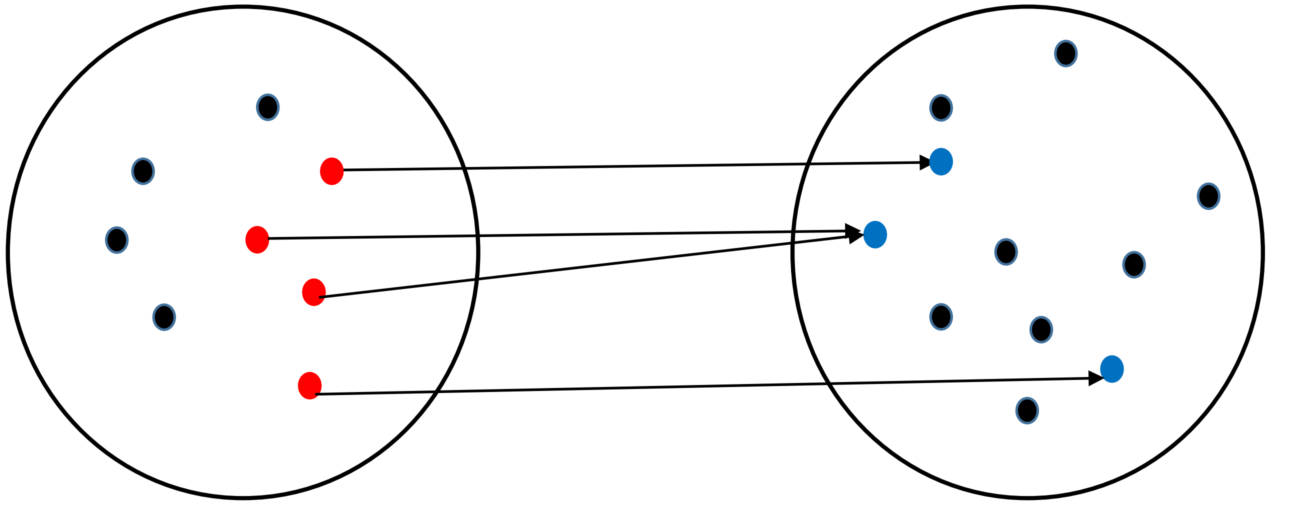
COMP 250

Lecture 29

hashing

Nov. 16, 2018

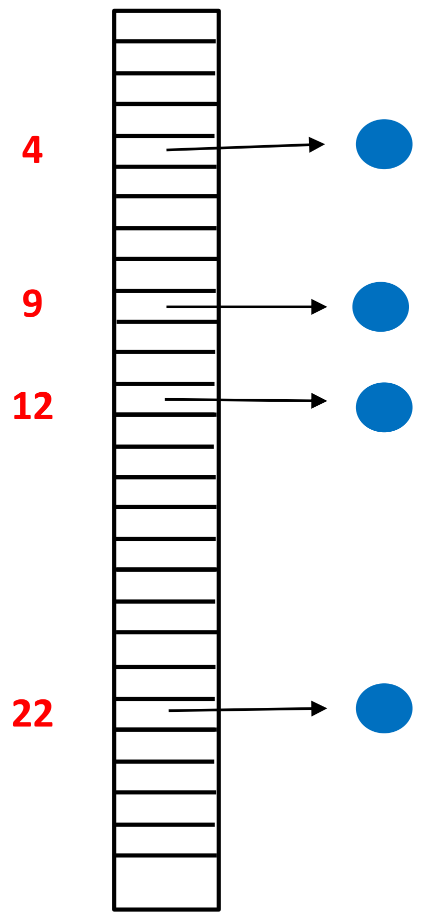
RECALL: Map



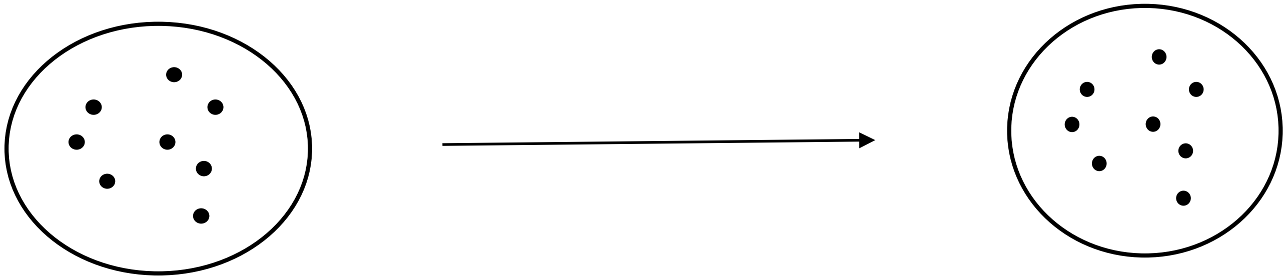
keys (type K) values (type V)

Each (key, value) pairs is an “entry”.
For each key, there is at most one value.

RECALL Special Case: keys are unique positive integers in small range



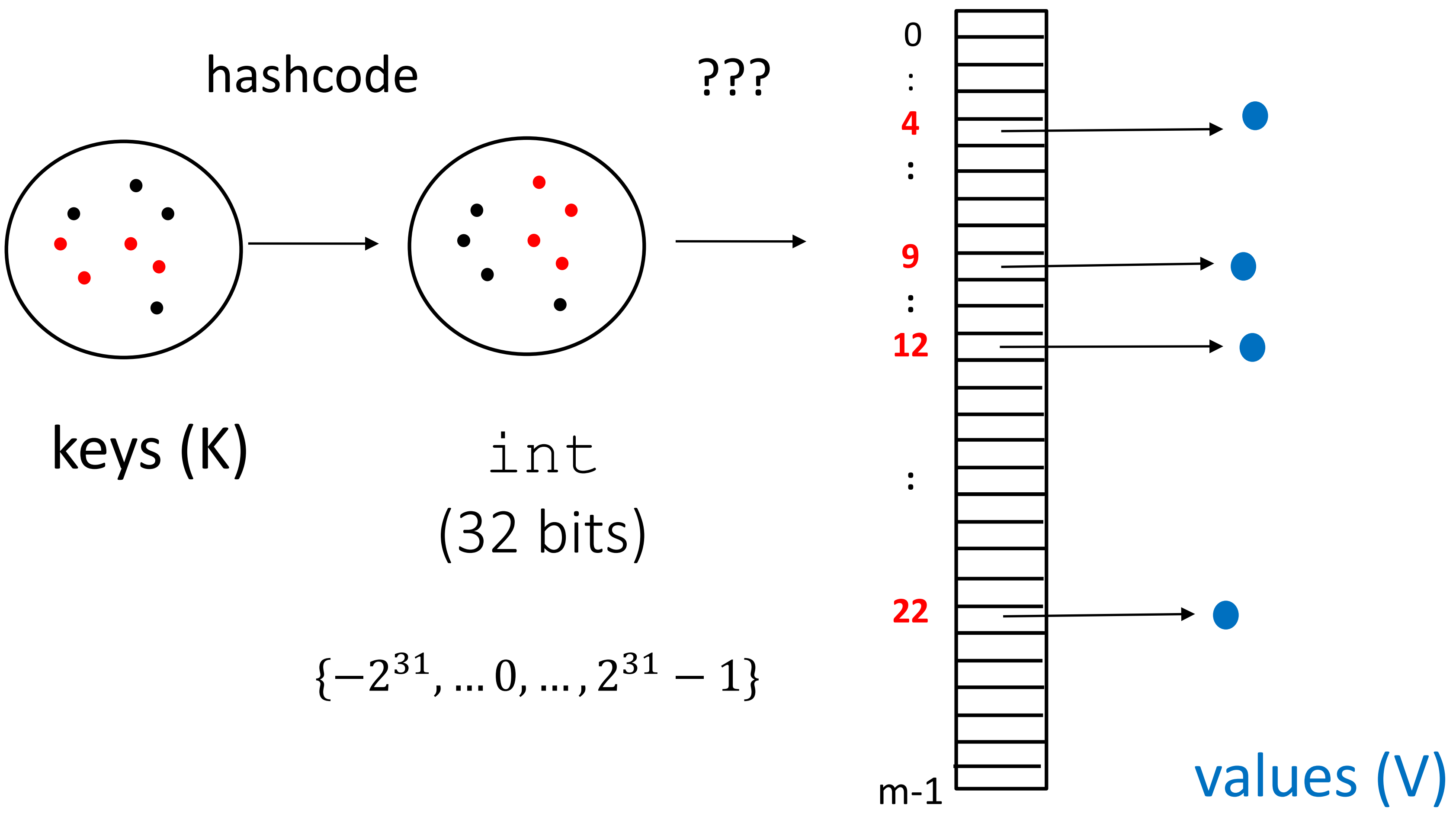
RECALL: Java hashcode()

