Algortimos de streaming

Como elegir un número al azar de un stream infinito con memoria constante y otros algoritmos online.

Que es un stream?

$X = <x_1, x_2, x_3, x_4, ..., x_n, ...>$

n desconocido

se lee serialmente

Algoritmo de streaming?

Algoritmos que leen streams y calculan algo usando menos memoria que todo el stream[1].

1: constante o sublineal

Disclaimer

Para que le agarren el gustito, no para que lo sepan.

En criollo

```
class Procesador(object):
    def __init__(self, parametros):
        pass

    def update(self, value):
        pass

    def value(self):
        pass
```

Motivación

- vuela coco
  - algortimos probabilisticos
  - soluciones piolas
  - problemas interesantes
- util
  - base de datos, machine learning, optimizacion, metricas, y mil cosas mas.
  - el procesamiento secuencial de datos es normalmente mucho mas rapido

Promedio

Dado un stream de números mantener el promedio de los números vistos

Promedio

```
class Promedio(Procesador):
    def __init__(self):
        self.n = 0
        self.suma = 0.0

    def update(self, value):
        self.n += 1
        self.suma += value

    def value(self):
        return self.suma / self.n
```

Ojo!

```
>>> sum([1, 1e100, 1, -1e100])
0.0

>>> math.fsum([1, 1e100, 1, -1e100])
2.0
```

Sliding Window

```
class SlidingWindow(Procesador):
    def __init__(self, size):
        self.size = size
        self.window = []

    def update(self, value):
        self.window.append(value)
        if len(self.window) > self.size:
            self.window = self.window[1:]

    def value(self):
        return sum(self.window) / len(self.window)
```

