A Few Sketches of Sketches

Adam Marcus
B12
@marcua

A <u>sketching data structure</u> summarizes a dataset, trading off accuracy for space.

Examples

- Sum
- Count
- Is item x in dataset D?
- How many X's are in D?
- How many distinct items are in D?
- What is a histogram of D?
- ...

Examples

- Sum
- Count
- Is item x in dataset D? → Bloom Filter
- How many X's are in D? → Count-min Sketch
- How many distinct items are in D?
- What is a histogram of D?
- ...

Bloom Filters

[Bloom, 1970]

Is item x in dataset D?

is http://badurl.com malicious? → Google

↓

Gigabytes

Bloom Filters

[Bloom, 1970]

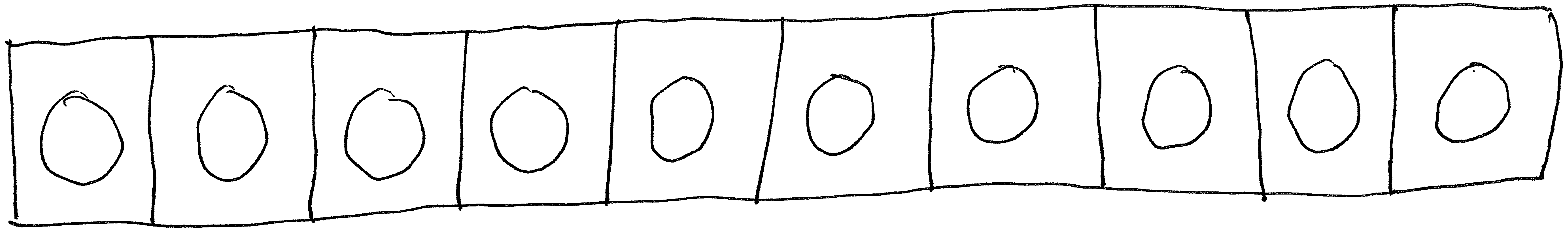
Is item x in dataset D?