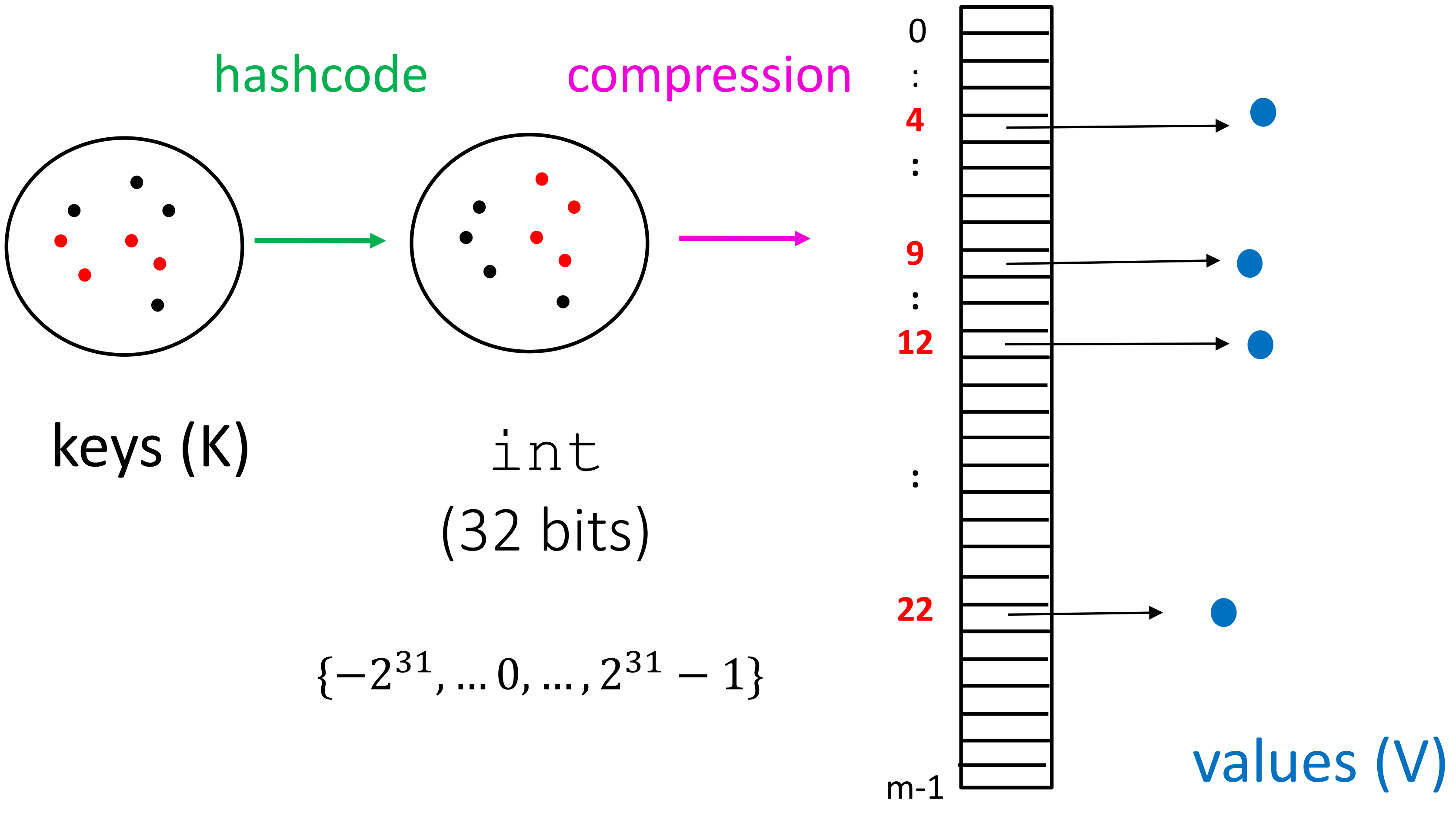


keys K

`int`
(32 bits)

Map Composition

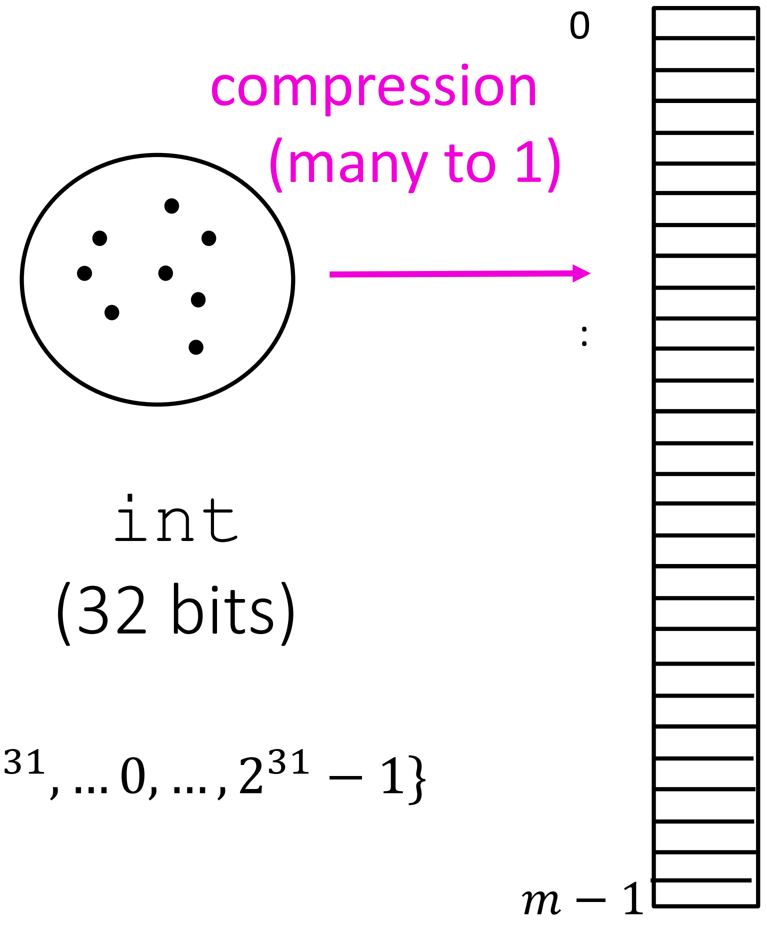




$\{-2^{31}, \dots 0, \dots, 2^{31} - 1\}$

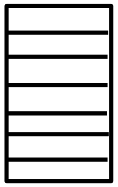
compression: $i \rightarrow |i| \; mod \; m,$

where m is the length of the array.