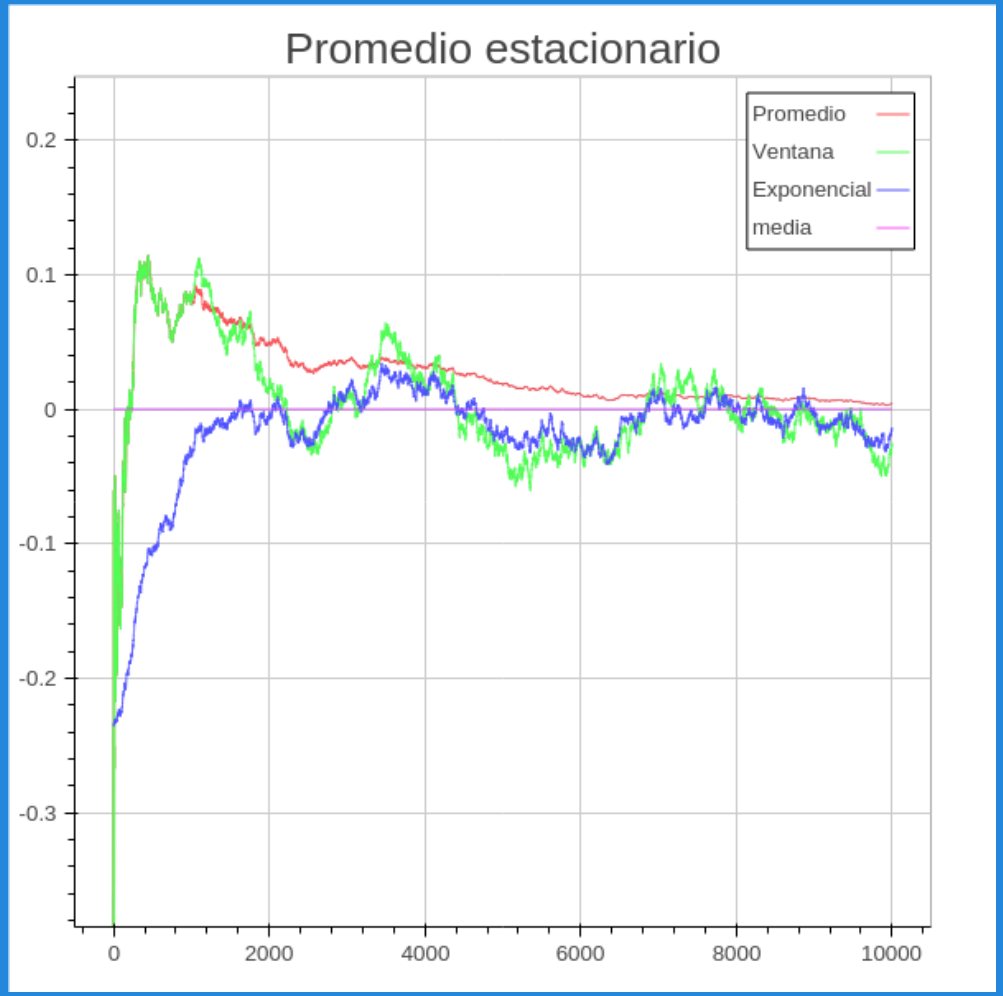
Exponencial

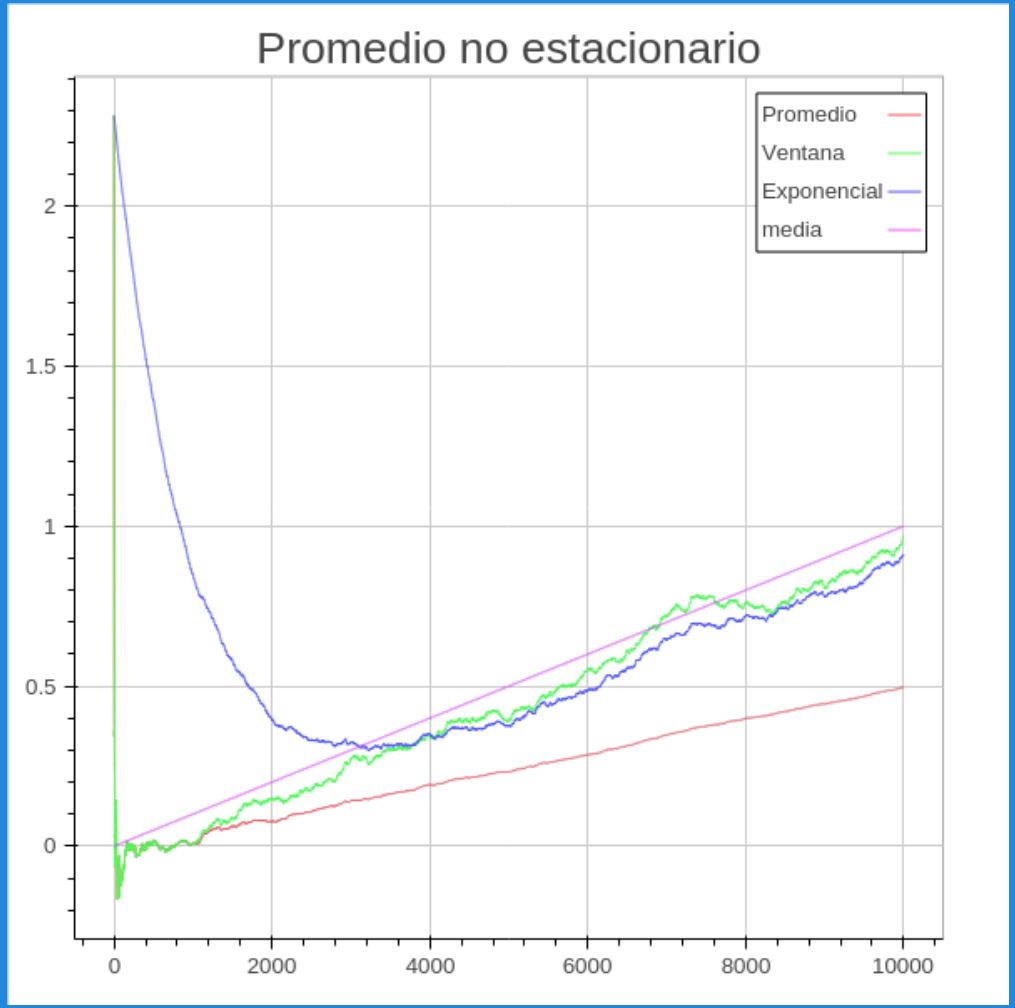
```
class Weighted(Procesador):
    def __init__(self, alpha):
        assert(alpha < 1)
        self.alpha = alpha
        self._value = None

    def update(self, value):
        if self._value is None:
            self._value = value
        else:
            self._value = self._value * (1 - self.alpha) + value * self.alpha

    def value(self):
        return self._value
```

ver http://webdocs.cs.ualberta.ca/~sutton/book/ebook/node19.html


Promedio estacionario
Promedio
Ventana
Exponencial
media
0.2
0.1
0
-0.1
-0.2
-0.3
0
2000
4000
6000
8000
10000

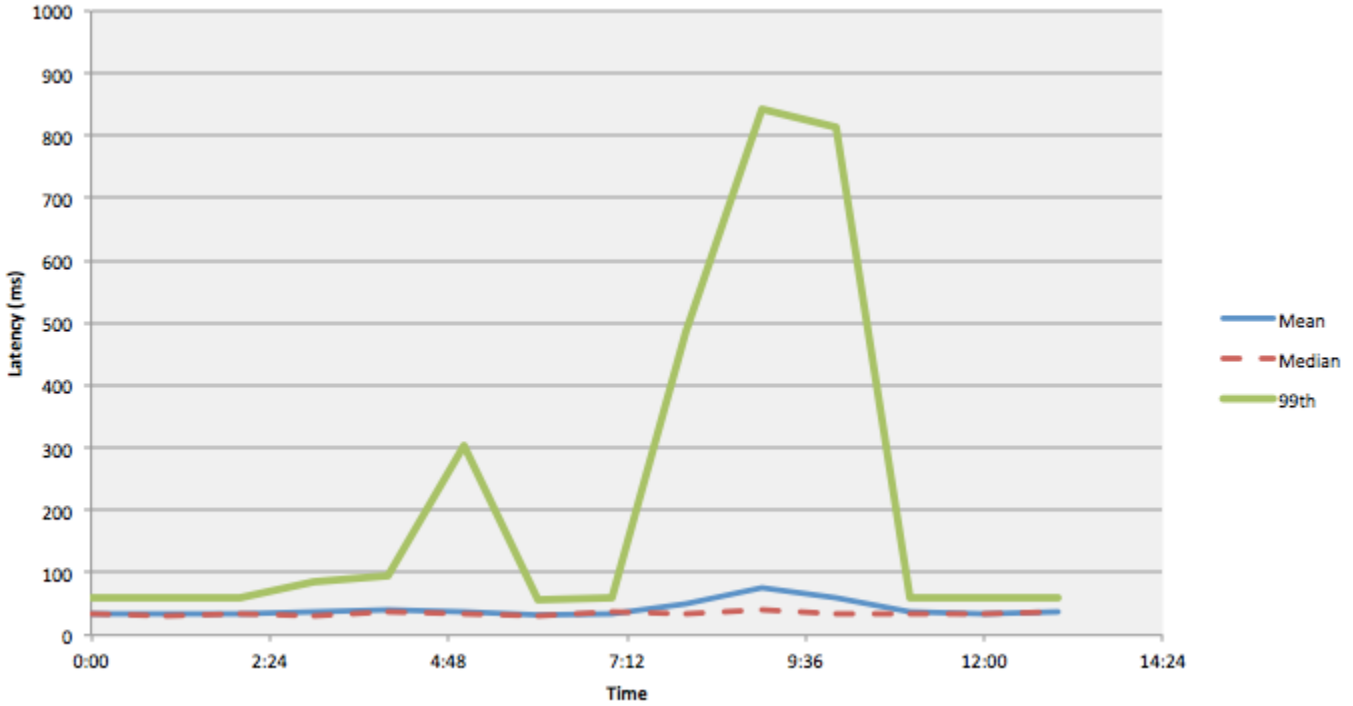

Promedio no estacionario
Promedio
Ventana
Exponencial
media
2
1.5
1
0.5
0
0
2000
4000
6000
8000
10000

Usos

Monitoreo!

(y un millón de cosas más)

Ojo como medimos latencia



Número al azar

Como elegir un elemento al azar de un stream

Número al azar

$\langle x_1 \rangle \rightarrow p(x_1) = 1$

$\langle x_1, x_2 \rangle \rightarrow p(x_i) = ½$

$\langle x_1, x_2, x_3 \rangle \rightarrow p(x_i) = ⅓$

⋮

$\langle x_1, x_2, x_3, ..., x_n \rangle \rightarrow p(x_i) = 1/n$

Inducción

$f(1) \rightarrow x_1$

$f(n) \rightarrow x_n \qquad p = 1/n$

$\qquad\quad f(n-1) \qquad p = 1 - 1/n$

Implementación

```
class EleccionUniforme(Procesador):
    def __init__(self):
        self.n = 0
        self._value = None

    def update(self, value):
        self.n += 1
        # range of random is [0.0, 1.0)
        if random.random() < 1. / self.n:
            self._value = value

    def value(self):
        return self._value
```

Usos

Muestreo!

Se puede extender a tomar múltiples elementos y priorizando los más recientes.