is http://badurl.com malicious? → Chrome Bloom Filter → Google

Chrome Bloom Filter ↓ Megabytes

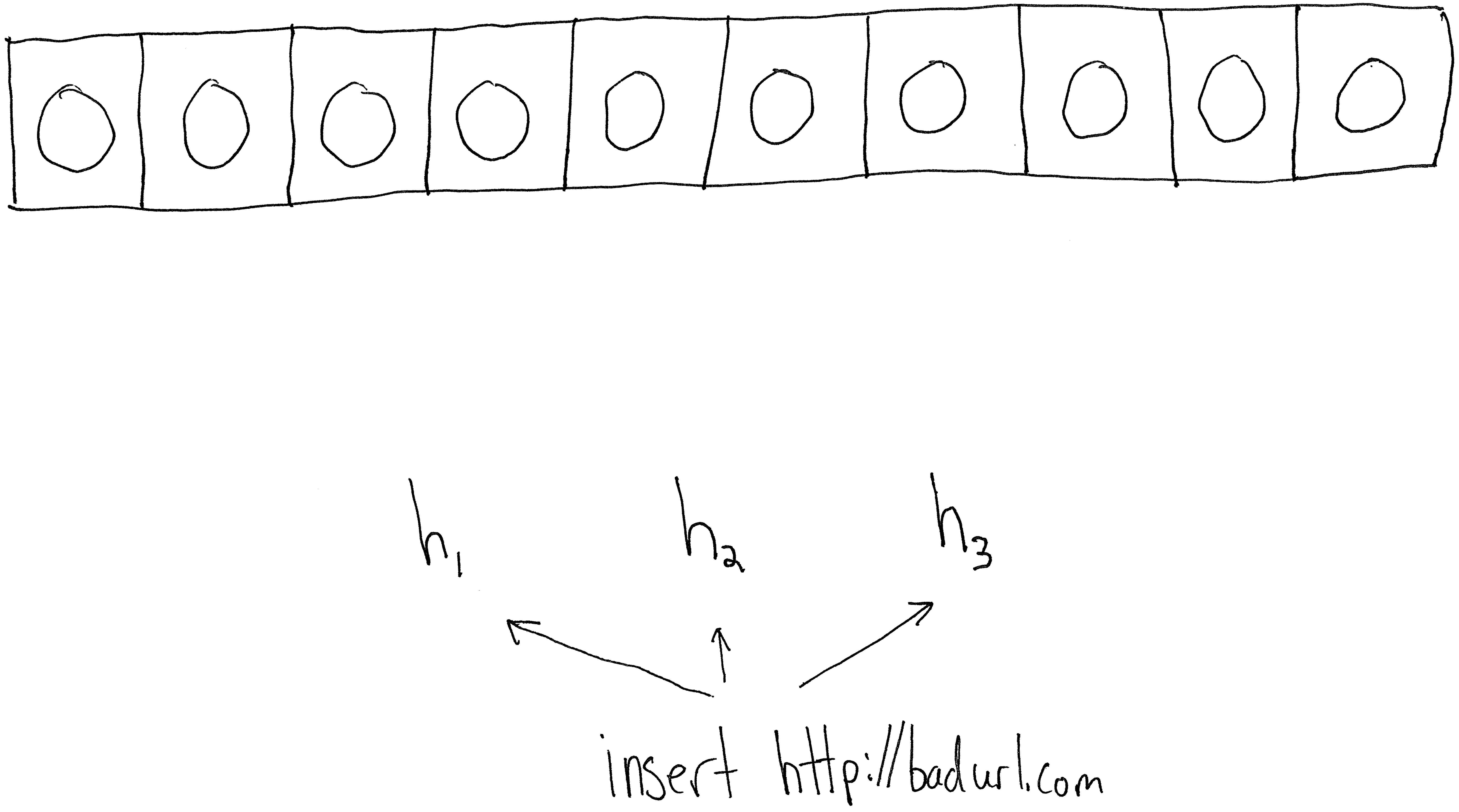
Google ↓ Gigabytes

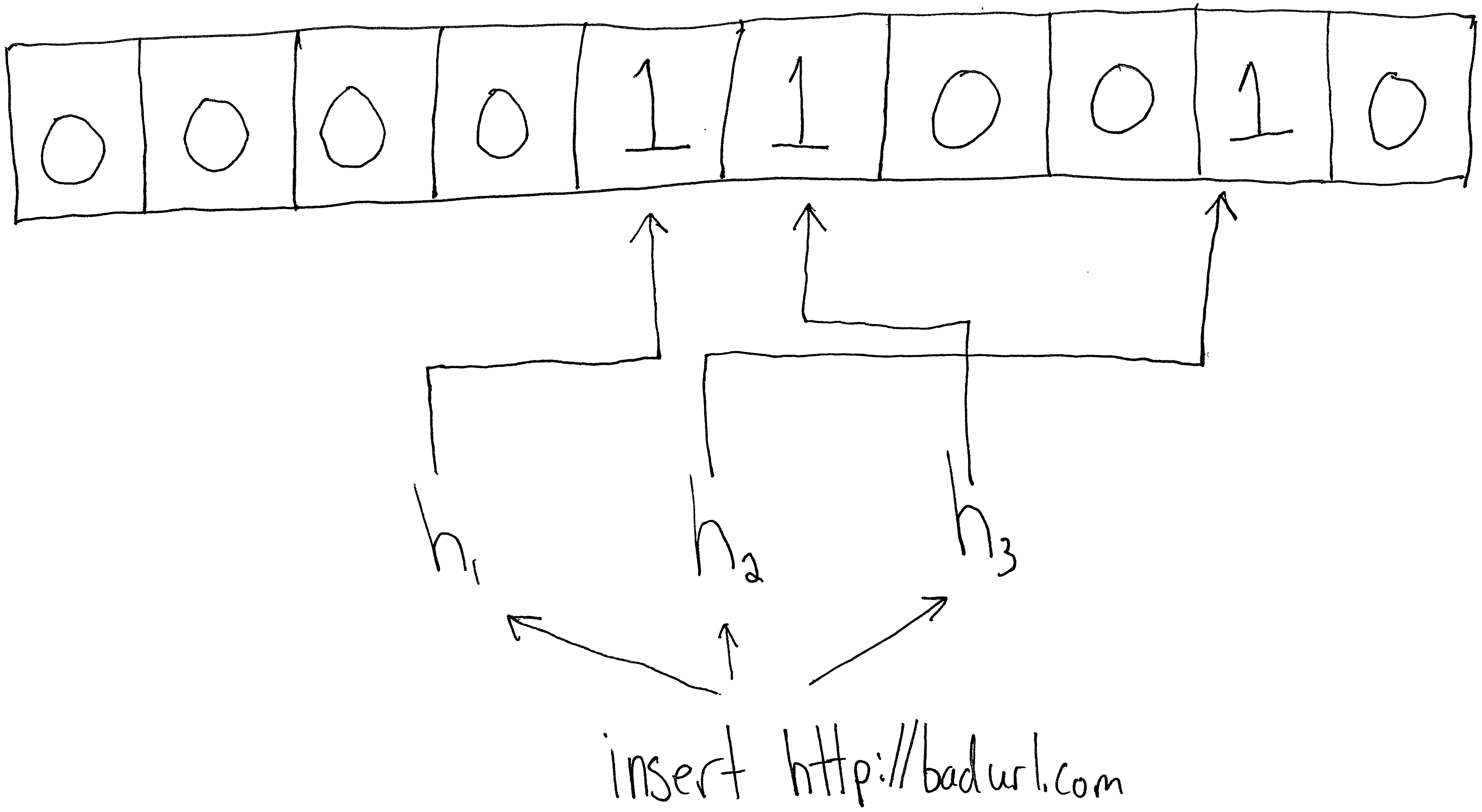
How it Works

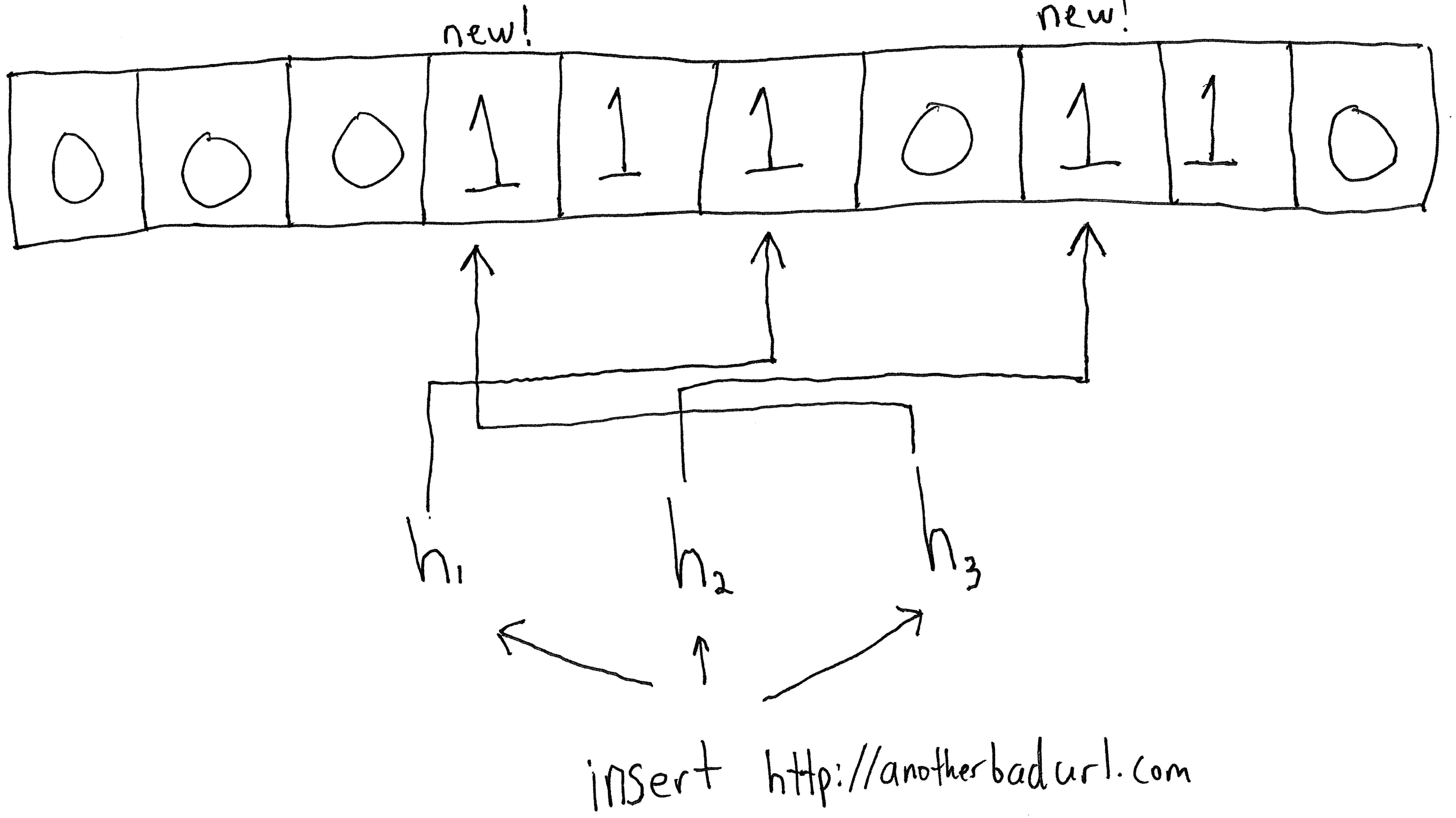


