

Hash Codes and Hashtables

Simon Robinson
<http://TechieSimon.com>
@TechieSimon



pluralsight 
hardcore dev and IT training

- 
- Hash codes enable collections with hash tables to work.
 - Such as `Dictionary<TKey, TValue>`.
 - Requirements of a hashing function.
 - Consistency with equality.
 - How to code up `GetHashCode()`.
 - Combine fields with XOR.
 - Exclusive OR.
 - What it does.
 - Why it's so good for hash codes.

How Hashtables Use Hash Codes

`object.GetHashCode()`

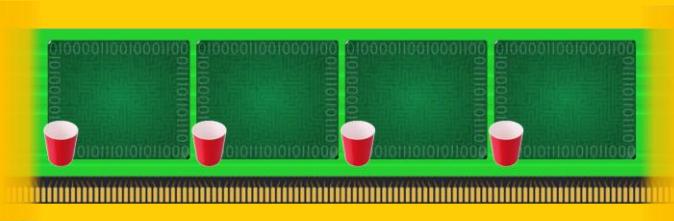
Required by hash tables

Hash tables
speed up
looking up items



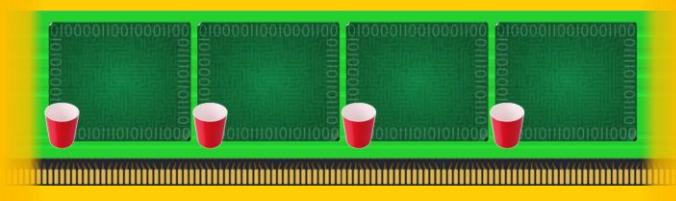
Collections with Hashtables

Dictionary< TKey , TValue >



(Hashtable of keys)

HashSet< T >

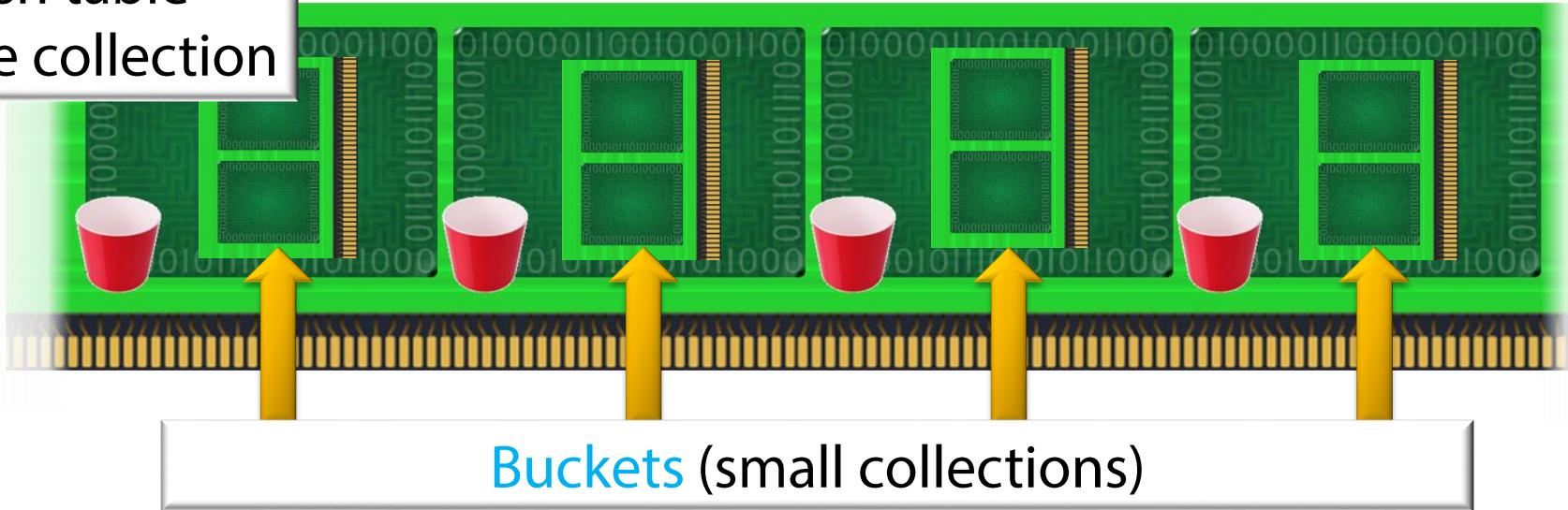


(Hashtable of values)