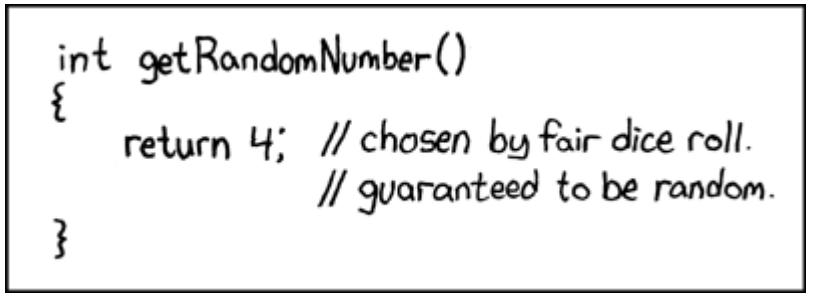
Testing


DILBERT By Scott Adams
TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
www.dilbert.com scottadams@aol.com
NINE NINE NINE NINE NINE NINE
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.
© 2001 United Feature Syndicate, Inc.

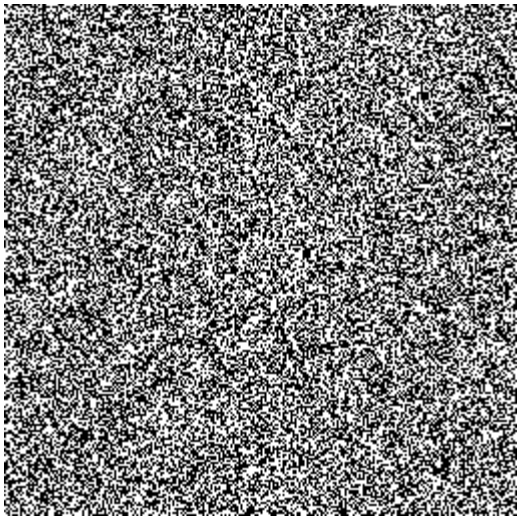
Seed fijo

Prueba que se ejecuta el algoritmo correctamente, no que elige bien al azar

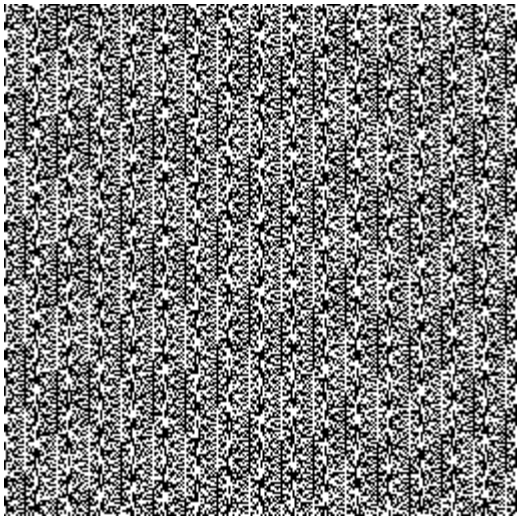
```
def test_uniforme():
    p = EleccionUniforme()
    random.seed(1)
    map(p.update, range(10))
    assert p.value() == 9
```