h_1 h_2 h_3

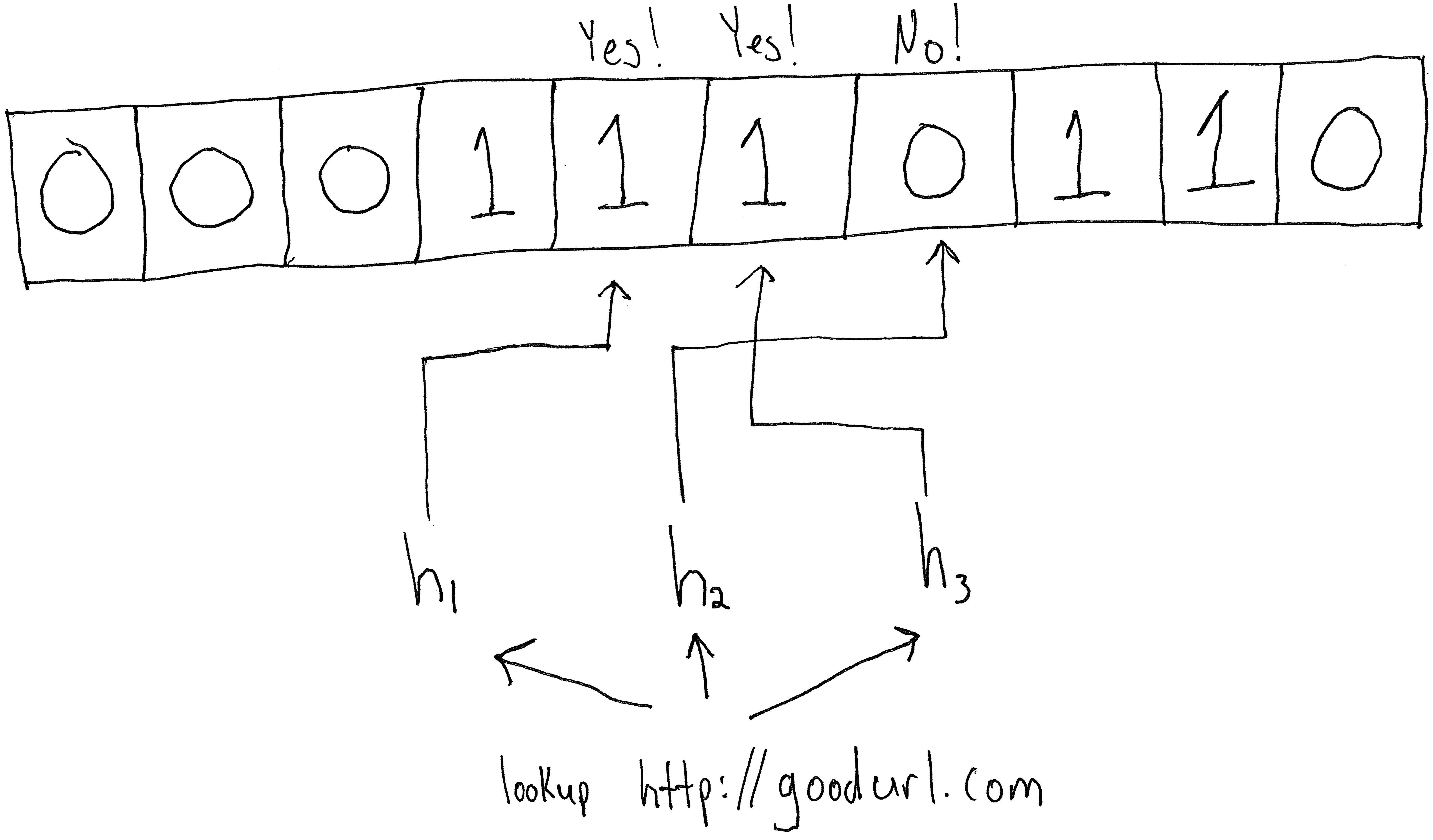
How it Works



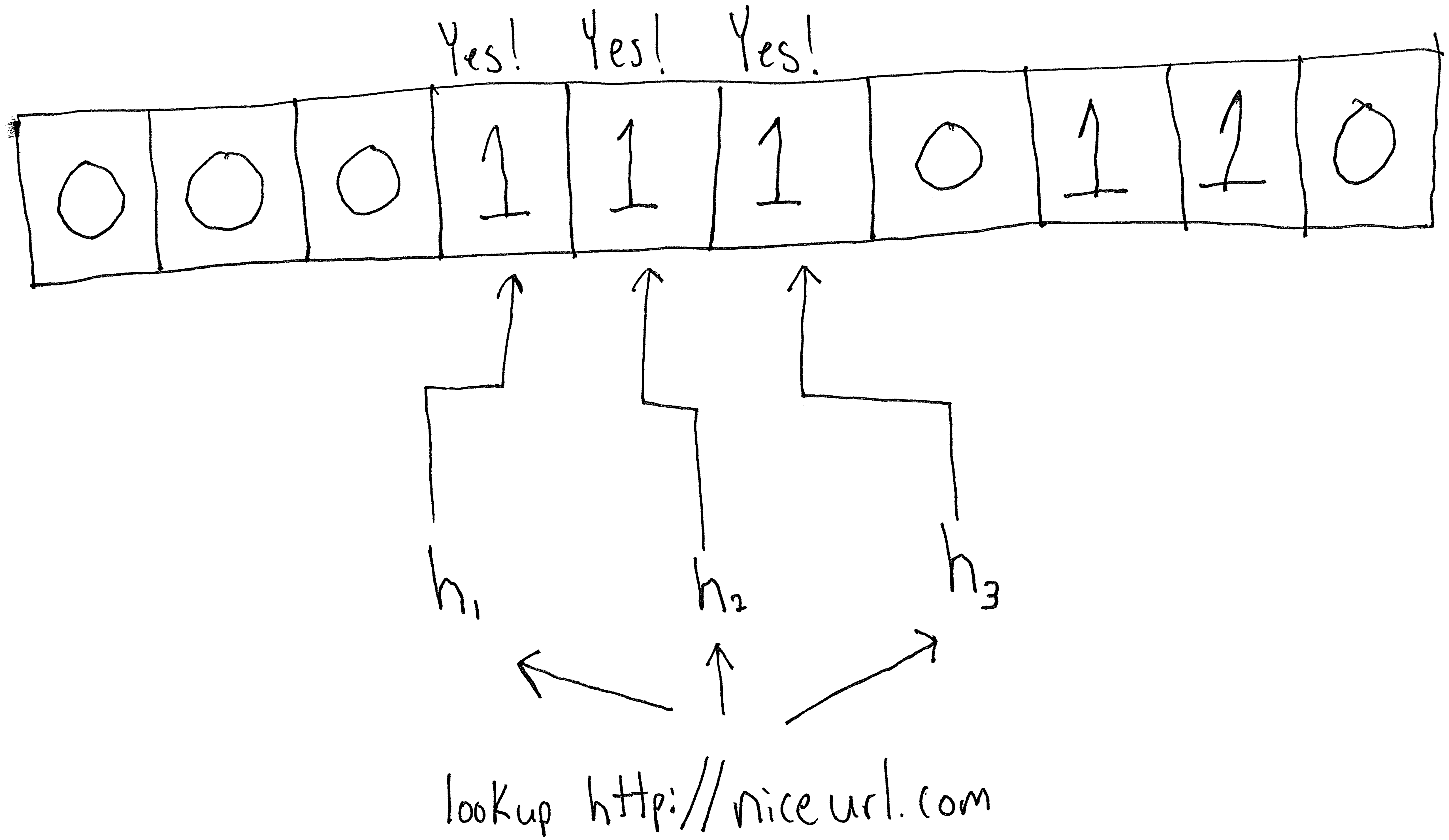

0 0 0 0 1 1 0 0 1 0
h1
h2
h3
insert http://badurl.com


new!
new!
0
0
0
1
1
1
0
1
1
0
h1
h2
h3
insert http://anotherbadurl.com

Nope!



False Positive!



Bloom filters guarantee an item is not present...

...and have a small chance of telling you an item is present when it's not

How Small a Chance?

$$\#\text{ of bits} = -\frac{(\#\text{ of items}) \ln(\text{false positive rate})}{(\ln 2)^2}$$

false positive rate	# bits per item
1/100	~ 10
1/1,000	~ 15
1/10,000	~ 20

false positive rate = 1/10,000
bits per item = ~20

10 million URLs
30 characters each

Size of strings alone =~ 1,144 MB
Bloom Filter =~ 24 MB!!!

Optimal # Hash Functions?

$$\#\text{ hash functions} = \left(\frac{\#\text{ bits}}{\#\text{ items}}\right)\ln 2 \approx .69\left(\frac{\#\text{ bits}}{\#\text{ items}}\right)$$

20 bits/item => ~14 hash functions

Count-Min Sketch

[Cormode & Muthukrishnan, 2003]