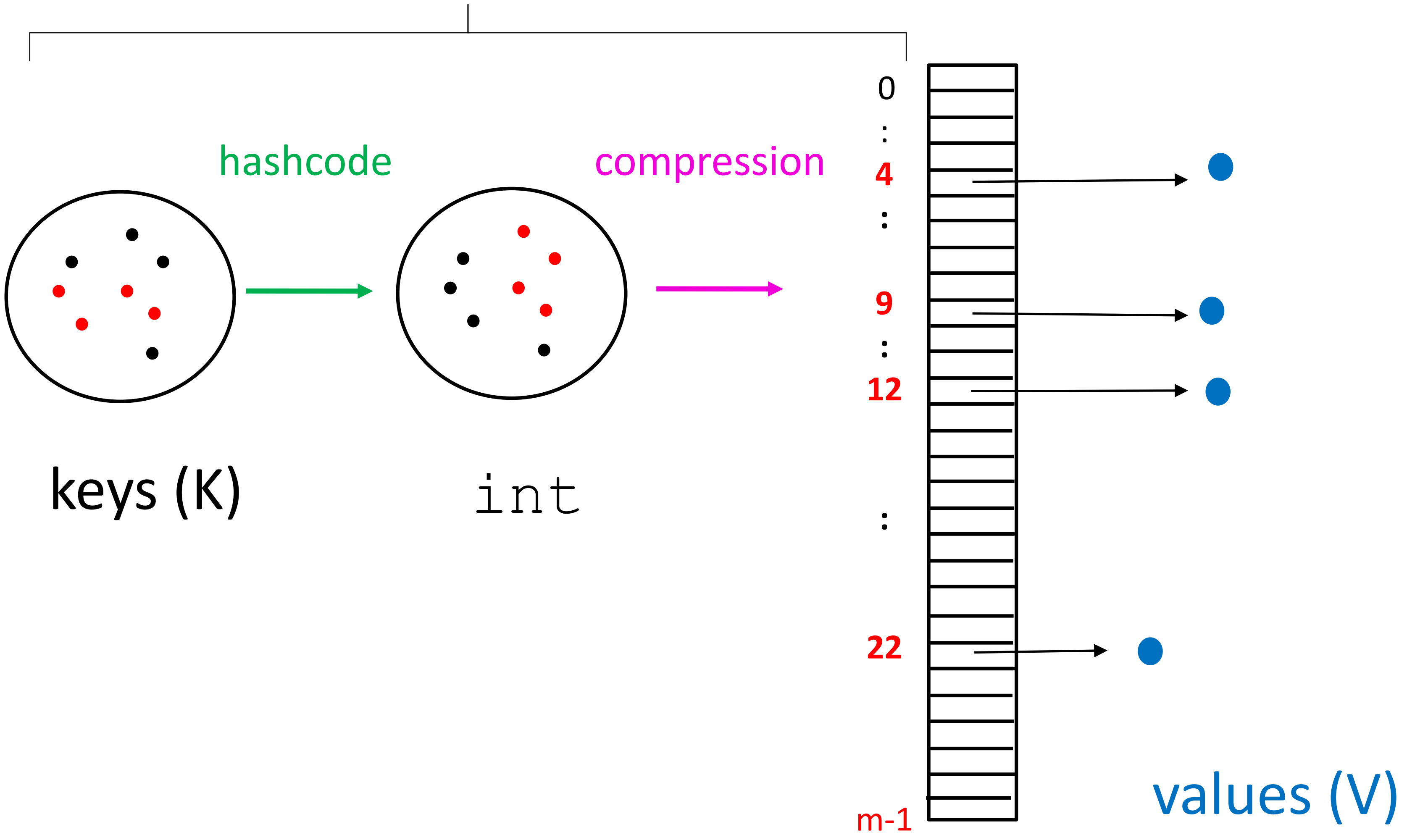
compression map: $i \rightarrow |i| \ mod \ m,$

hash code	hash value (hashcode mod 7)
41	6
16	2
25	4
21	0
36	1
35	0
53	4



“hash function” $\equiv$ compression $\circ$ hashCode "hash values"

