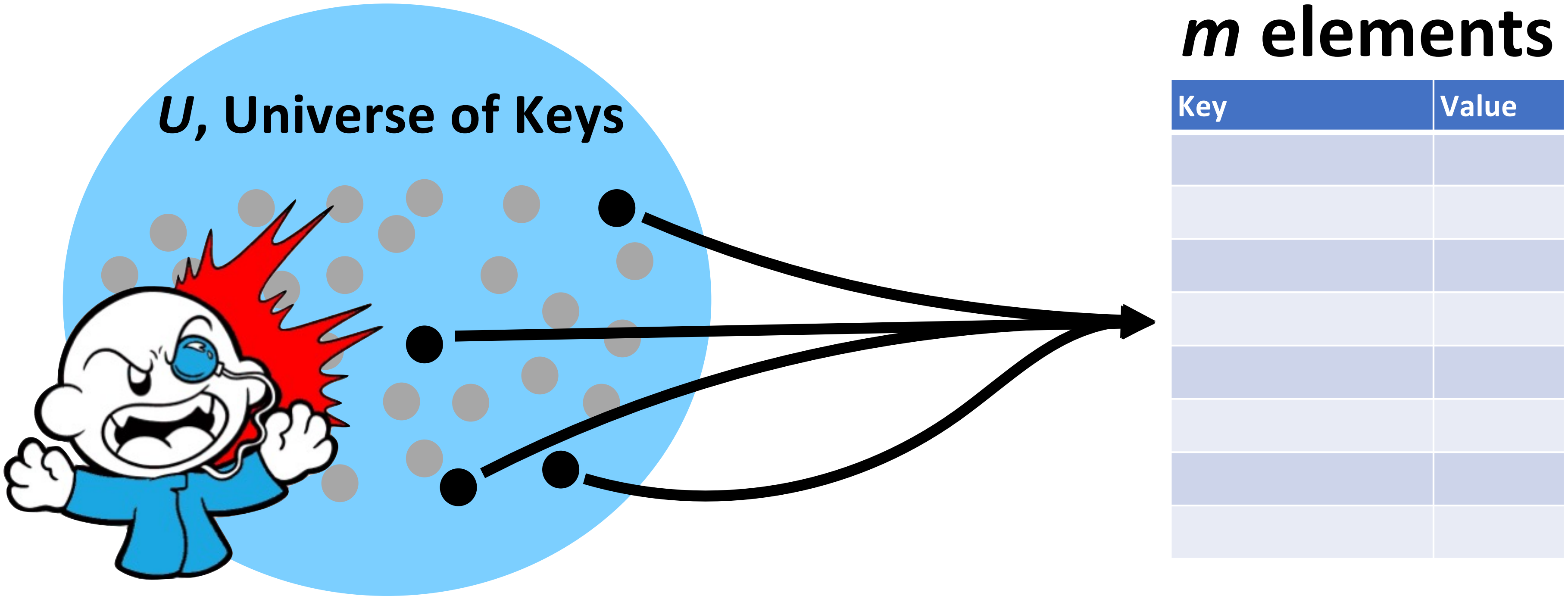
***U*, Universe of Keys**

***m* elements**

Key	Value

General Purpose Hashing

By fixing h, we open ourselves up to adversarial attacks.



Key	Value

Image by Matthew Loffhagen

A Hash Table based Dictionary

Client Code:

```
Dictionary<KeyType, ValueType> d;
d[k] = v;
```

A **Hash Table** consists of three things:

1. A hash function

2. A data storage structure

3. **A method of addressing *hash collisions***