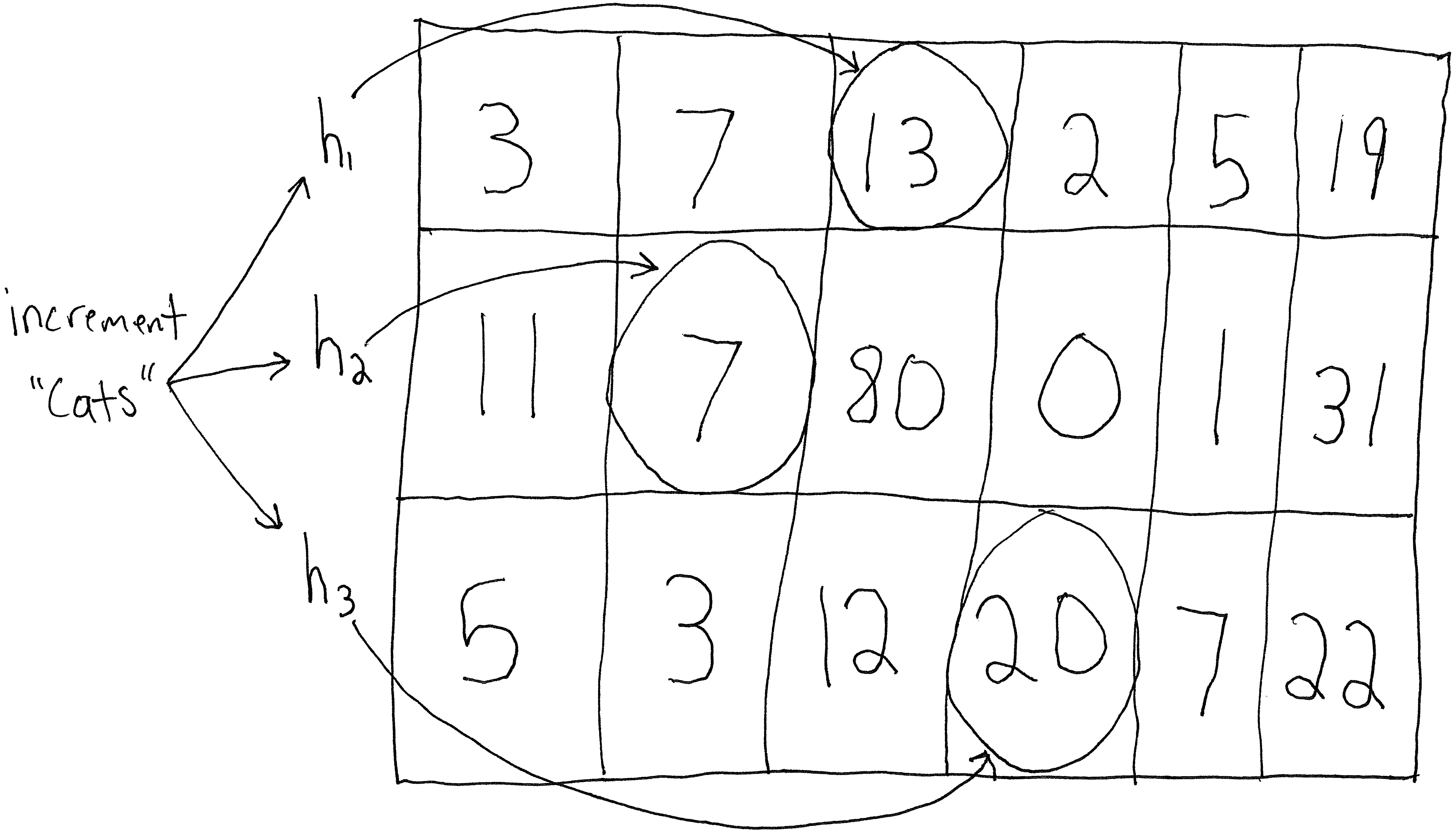
How many X's are in dataset D?

Cats	35,000,000
Bats	4,000,000
Flats	2,000,000
Mats	100,000
Drats	50,000

Count the number of words on the web

How it Works

h_1	3	7	12	2	5	19
h_2	11	6	80	0	1	31
h_3	5	3	12	19	7	22


increment "cats"
h1
h2
h3
3 7 13 2 5 19
11 7 80 0 1 31
5 3 12 20 7 22

Count "flats"

h_1	3	7	13	2	5	19
h_2	11	7	80	0	1	31
h_3	5	3	12	20	7	22

5, 7, 12

Count "flats"

h_1	3	7	13	2	5	19
h_2	11	7	80	0	1	31
h_3	5	3	12	20	7	22

$$\min(5, 7, 12) = 5$$

Count "flats"

h_1	3	7	13	2	5	19
h_2	11	7	80	0	1	31
h_3	5	3	12	20	7	22

$$\text{overestimate} \leq \frac{2\left(\#\text{total increments}\right)}{\#\text{counters per hash}}$$

$$\text{With probability } 1 - \left(\frac{1}{2}\right)^{\#\text{hashes}}$$

1 Million Increments

overestimate ≤ 100 → 20,000 counters per hash

probability .999 → 10 hashes

20,000 counters X 10 hashes X 4 bytes

$<$ 800KB !!!

A Count-Min Sketch Provides

... an overestimate of the count

... that is more meaningful for the frequent items ("heavy hitters") in your data

Final Thoughts

Bloom Filter

Count-Min Sketch

HyperLogLog

T-Digest

...

Use Sketches When

Large data → Small summary

Unbounded stream → Bounded memory

Embrace Randomness

Sketch Away!

Adam Marcus
B12
@marcua