

The Power of Hashing

Mikkel Thorup

University of Copenhagen



Distribute objects in storage boxes.

Where did we put

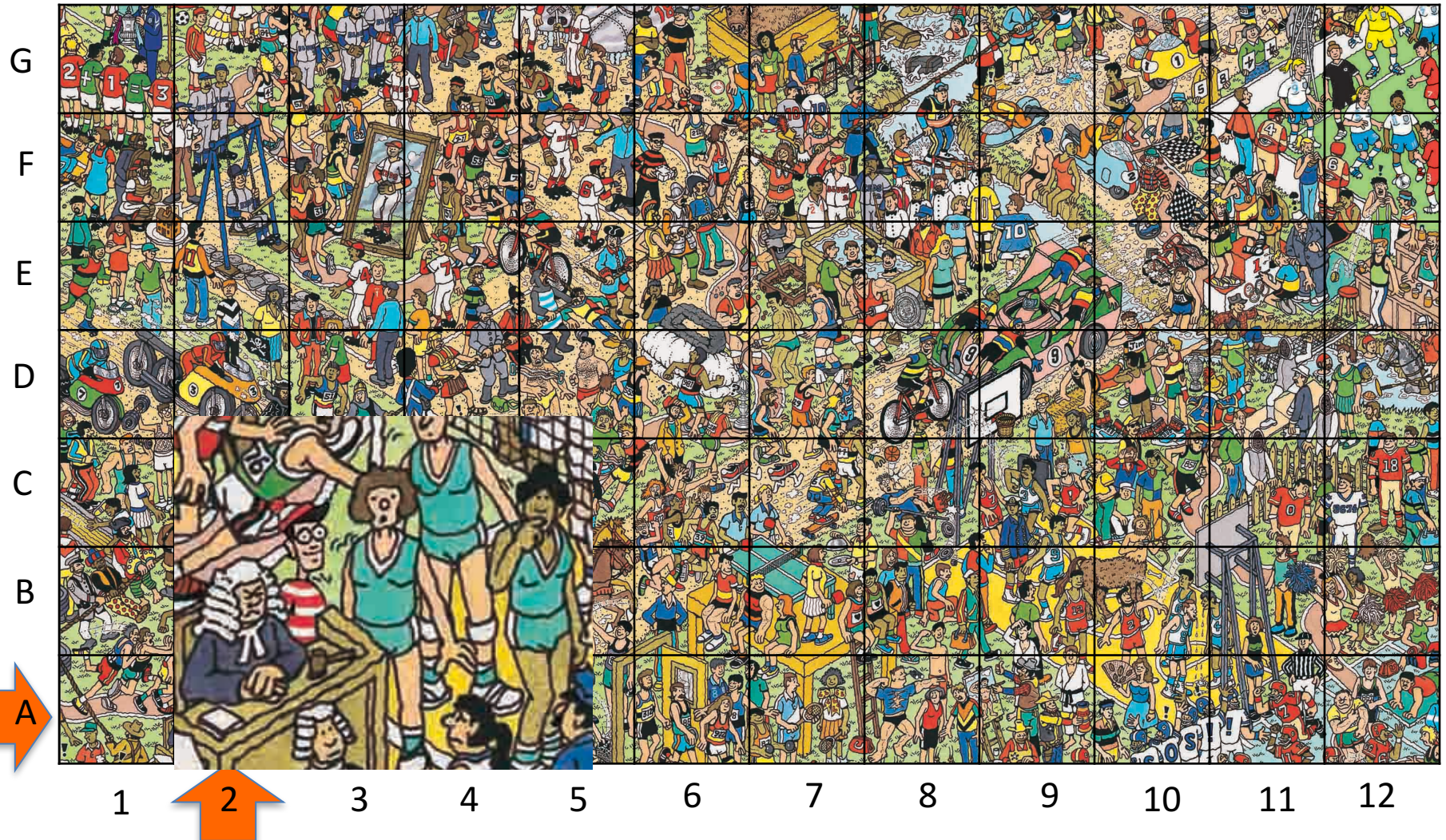


?