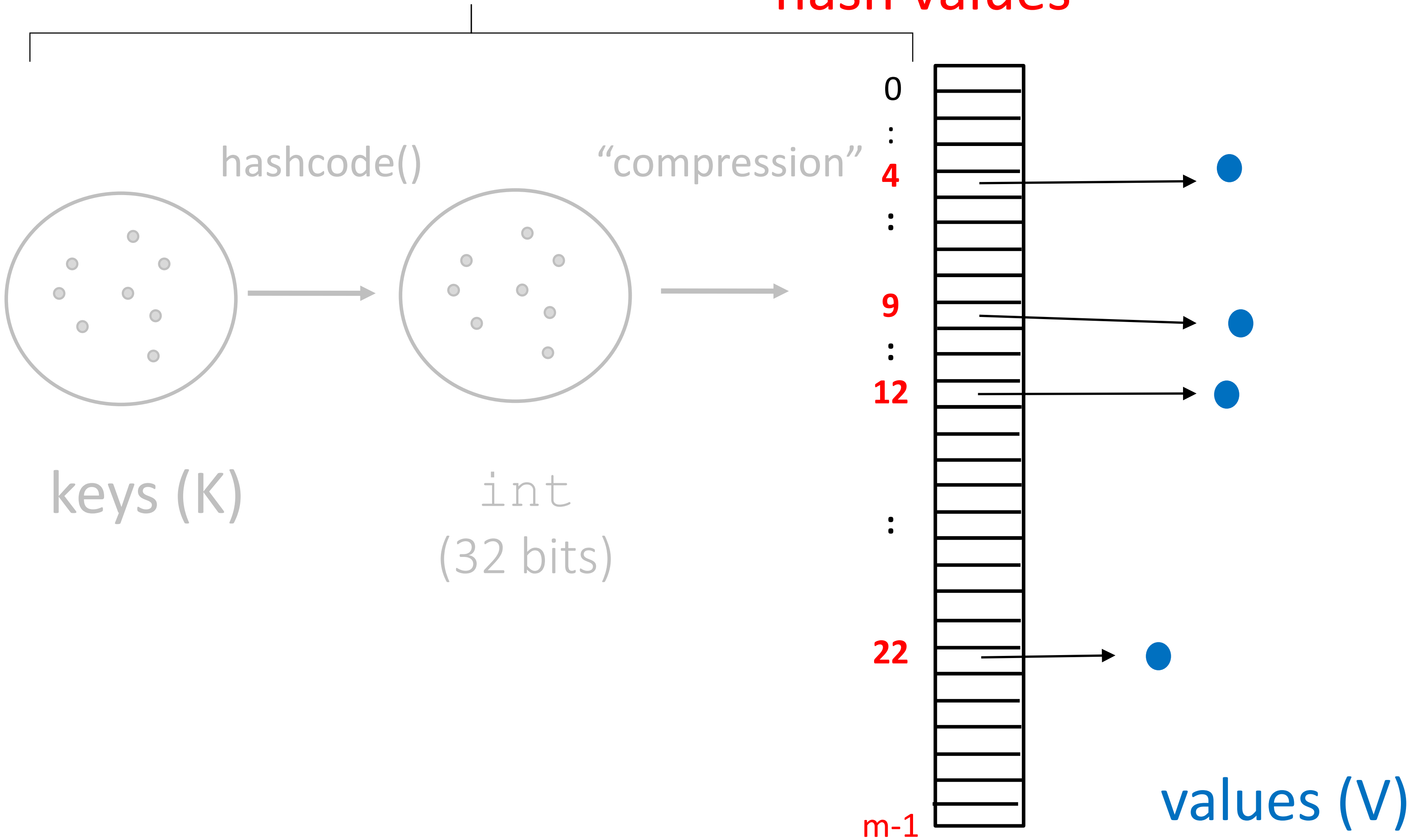
"hash function" : keys → {0, …, m-1}



Heads up! “Values” is used in two ways.

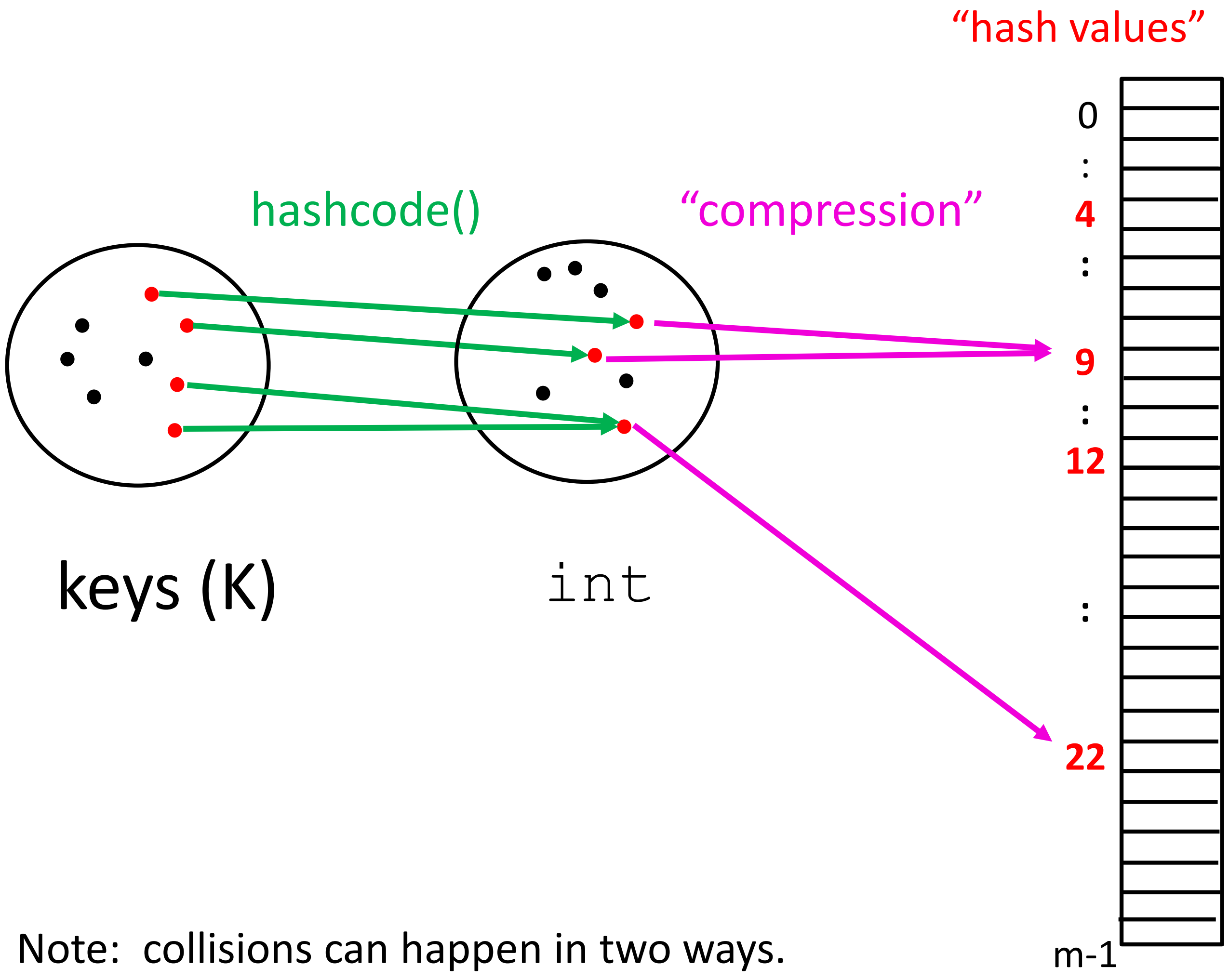
hash function : keys → {0, …, m-1}

“hash values”



Collision:

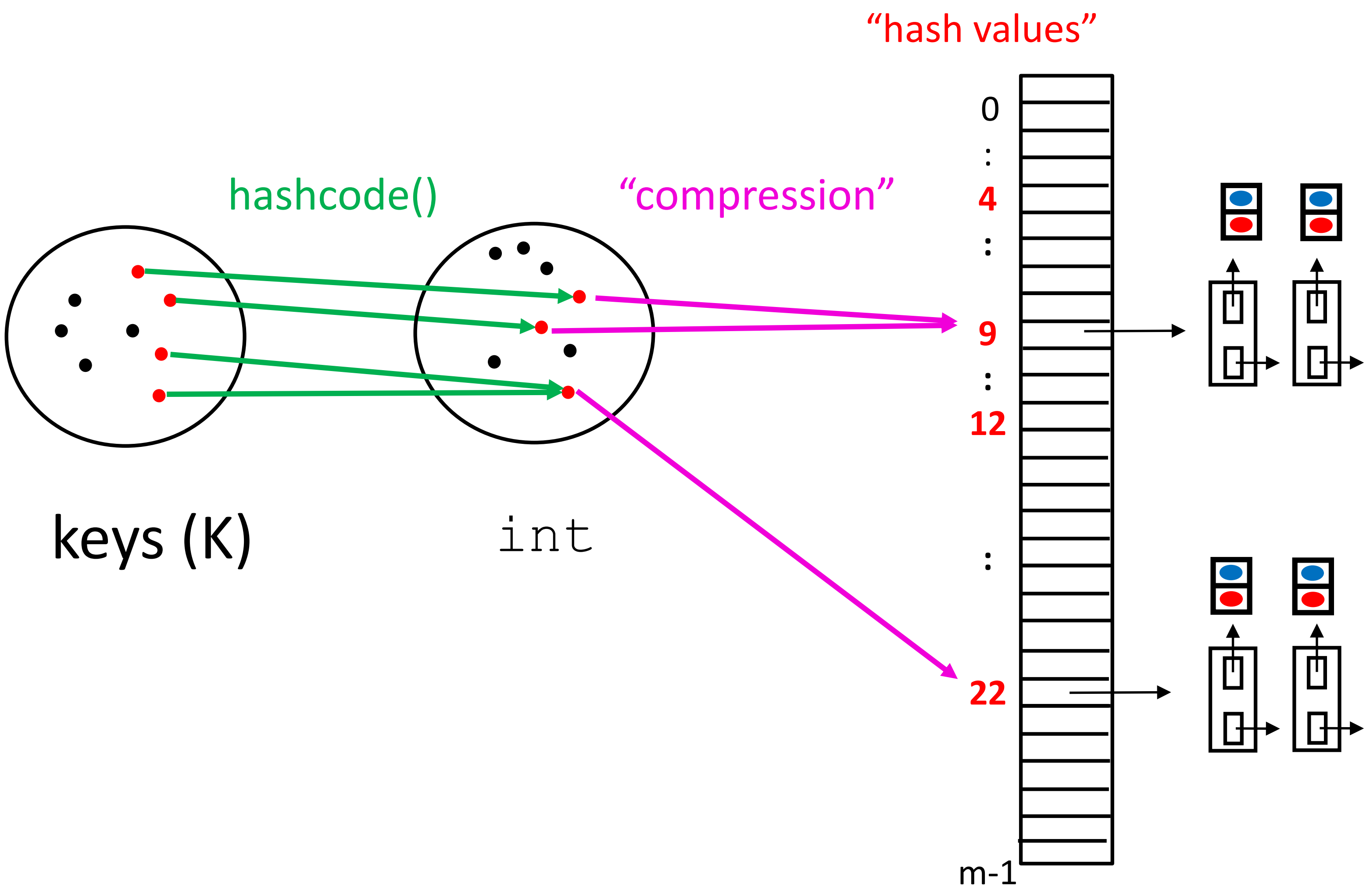
when two or more keys k map to the same hash value.