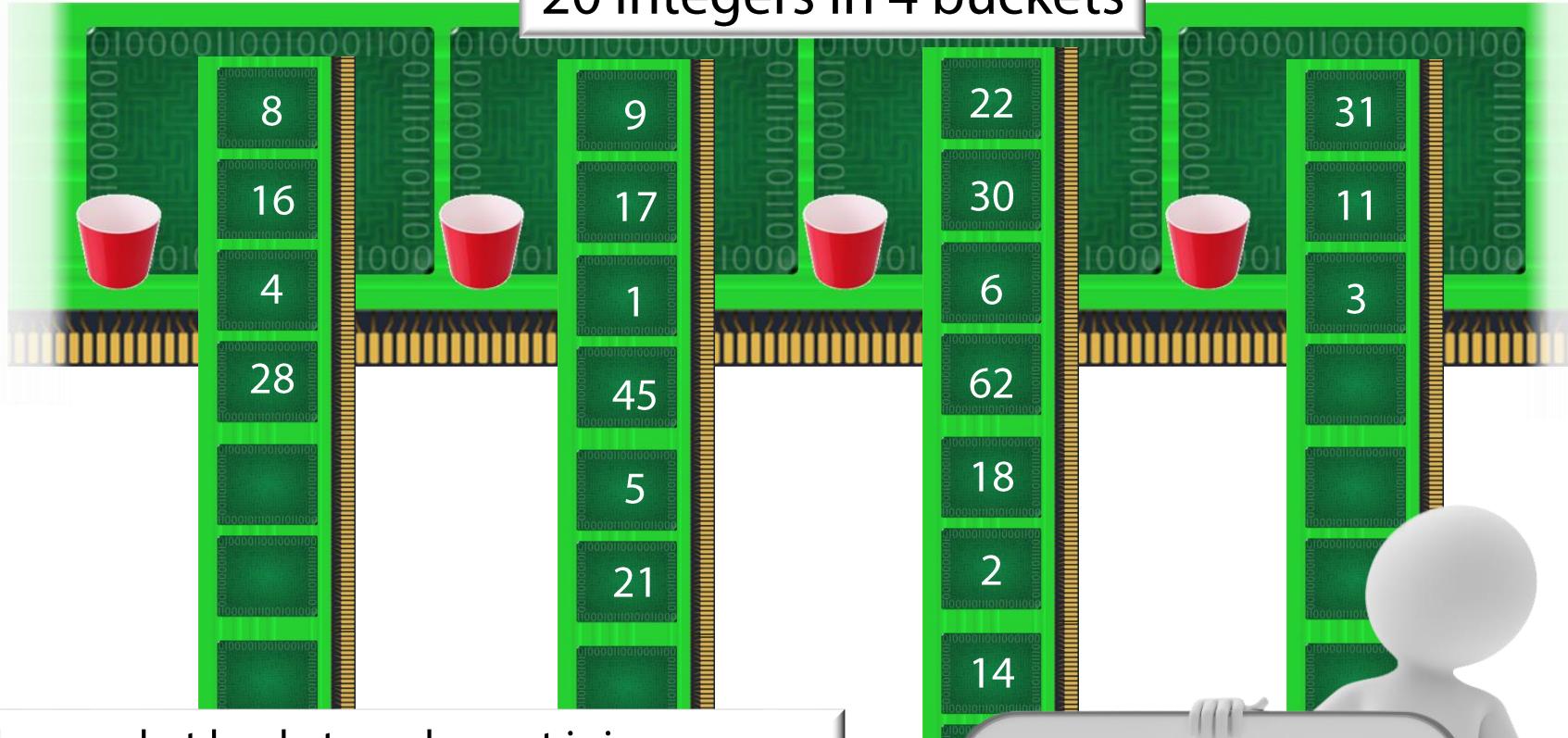
How Hashtables Work

Hash table
– large collection



How Hashtables Work

20 integers in 4 buckets



If you know what bucket an element is in...