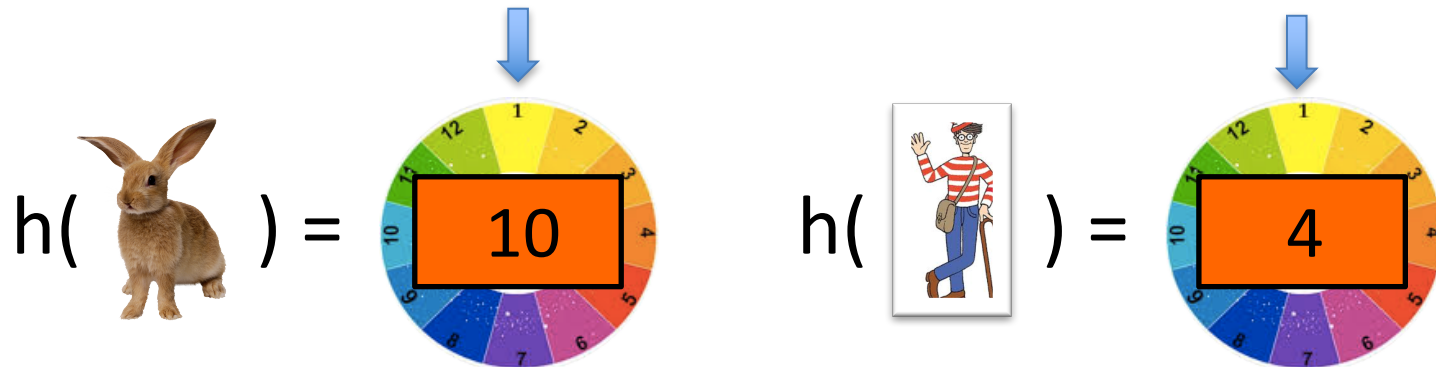
| | | |
|---|--|---|
| 1  | 5  | 9  |
| 2  | 6  | 10  |
| 3  | 7  | 11  |
| 4  | 8  | 12  |

Where is ?




Fully-Random Hash Functions

What we want is a re-computable fully-random hash function h assigning independent random hash value to every possible object:

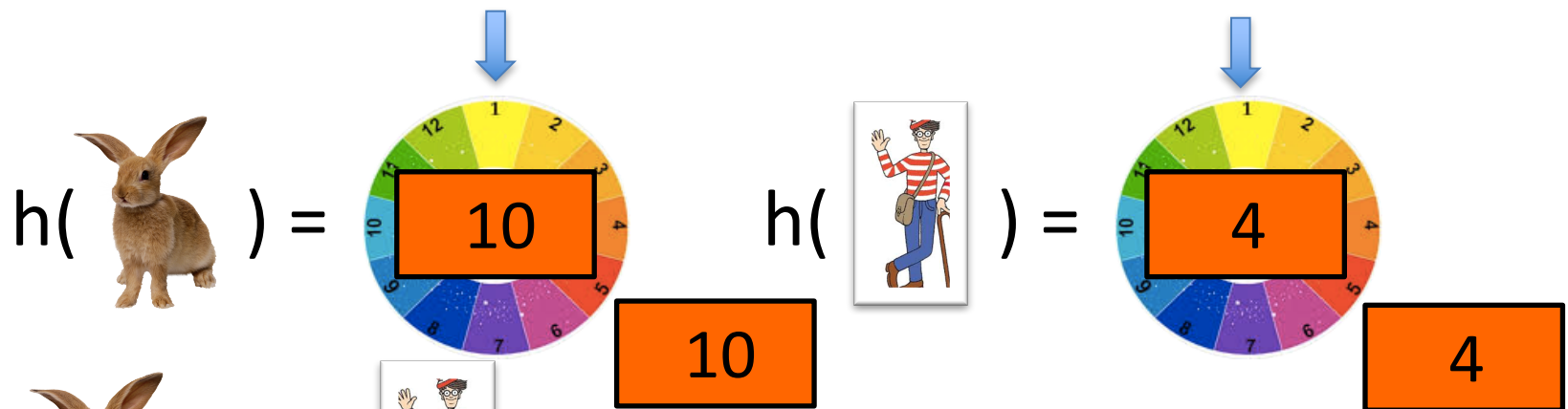


$h(\text{rabbit}) = h(\text{pirate})$ with probability $1/12$.