wat



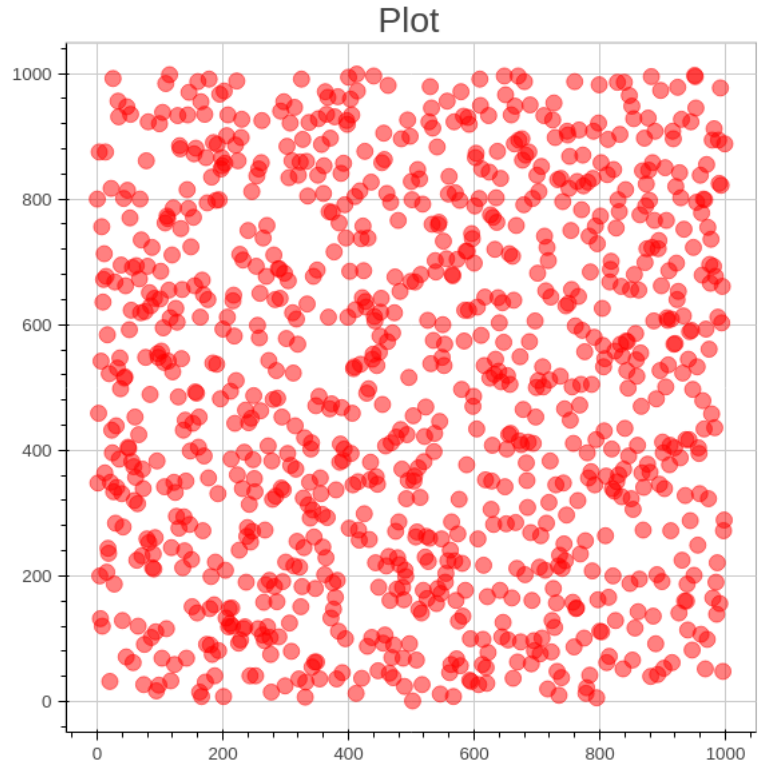
random.org

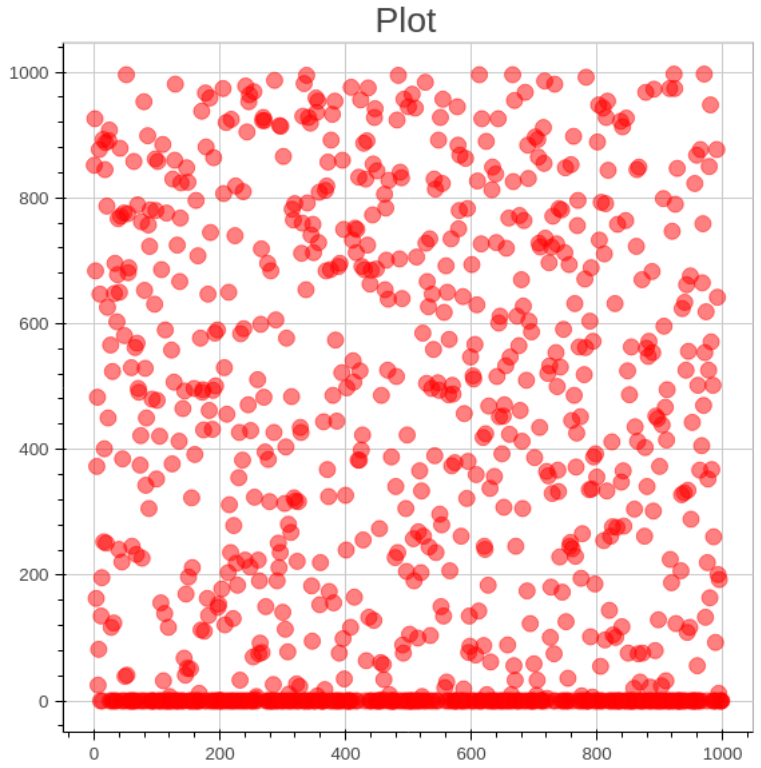


PHP rand() on Windows

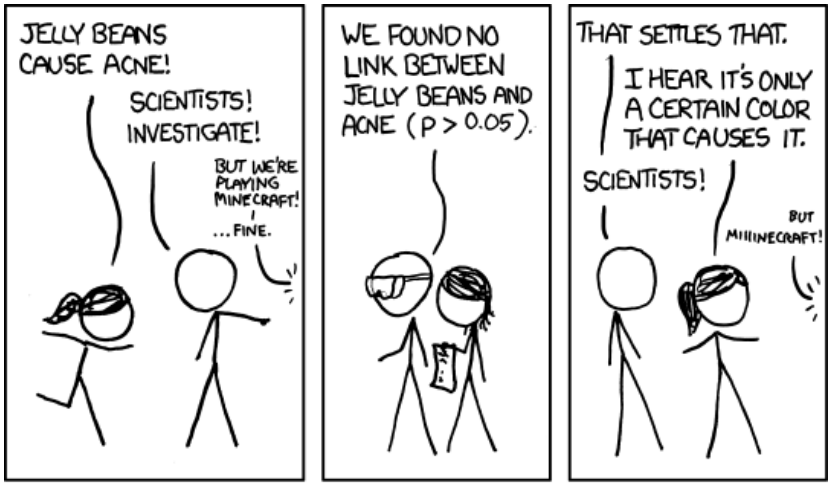
source: http://www.random.org/analysis/

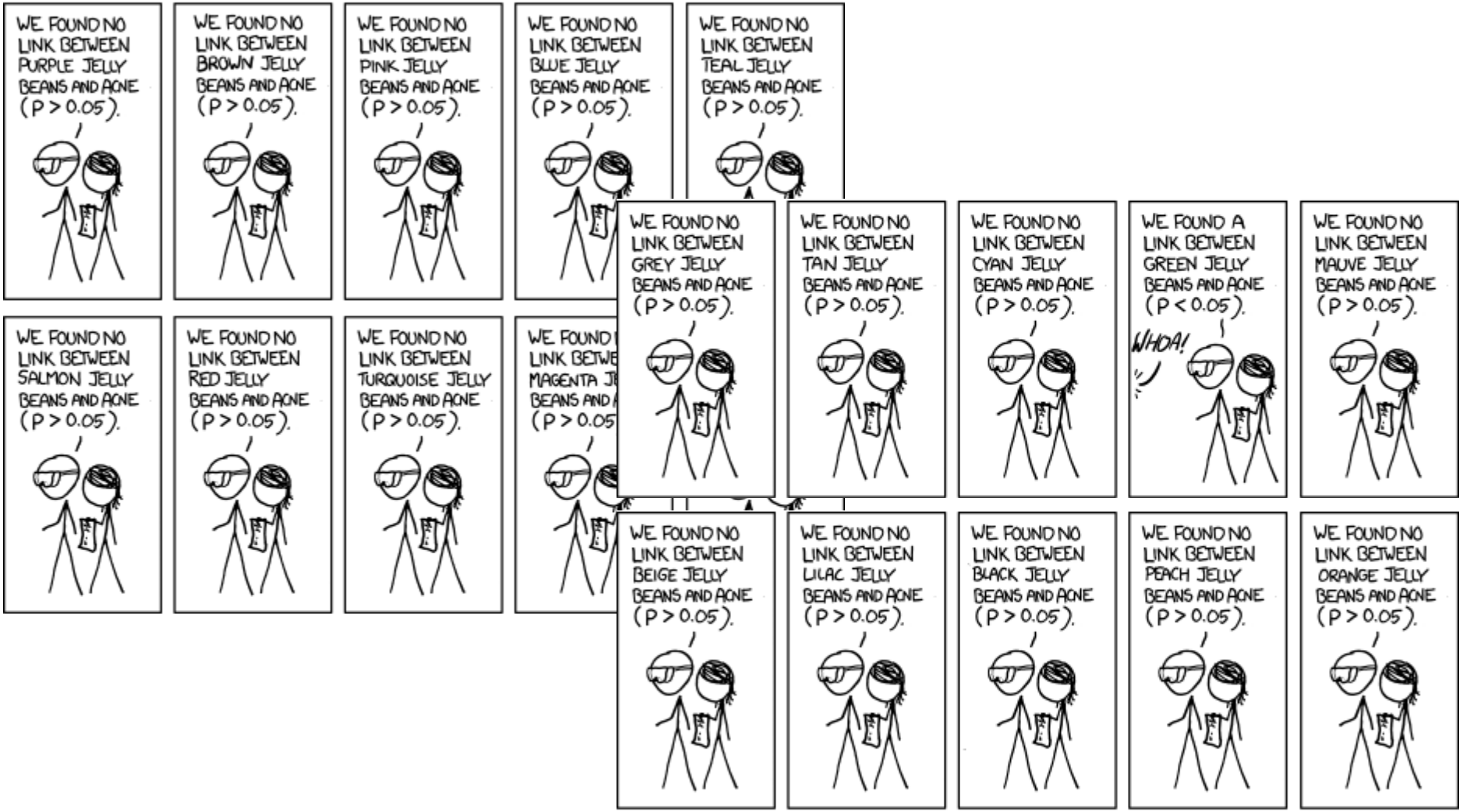
Visual

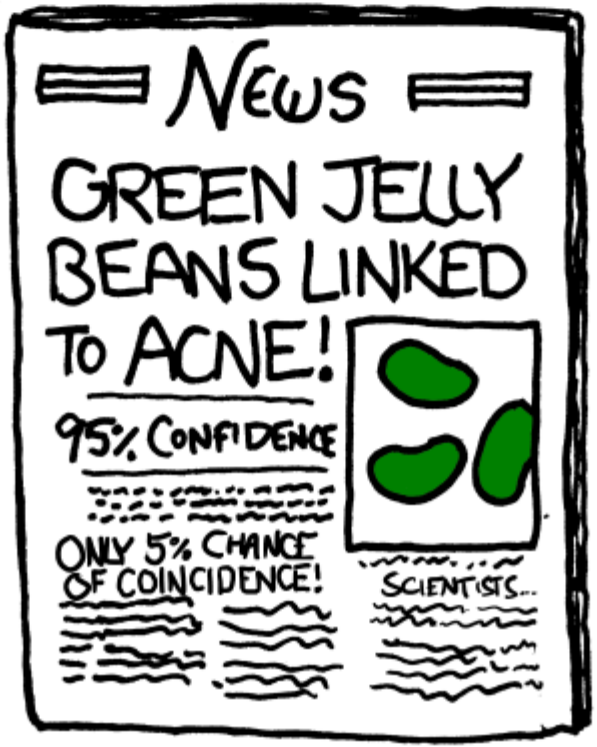

Plot
1000
800
600
400
200
0
0
200
400
600
800
1000


Plot
1000
800
600
400
200
0
0
200
400
600
800
1000

Estadística!