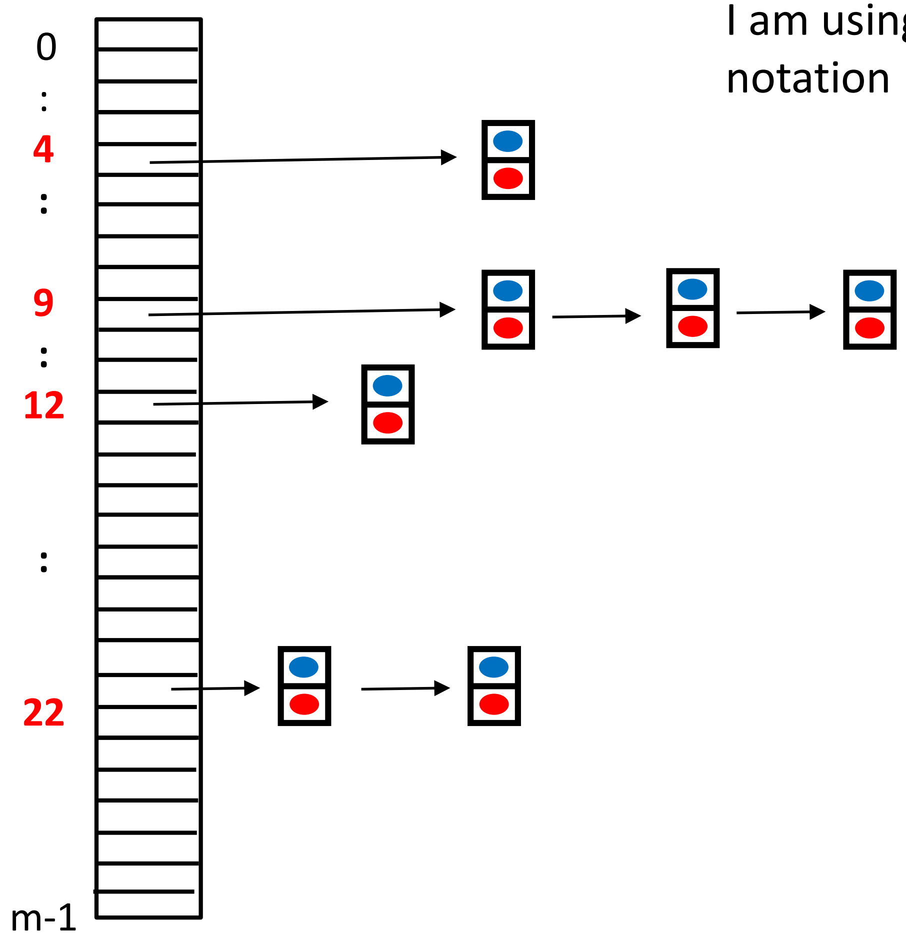
Note: collisions can happen in two ways.

Solution: Hash Table (or "Hash Map"):

Each array slot holds a singly linked list of entries.



Each array slot + linked list is called a **bucket**. So there are m buckets.



I am using a simpler linked list notation here.

Why is it necessary to store (key, value) pairs in the linked list?

Why not just the values?

Why is it necessary to store (key, value) pairs in the linked list?

Why not just the values?

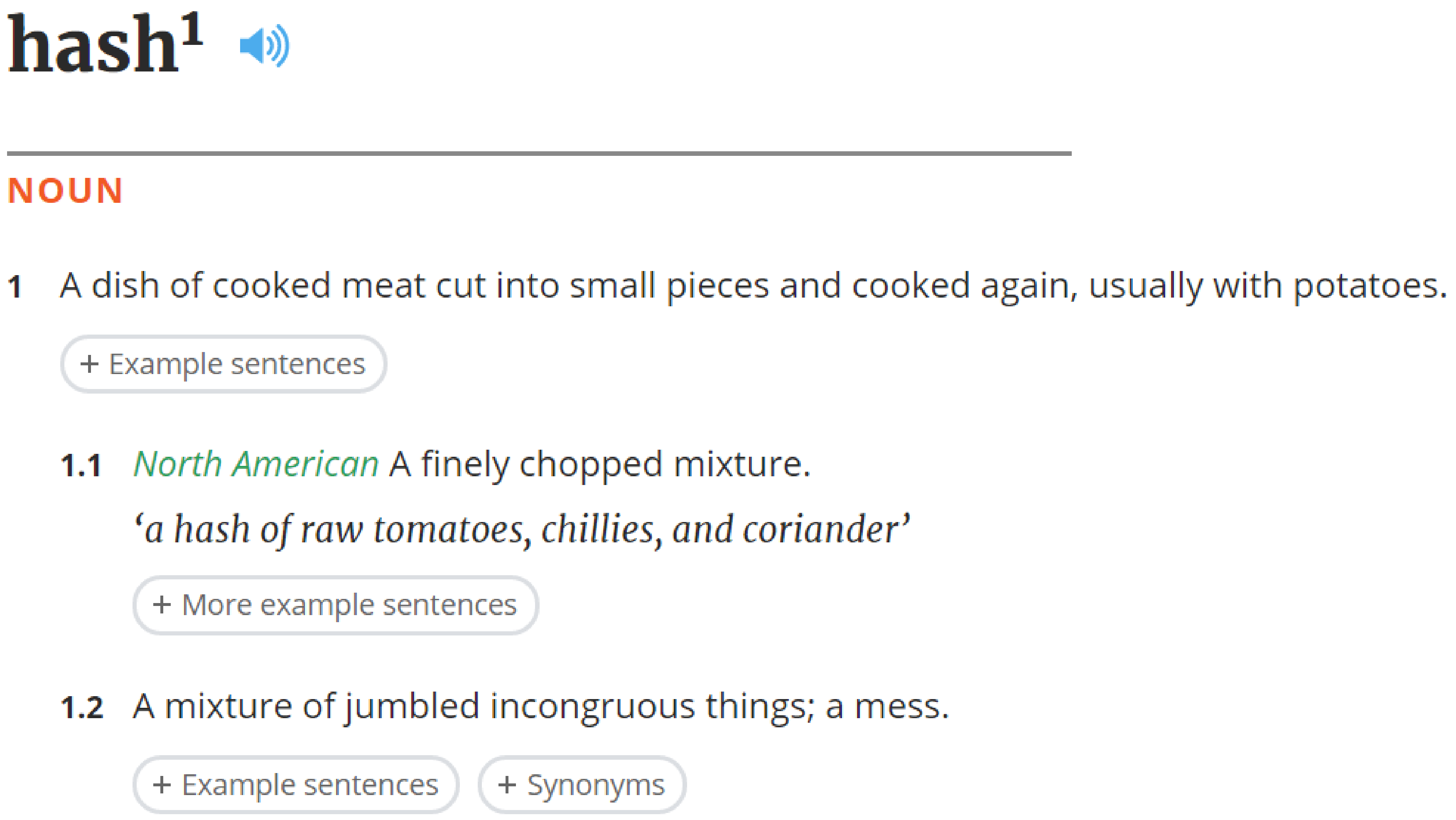
Answer: Collisions can occur.   Multiple keys can map to the same bucket and multiple keys can have the same value in each bucket.