...You only need to search for it in that bucket

...You can find it more quickly



Only works if elements
are evenly spread

How Do You Choose a Bucket?

var x;

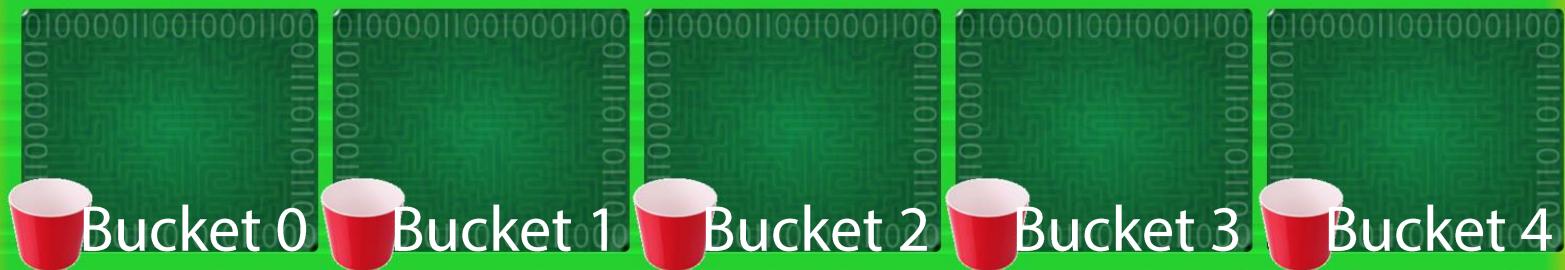
Value (can be any type)
to go in hash table

```
int bucketIndex = x.GetHashCode() % nBuckets
```

This is the bucket the value
should go in

Adding Strings to a HashSet

```
var myDict = new Dictionary<string, T>()
```



To add "apple"

"apple".GetHashCode() → returns 1 657 858 284
"apple".GetHashCode() % 5 → returns 4

Adding Strings to a HashSet

```
var myDict = new Dictionary<string, T>()
```



To add "pear"

"apple".GetHashCode()	returns -422 187 275
"apple".GetHashCode() % 5	returns 0

Looking up Strings in a HashSet

```
var myDict = new Dictionary<string, T>()
```



This is all for performance
so you don't need
to search many elements

Dict["apple"]

de()

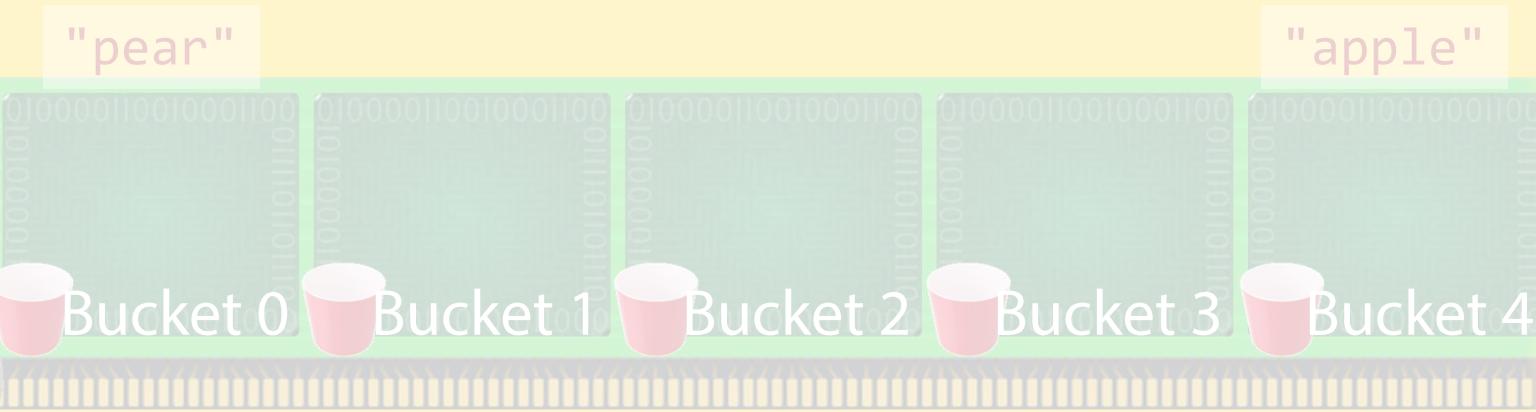
de() % 5

• returns 1 657 858 284

returns 4

Hash Codes and Equality

```
var myDict = new Dictionary<string, T>()
```



This gives the
link
to equality

Looking up a String

```
var myDict = new Dictionary<string, T>()
```



myDict["Apple"]

"Apple".GetHashCode()

```
"Apple".GetHashCode() % 5
```

► returns 1 657 859 532

returns 2

Looking up a String

```
var myDict = new Dictionary<string, T>()
```

"pear"

"apple".GetHashCode() % 5

"apple"

leads to bucket 4

"Apple".GetHashCode() % 5

leads to bucket 2

Different strings
lead to
different buckets

Looking up a String

Suppose keys are case-insensitive:

```
var myDict = new Dictionary<string, string>
    (StringComparer.OrdinalIgnoreCase);
```

With this dictionary, if we look up "Apple"...