WE FOUND NO LINK BETWEEN PURPLE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BROWN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN PINK JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BLUE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TEAL JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN SALMON JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN RED JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TURQUOISE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN GREY JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TAN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN CYAN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND A LINK BETWEEN GREEN JELLY BEANS AND ACNE (P < 0.05).
WHOA!
WE FOUND NO LINK BETWEEN MAUVE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BEIGE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN LILAC JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BLACK JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN PEACH JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN ORANGE JELLY BEANS AND ACNE (P > 0.05).




```
$ nosetests statistical_tests.py
.............F.....
======================================================================
FAIL: test_fails.test_15
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/usr/lib/python2.7/dist-packages/nose/case.py", line 197, in runTest
    self.test(*self.arg)
  File "statistical_tests.py", line 46, in test_15
    assert(p < 0.05)
AssertionError

----------------------------------------------------------------------
Ran 20 tests in 0.002s

FAILED (failures=1)
```

Sketches

Representamos el stream en espacio menor al que ocupa.

Tradeoff entre espacio y precisión.

Las respuestas suelen tener un $\text{error} < \varepsilon$ con $\text{probabilidad} < 1\text{-}\delta$.

ε y δ suelen ser los parámetros para construir el sketch.

Usos

- Agregacion
  - escalabilidad
  - redes de sensores
- Queries mas interesantes

Count-min Sketch

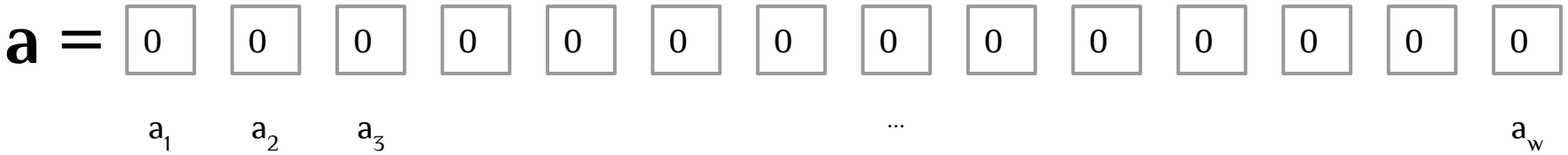
Un sketch que nos permita (entre otras cosas) dado un stream de elementos, estimar cuántas veces vimos cada uno.