Load factor of hash table

$$\equiv \frac{\text{number of entries}}{\text{number of buckets, } m}$$

One typically keeps the load factor below 1.
In the Java HashTable and HashMap classes, the default load factor is 0.75.

https://en.oxforddictionaries.com/definition/hash#hash_Noun_200



hash¹

NOUN

1 A dish of cooked meat cut into small pieces and cooked again, usually with potatoes.

+ Example sentences

1.1 *North American* A finely chopped mixture.

‘a hash of raw tomatoes, chillies, and coriander’

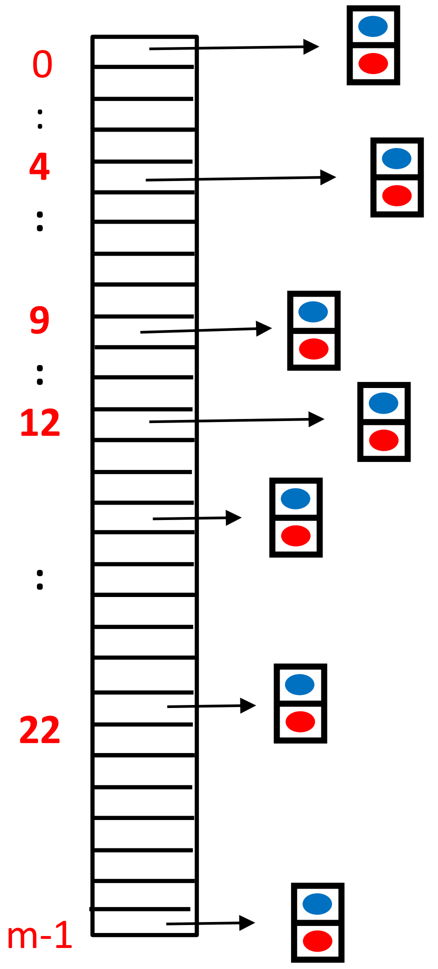
+ More example sentences

1.2 A mixture of jumbled incongruous things; a mess.

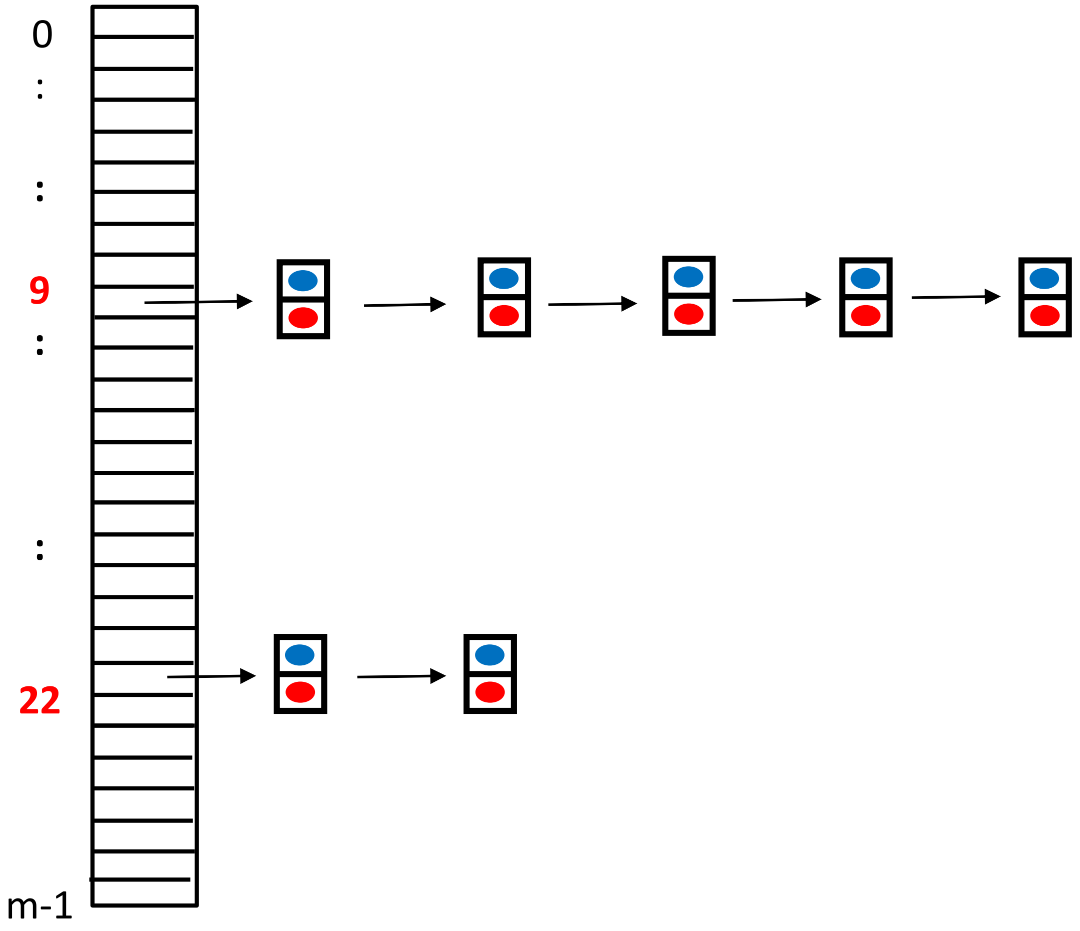
+ Example sentences + Synonyms

(Unrelated to the resin collected from the flowers of the cannabis plant i.e. hashish)

“Good Hash”



“Bad Hash”



Example

$$h : K \rightarrow \{0, 1, \ldots, m-1\}$$

Example: Suppose keys are McGill Student IDs, e.g. 260745918.