...we expect to match with "apple":

... This only works if...

`GetHashCode("apple") % 5`

`GetHashCode("Apple") % 5`

} → leads to
the same
bucket



`string.GetHashCode()` won't do this

- so this dictionary can't use `string.GetHashCode()`

GetHashCode() Requirements

If two values x and y evaluate equal ...
...then they **must** have the same hash code



http://msdn.microsoft.com/en

IEqualityComparer(T) Meth...

Microsoft

MSDN subscriptions Get tools Sign in

Developer Network

var myDict = new Dictionary<string, string>(StringComparer.OrdinalIgnoreCase);

Collapse All Export (0) Print

MSDN Library .NET Development .NET Framework 4.5 .NET Framework Class Library System.Collections Namespaces System.Collections.Generic IEqualityComparer(T) Interface IEqualityComparer(T) Methods Equals Method GetHashCode Method

IEqualityComparer<T> Methods

.NET Framework 4.5 | Other Versions | 1 out of 1 rated this helpful - Rate this topic

Invoked by the dictionary to check if keys are equal

Methods

	Name	Description
	Equals	Determines whether the specified objects are equal.
	GetHashCode	Returns a hash code for the specified object.

Top

See Also

Invoked by the dictionary to get hash codes for the keys

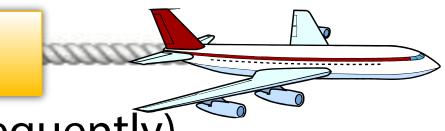
IEqualityComparer(T) Interface
System.Collections.Generic Namespace

GetHashCode() Requirements

If two values x and y evaluate equal ...
...then they **must** have the same hash code



Hash codes must be quick to calculate
(Because dictionaries etc. invoke GetHashCode() frequently)



Hash codes are evenly spread across **int** values



(Because you want values to be evenly spread over the buckets)



What is Exclusive-OR (XOR)

XOR is a bitwise operator

For each bit: Ans = 1 if bits are different
0 otherwise

$$\begin{array}{r} 24 \\ \text{XOR} \\ 45 \\ \hline = 53 \end{array}$$

$$\begin{aligned} &= b \quad 00011000 \\ &= b \quad 00101101 \\ &\hline &= b \quad 00110101 \end{aligned}$$



- Very fast (can be done in hardware)
- Answer not directly related numerically to inputs

XOR tends to give even spread when you take the remainder





Math For Programmers

This course covers the maths behind how your computer stores and manipulates data. You'll learn how to read binary and hexadecimal, how both integers and floating point numbers are stored and the limitations of using them. Advice on best practices and how to work effectively with boolean values and bitwise operators.

8+1 4

Tweet 7

Like 44 Share

in Share 7

Table of Contents

Description

Exercise Files

Assessment

Authored by: Simon Robinson

Duration: 5h 16m

Level: Intermediate

Released: 8/26/2013

Course Rating: ★★★★★

You are currently not signed in. Please [sign in](#) to access subscriber-only content.

+ expand all | - collapse all | remove highlights

Progress

Duration

	Math for Programmers	00:08:39
	Types of Data	00:33:25
	Working in Binary	00:45:55
	Integers	00:58:15
	Floating Point Numbers	01:21:54
	Logic, Booleans and Bitwise Operations	00:47:26
	Errors and Accuracy	00:40:53



feedback & support

Hash Code Algorithms

Basic Procedure:

1. Take each field that's used to evaluate equality

For each field...

- (i). Map any 'equal' values into the same value
- (ii). Get the hash code of the mapped field value

2. XOR the individual hash codes together

This is likely to be good enough for most cases
(but there are more advanced techniques)



Summary

- Motivation of GetHashCode() is to support some collections.
 - Notably `Dictionary< TKey, TValue >`.
- If instances evaluate as equal, they MUST return the same hash code.
- Hash codes should be:
 - Quick to compute.
 - Evenly spread out.
- Combine field hash codes with XOR:
 - Good way to satisfy these requirements.