stream = <s, t, s, u, s, v, s>

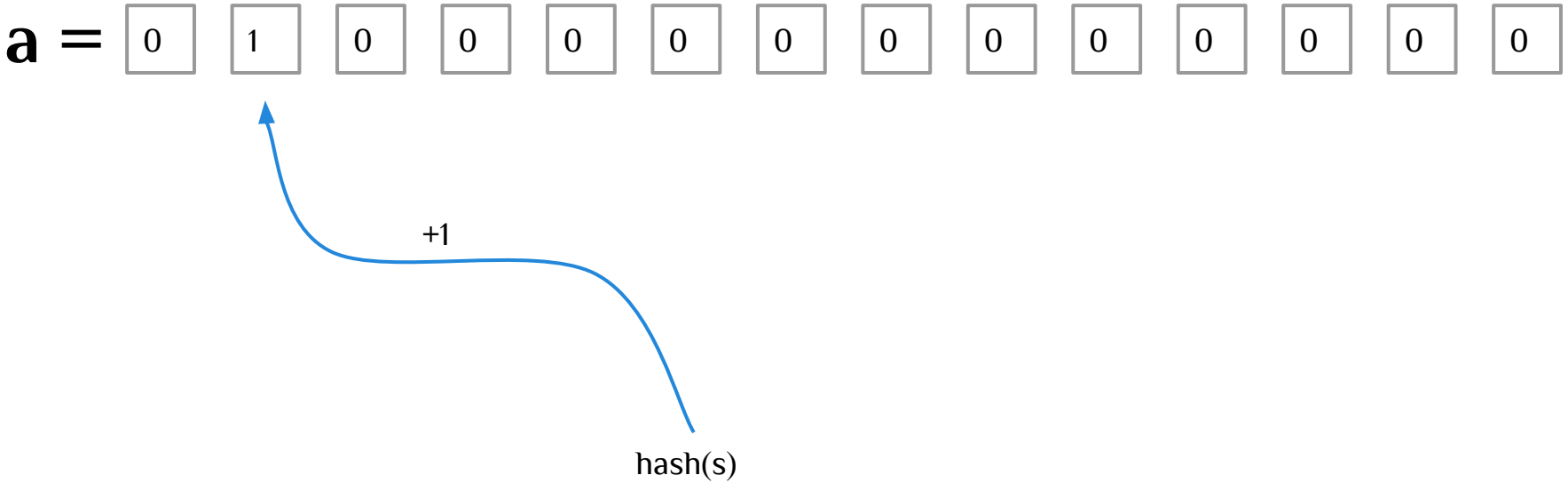
Point(s) = 4

nota: ver disclaimer, simplificado para beneficio del orador

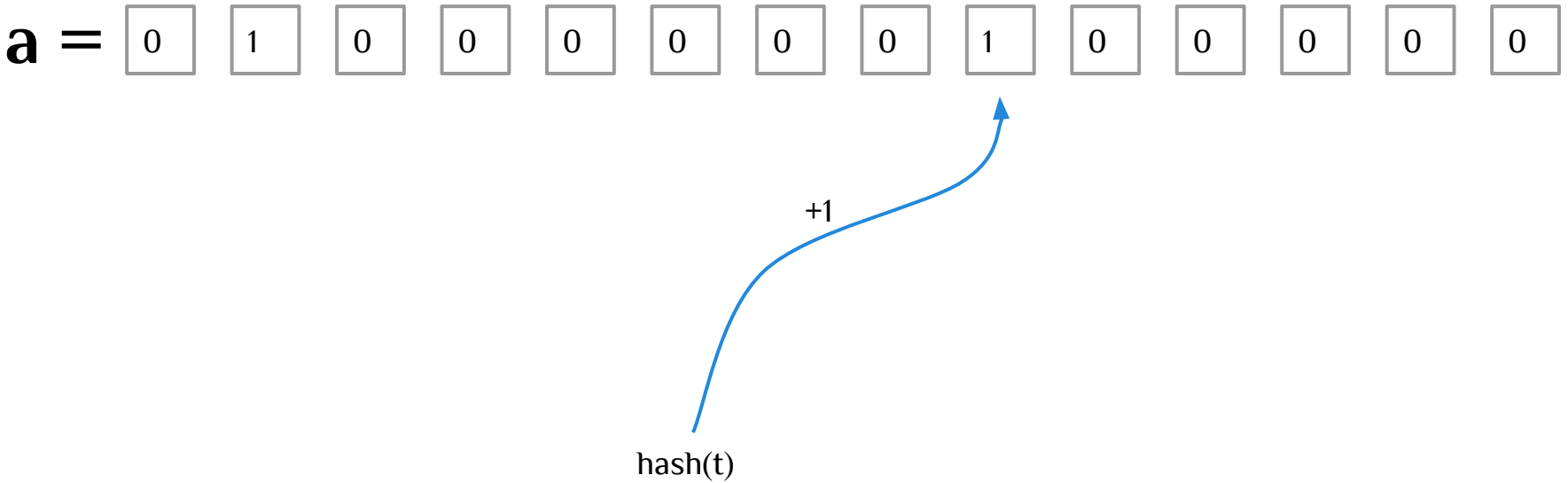
Count-min Sketch



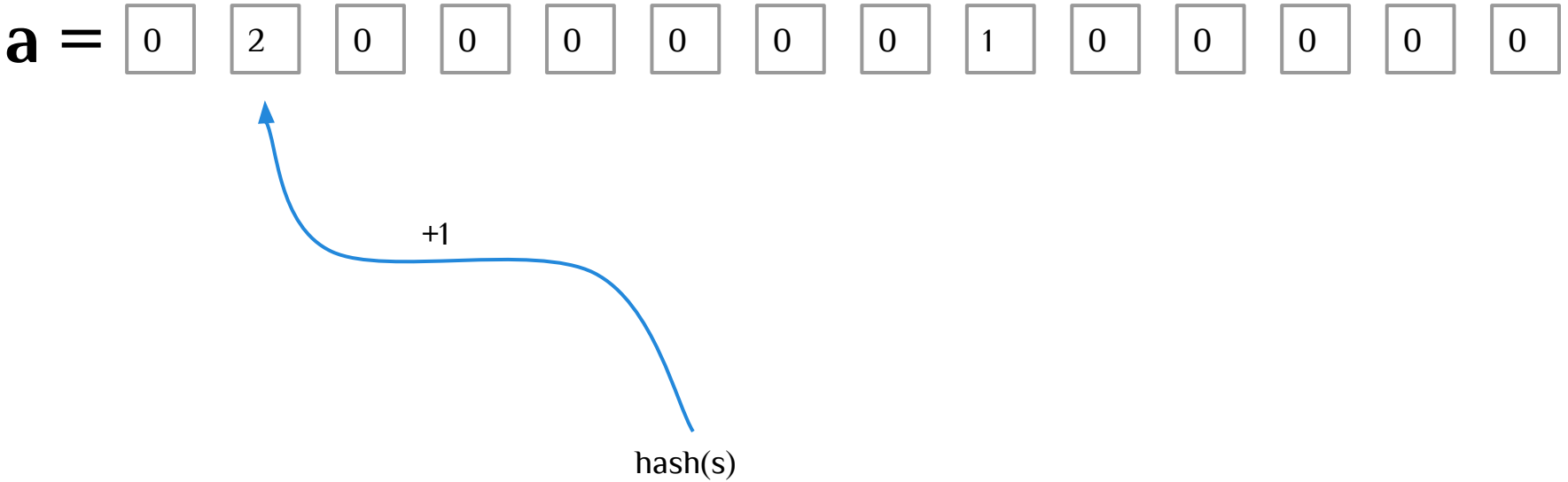
Count-min Sketch



Count-min Sketch



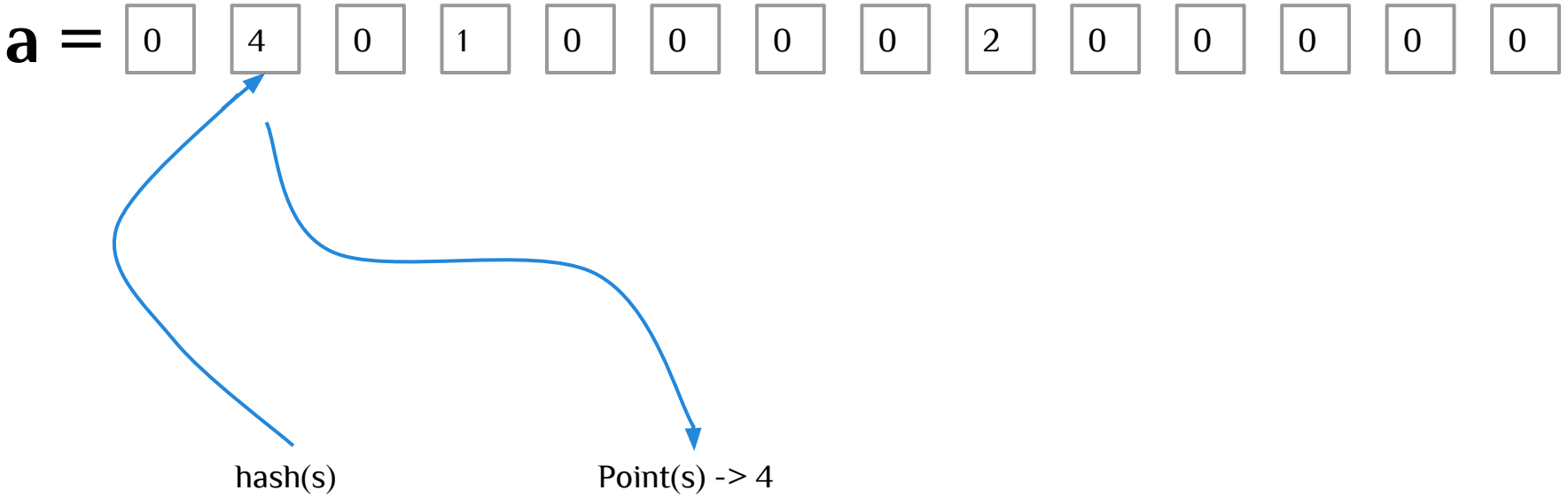
Count-min Sketch



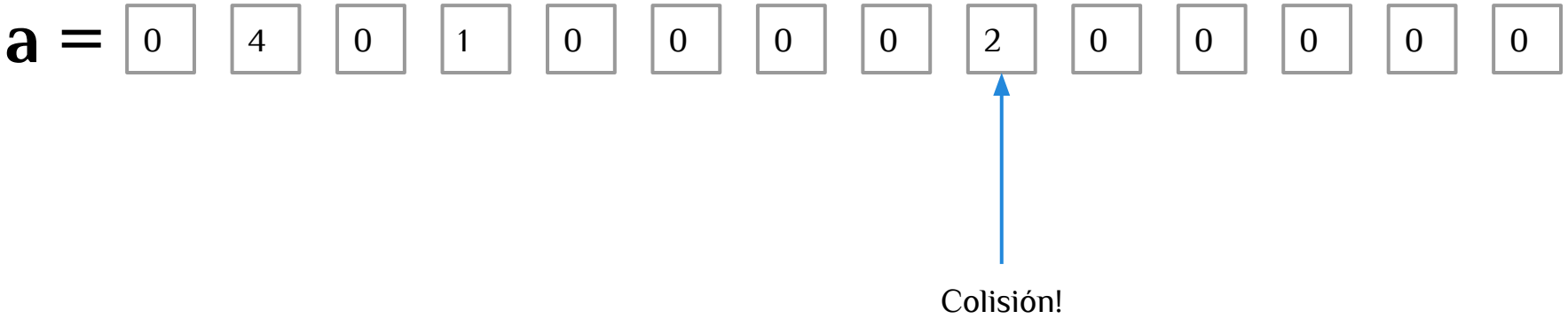
Count-min Sketch

$\mathbf{a} =$ | 0 | 4 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |

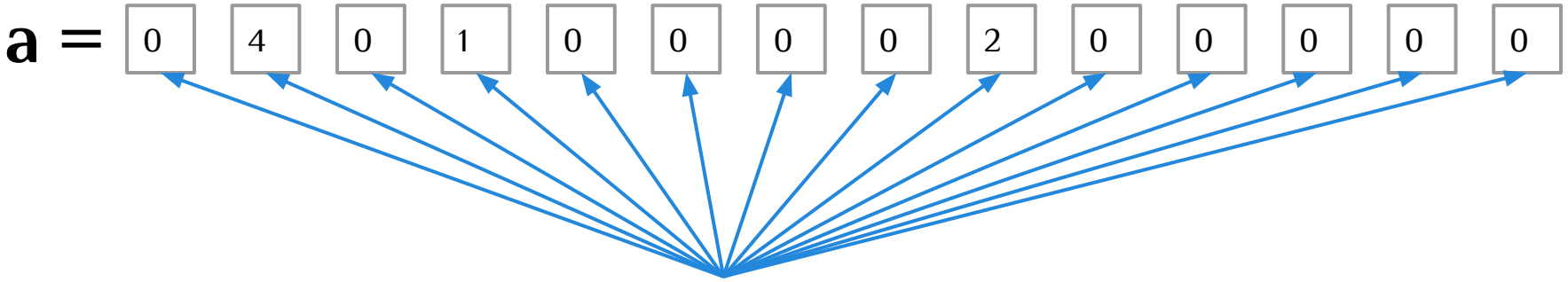
Count-min Sketch



Count-min Sketch



Count-min Sketch

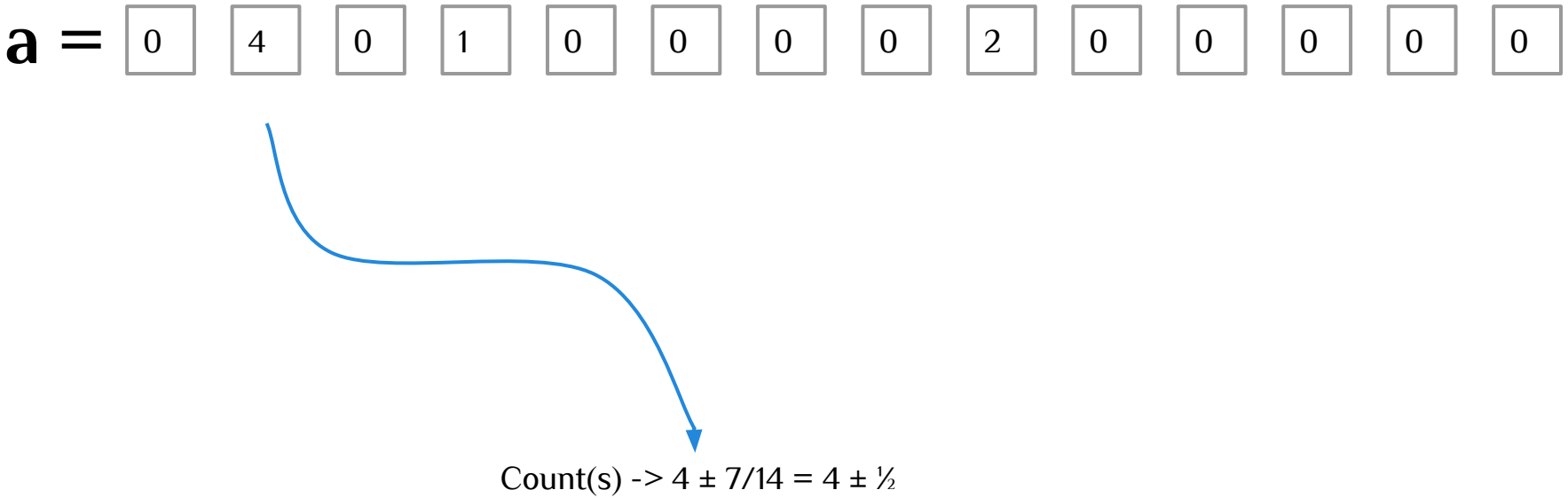


$$E(a_i) = \frac{|stream|}{w}$$

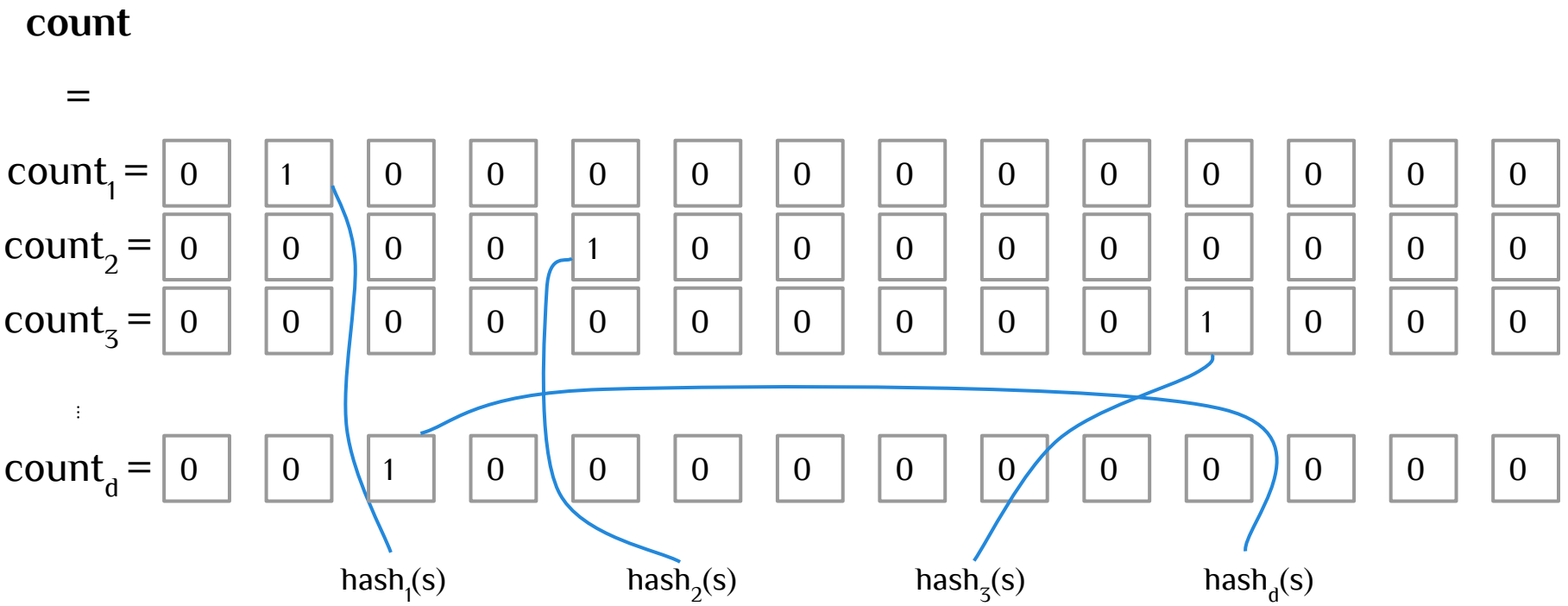
Count(s) <= Point(s) <= Count(s) + e|stream|/w

(con alguna probabilidad)

Count-min Sketch



Mejorando la probabilidad de error



Point(s)

$$\text{Point}(s) = \min_j \text{count}_j[\text{hash}(s)]$$

Elegimos el mejor caso!

Garantias

$$Count(s) <= Point(s)$$

$$Point(s) <= Count(s) + e|stream|/w$$

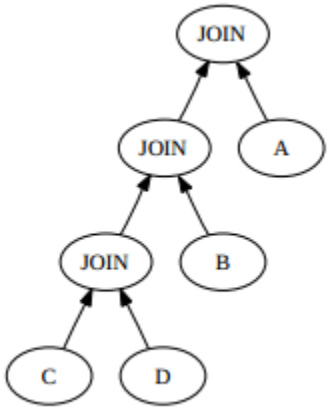
con probabilidad