How many buckets to choose ?

Good hash function?

Bad hash function ?

Example

$$h : K \rightarrow \{0, 1, \ldots, m-1\}$$

Example: Suppose keys are McGill Student IDs, e.g. 260745918.

How many buckets to choose ? 100,000

Good hash function?

Bad hash function ?

Example

$h : K \rightarrow \{0, 1, \ldots, m-1\}$

Example: Suppose keys are McGill Student IDs, e.g. 260745918.

How many buckets to choose ? 100,000

Good hash function? rightmost 5 digits

Bad hash function ?

Example

$$h : K \rightarrow \{0, 1, ..., m\text{-}1\}$$

Example: Suppose keys are McGill Student IDs, e.g. 260745918.

How many buckets to choose ? 100,000

Good hash function? rightmost 5 digits

Bad hash function ? leftmost 5 digits

Performance of Hash Maps

- put(key, value)
- get(key)
- remove(key)

If load factor is less than 1 and if hash function is good, then these *operations are O(1) "in practice". This beats all potential map data structures that I considered last lecture.*

If we have a bad hash, we can choose a different hash function.

Performance of Hash Maps

- put(key, value)
- get(key)
- remove(key)
- contains(value) ?

Performance of Hash Maps

- put(key, value)
- get(key)
- remove(key)
- contains(value)

It will need to look at each bucket and
search its linkedlist for that value.
So we don’t want too big an array.

Performance of Hash Maps

- put(key, value)
- get(key)
- remove(key)
- contains(value)
- getKeys()
- getValues()

These last three methods all require traversing the hash table, which takes time $O(n+m)$ where n is number of entries.

Java HashMap <K, V> class

- In constructor, you can specify initial number of buckets, and load factor.

- How is hash function specified ?

Java HashMap <K, V> class

- In constructor, you can specify initial number of buckets, and maximum load factor
- How is hash function specified ?

  Use key's hashCode(), take absolute value, and compress it by taking mod of the number of buckets.

Java HashSet<E> class

Similar to HashMap, but there are no values. Use it to store a *set* of objects of some type.

- add(e)
- contains(e)
- remove(e)
- ……

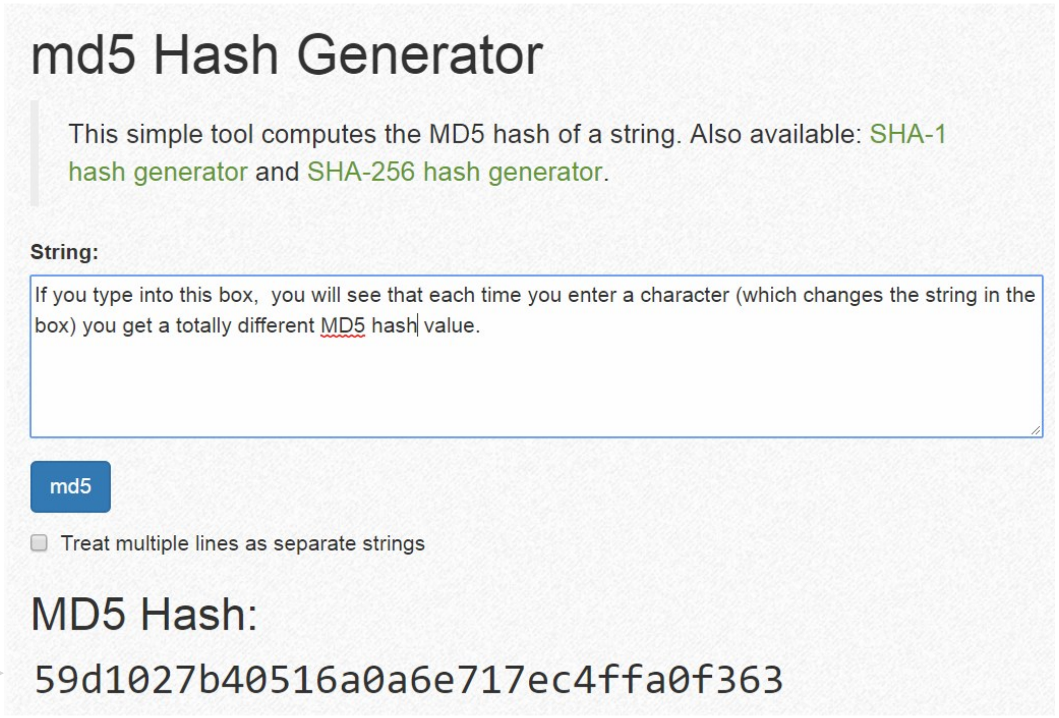
If hash function is good, then these operations are O(1).

Note that this is not a list! There is no 1st, 2nd, …. element.

Cryptographic Hashing

h: key (String) → hash value (e.g. 128 bits)

e.g. online tool for computing md5 hash of a string



md5 Hash Generator

This simple tool computes the MD5 hash of a string. Also available: SHA-1 hash generator and SHA-256 hash generator.

String:

If you type into this box, you will see that each time you enter a character (which changes the string in the box) you get a totally different MD5 hash value.

MD5 Hash:

```
59d1027b40516a0a6e717ec4ffa0f363
```

32 hexadecimal digits (128 bits)

Application: Password Authentication

e.g. Web server needs to authenticate users.

What map are we talking about here ?

Application: Password Authentication

{ (userID, password) } defines a *map*.

Keys are Strings (username or a number e.g. credit card)

Values are Strings (passwords)

Password Authentication (unsecure)

Suppose the {(userID, password)} map is stored in a *text* file on the web server where user logs in.

What would the user do to log in?

What would the web server do?

What could a mischievous hacker do?

Password Authentication (unsecure)

Suppose the {(userID, password)} map is stored in a *text* file on the web server where user logs in.

What would the user do to log in?

Enter username (key) and password (value).

What would the web server do?

Check if this entry matches what is stored in the map.

What could a mischievous hacker do?

Steal the text file, and login to user accounts.

Password Authentication (secure)

The map {(username, h(password)) } is stored in a file on the web server.

What would the user do?

Enter a username and password.

What would the web server do ?

What could a mischievous hacker try to do?

Password Authentication (secure)

The map {(username, h(password)) } is stored in a file on the web server.

What would the user do?

Enter a username and password.

What would the web server do ?

Hash the password and compare to entry in map.

What could a mischievous hacker try to do?

Password Authentication (secure)

The map {(username, h(password)) } is stored in a file on the web server.

What would the user do?

Enter a username and password.

What would the web server do ?

Hash the password and compare to entry in map.

… and throw away the password.