With 18 other objects, on average  expected to share box with $18/12=1.5$ objects.

Fully-Random Hash Functions

What we want is a **re-computable** fully-random hash function h assigning independent random hash value to every possible object:





$h(\text{rabbit}) = h(\text{character})$ with probability $1/12$.

With 18 other objects, on average  expected to share box with $18/12=1.5$ objects.

Random Hash Functions

Re-computable random hash function h
 assigning random hash value to every object.

On computer objects have numbers:  **385**,  **936**

Pick two random  **945** ,  **749** < 1009

$$h(\text{rabbit}) = (((\text{945} \times \text{385} + \text{749}) \bmod 1009) \bmod 12) + 1 = 2 \neq 0$$

$$h(\text{Wile E. Coyote}) = (((\text{945} \times \text{936} + \text{749}) \bmod 1009) \bmod 12) + 1 = 5 = \text{prob} < 1/12$$

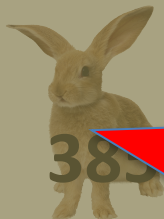
Distribute objects
in storage boxes.



936



218



385

50
CANCELLED

$$(((\text{807} \times \text{679} + \text{495}) \text{ mod } 1009) \text{ mod } 12) + 1$$

mod 1009) mod 12) + 1

= 130

807



SAW RANDOM CHOICES

1

5

3

4

12

Probabilistic guarantees

- Research in randomized algorithms often assumes abstract fully random hash functions.
- Often it suffices with implementable probabilistic guarantees like:
- **Low collision probability.** With t different hash values, two given objects get same hash value with probability $\leq 1/t$

$$h(\text{🐰}) = h(\text{👤}) \text{ with probability } \leq 1/t.$$

- **Application.** Hash table for storing and finding objects in computer memory, signatures, etc.

- We try to understand what types of probabilistic properties are possible with realistic hash functions, e.g., the low collision probability from before, or
- **Fair minwise hashing.** In any given set of objects, each object has (almost) the same chance of getting the smallest hash value.
- **Applications.** Big Data. Classification of web-pages, recommendation systems, spam filters, artificial intelligence and machine learning, many randomized algorithms such as QuickSort, computational geometry etc.

| Peter | |
|---------------------------|------|
| | hash |
| The Shawshank Redemption | 83 |
| The Godfather | 21 |
| The Godfather II | 44 |
| Star Wars Episode V | 07 |
| The Dark Knight | 26 |
| Apocalypse Now | 86 |
| Pulp Fiction | 78 |
| Goodfellas | 68 |
| The Lord of the Rings III | 20 |
| Fight Club | 95 |
| Smallest hash value | 07 |

| Hans | |
|---------------------|------|
| | hash |
| A Hard Day's Night | 75 |
| The Godfather | 21 |
| Singin' in the Rain | 63 |
| Finding Nemo | 40 |
| Repulsion | 77 |
| Inside Out | 39 |
| Boyhood | 88 |
| King Kong | 64 |
| Toy Story 2 | 73 |
| The Seven Samurai | 15 |
| Smallest hash value | 15 |

