Data Structures and Algorithms
Counting Sketches

CS 225
G Carl Evans

April 26, 2023

Department of Computer Science

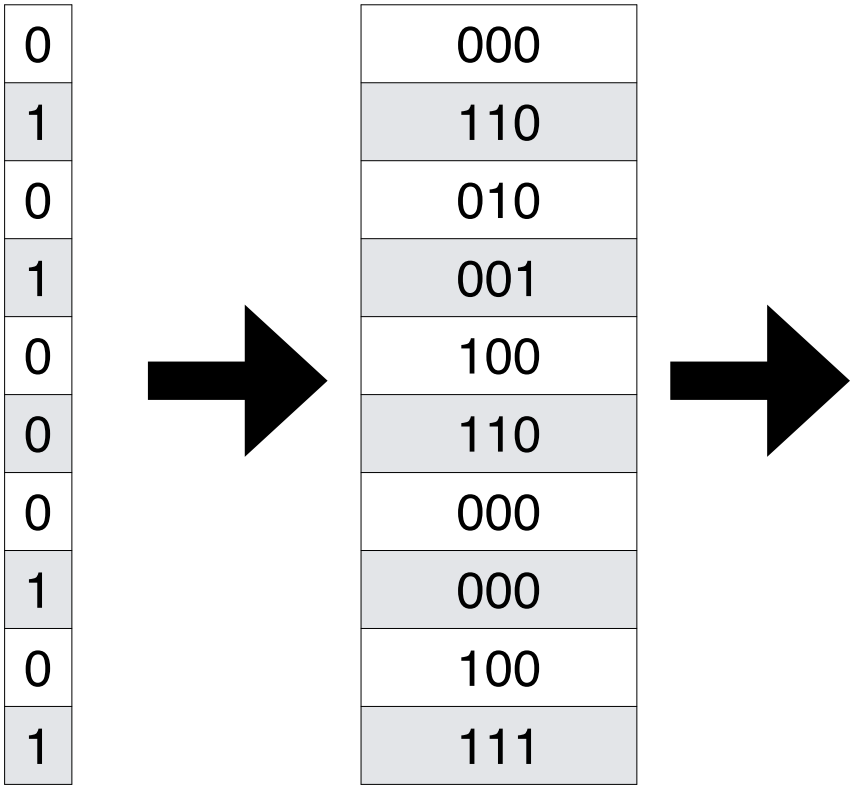
Counting Sketches

A **sketch** is a compact (reduced) representation of a dataset that acts as a replacement for calculations.

Sometimes we need more information than ‘presence/absence’…

3201
946
5581
8945
6145
8126
3887
8925
1246
8324
4549
9100
3887
8499
8970

Count Min Sketch



Count Min Sketch

A **count-min sketch** has k rows, each with its own hash function.

Each bucket is b-bits (allowing 2^b count values)

$h_1(x)$	0	0	0	0	0	0	0	0	0	0
$h_2(x)$	0	0	0	0	0	0	0	0	0	0
$h_3(x)$	0	0	0	0	0	0	0	0	0	0
$h_4(x)$	0	0	0	0	0	0	0	0	0	0
$h_5(x)$	0	0	0	0	0	0	0	0	0	0

Count Min Sketch Insertion

S = { 1, 3, 8, 16}

$h_1(k) = k \% 7$ $h_2(k) = k+3(k\%2) \% 7$ $h_3(k) = |k - 4| \% 7$

h_1	0	0	0	0	0	0	0
h_2	0	0	0	0	0	0	0
h_3	0	0	0	0	0	0	0

Count Min Sketch Find

S = { 1, 3, 8, 16}

$h_1(k) = k \% 7$ $h_2(k) = k+3(k\%2) \% 7$ $h_3(k) = |k - 4| \% 7$

```
_find(16)
```

```
_find(1)
```

h_1	0	2	1	1	0	0	0
h_2	0	1	1	0	1	0	1
h_3	0	1	0	1	1	1	0

Count Min Sketch Find

What is our estimated count of x?

How many **known** collisions?

$h_1(x)$	10	5	13	17	8	1	6
$h_2(x)$	3	7	20	12	2	9	7
$h_3(x)$	12	6	7	5	9	18	3
$h_4(x)$	4	19	26	1	4	5	1
$h_5(x)$	6	6	8	11	6	7	16

Improving Count Min Sketch

Given what we know about collisions here, can we improve our insert strategy?

$h_1(x)$	10	5	13	17	8	1	6
$h_2(x)$	3	7	20	12	2	9	7
$h_3(x)$	12	6	7	5	9	18	3
$h_4(x)$	4	19	26	1	4	5	1
$h_5(x)$	6	6	8	11	6	7	16

Count Min Sketch Improved Insertion

S = { … , 1, 3, 8, 16, …}

$h_1(k) = k \% 7$ $h_2(k) = k+3(k\%2) \% 7$ $h_3(k) = |k - 4| \% 7$

h_1	7	9	10	1	5	7	6
h_2	4	8	7	5	4	10	7
h_3	15	12	2	5	8	2	1

Count Min Sketch Insertion

Minimal Increase: When inserting, only update the minimum count.

Default

h_1	7	**11**	**11**	**2**	5	7	6
h_2	4	**9**	**8**	5	**5**	10	**8**
h_3	15	**13**	2	**6**	**9**	**3**	1

Minimal Increase

h_1	7	9	10	**2**	5	7	6
h_2	4	**9**	7	5	**5**	10	7
h_3	15	12	2	5	**9**	**3**	1

Count-Min Sketch

A probabilistic data structure storing a set of values

Has **four** key properties:

- k, number of hash functions
- n, expected number of insertions
- m, filter size in ***registers***
- b, **number of bits per register**

		$h_{\{1,2,3,\ldots,k\}}$	
0	3	1	0
0	0	2	3
2	0	1	0
3	1	0	1
0	0	2	2

Minimal increase reduces overcounting by identifying collisions.

Count value returned here [underestimates / overestimates / matches] the true count of the query.

Trivia Point 1: Deletion is Possible!

$h_1(x)$	0	1	2	2	3	1	1
$h_2(x)$	3	2	0	0	2	2	1
$h_3(x)$	0	1	1	2	4	2	0
$h_4(x)$	4	1	3	0	2	0	0
$h_5(x)$	1	1	5	1	0	1	1