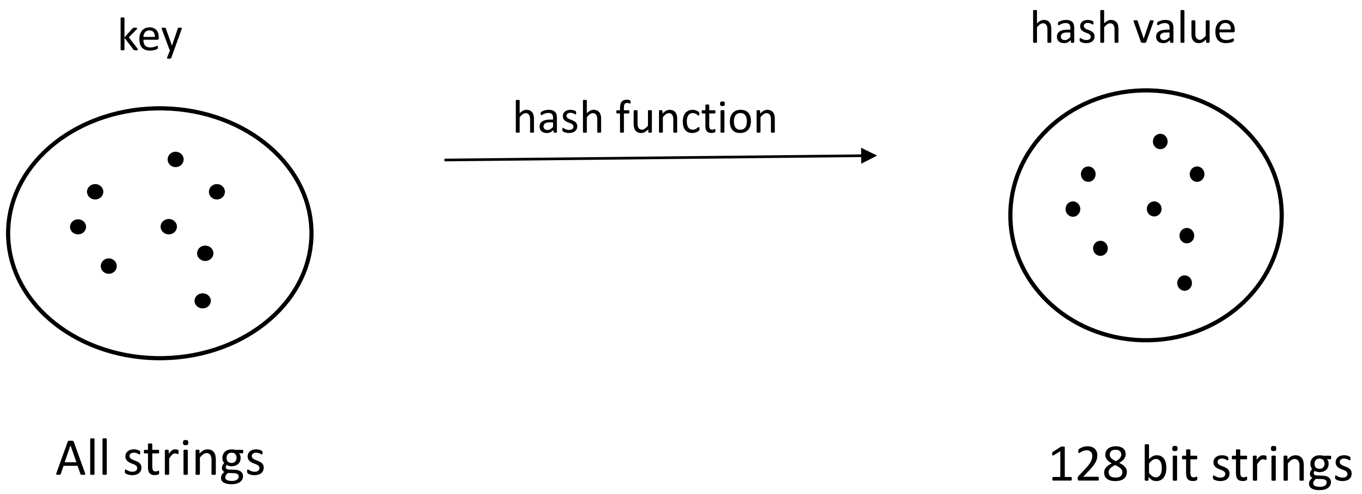
What could a mischievous hacker try to do?

“Brute force” or “dictionary” attack.

Cryptographic Hashing

We want a hash function h( ) such that one can infer almost nothing about the key from the hash value h(key).

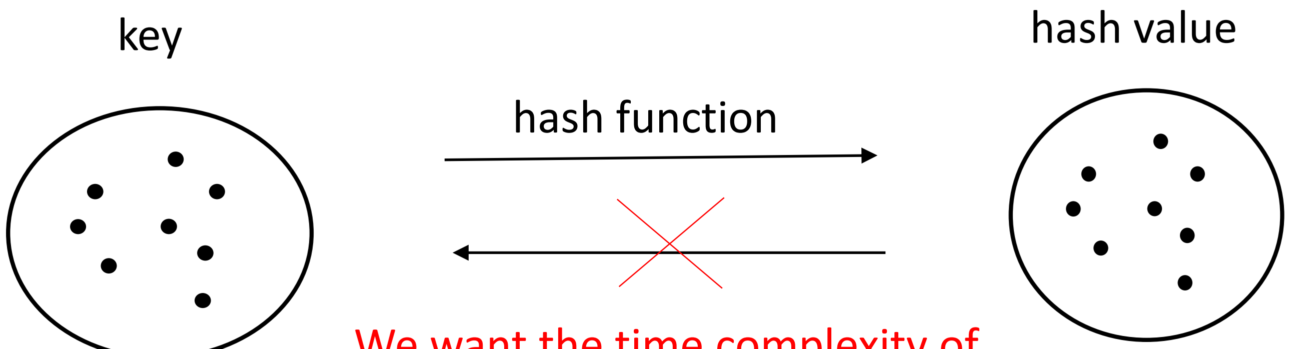
Small changes in the key give very different hash values.



Cryptographic Hashing

We want a hash function h( ) such that one can infer almost nothing about the key from the hash value h(key).

Small changes in the key give very different hash values.

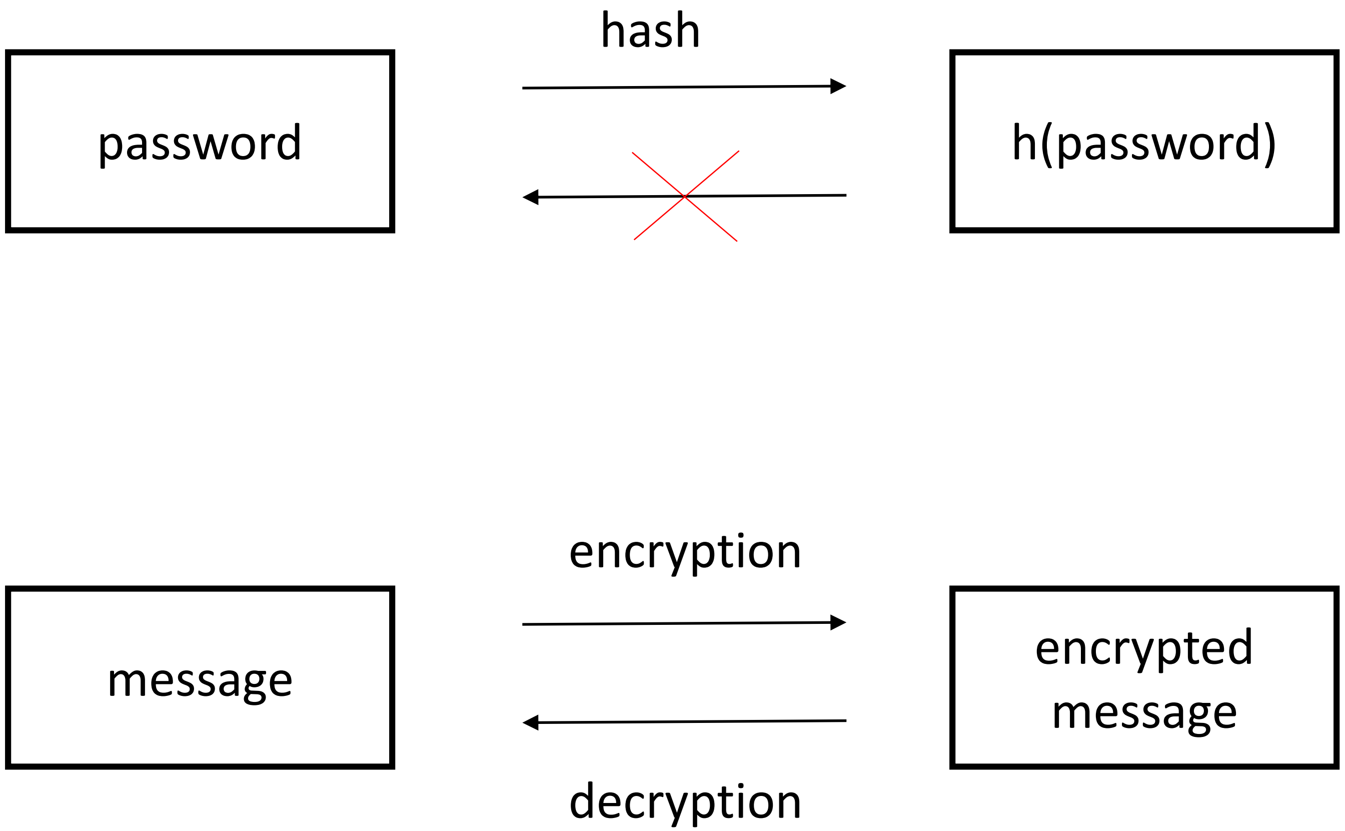


We want the time complexity of finding *some* key that maps to a given a hash value to be very high (such as brute force).

All strings

128 bit strings

Do not confuse hashing with encryption/decryption.



You learn about RSA encryption in MATH 240.

Good luck on Quiz 4.

Have a good weekend.