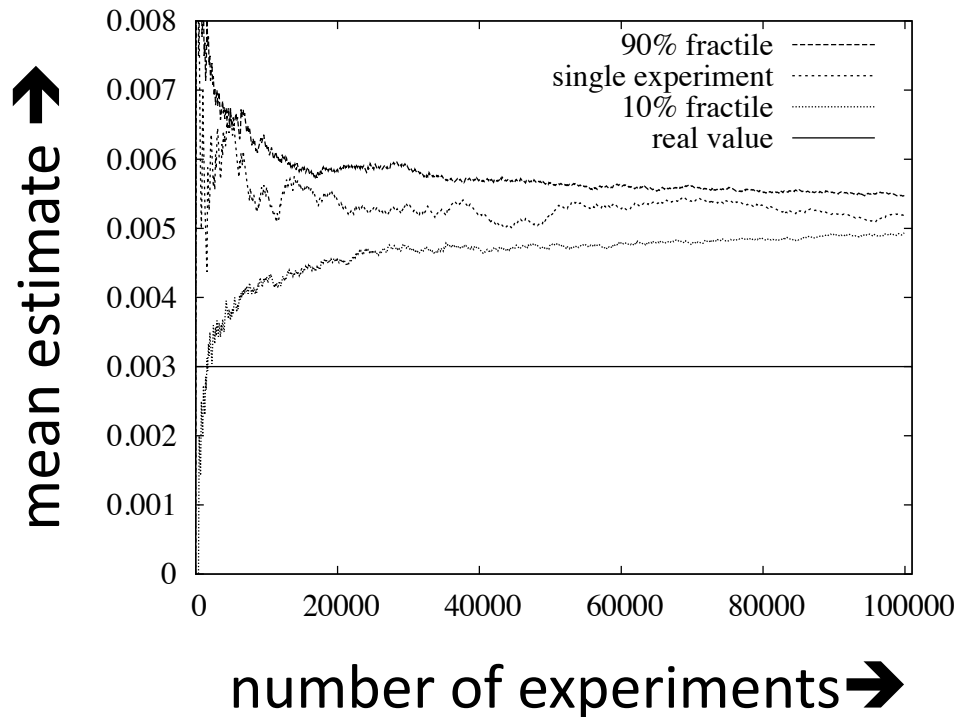
| Lars | |
|--------------------------|------|
| | hash |
| The Godfarther | 21 |
| Raiders Of The Lost Ark | 45 |
| Star Wars Episode V | 07 |
| The Shawshank Redemption | 83 |
| Jaws | 15 |
| Goodfellas | 68 |
| Apocalypse Now | 86 |
| Singin' In The Rain | 63 |
| Pulp Fiction | 78 |
| Fight Club | 95 |
| Smallest hash value | 07 |

Hashing films to see who has similar taste:
Those with same smallest hash value ?!?

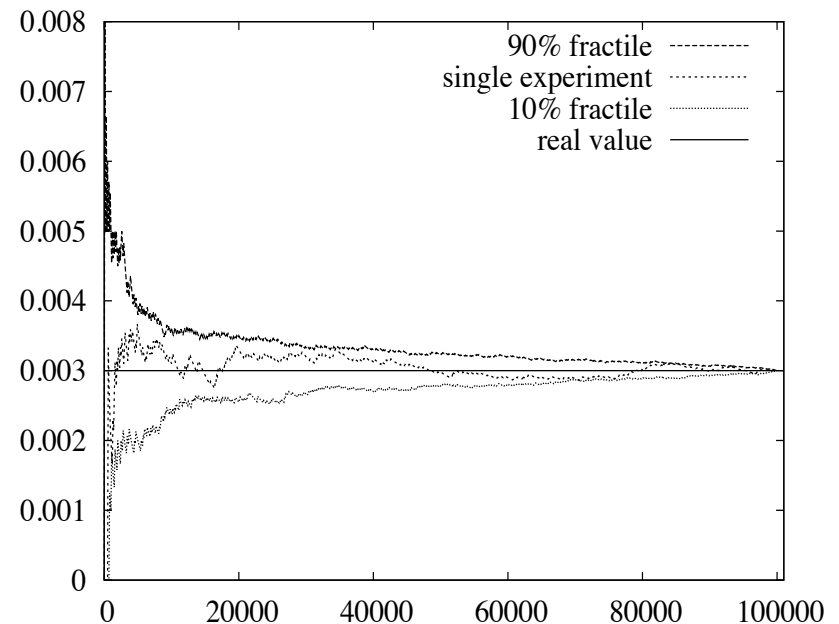


- [illegible]

Similarity estimation experiment



Bad use of hashing



Proper use of hashing

Sorry

- Will now switch to a TeX based presentation for math...