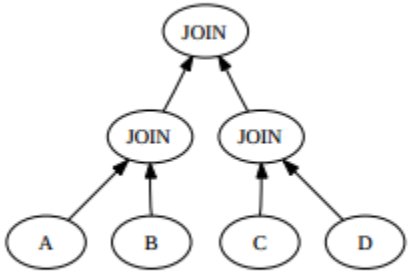
$$1 - e^{-d}$$

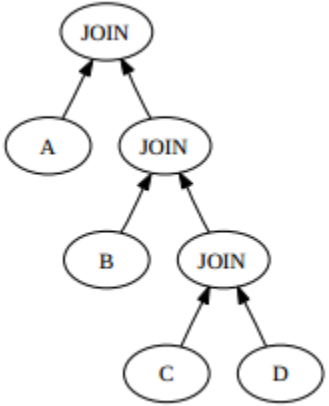
Ejemplos

w	d	size	error	probabilidad
		w * d	**e / w * 100%**	**$1-e^{-d}$**
6	3	18	45%	0.95
272	6	1632	1%	0.995
2719	8	21752	0.1%	0.9995

Usos: Query planning

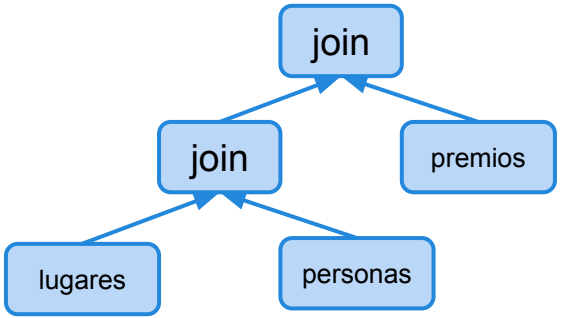


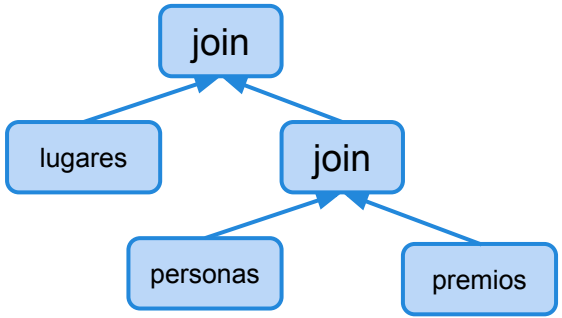




Usos: Query planning

Donde nacieron las personas que ganaron premios nobel?





pg_stats

Name	Type	Description
schemaname	name	Name of schema containing table
tablename	name	Name of table
attname	name	Name of the column described by this row
n_distinct	real	If greater than zero, the estimated number of distinct values in the column. If less than zero, the negative of the number of distinct values divided by the number of rows. (The negated form is used when ANALYZE believes that the number of distinct values is likely to increase as the table grows; the positive form is used when the column seems to have a fixed number of possible values.) For example, -1 indicates a unique column in which the number of distinct values is the same as the number of rows.
most_common_vals	anyarray	A list of the most common values in the column. (Null if no values seem to be more common than any others.)
most_common_freqs	real[]	A list of the frequencies of the most common values, i.e., number of occurrences of each divided by total number of rows. (Null when most_common_vals is.)
histogram_bounds	anyarray	A list of values that divide the column's values into groups of approximately equal population. The values in most_common_vals, if present, are omitted from this histogram calculation. (This column is null if the column data type does not have a < operator or if the most_common_vals list accounts for the entire population.)
correlation	real	Statistical correlation between physical row ordering and logical ordering of the column values. This ranges from -1 to +1. When the value is near -1 or +1, an index scan on the column will be estimated to be cheaper than when it is near zero, due to reduction of random access to the disk. (This column is null if the column data type does not have a < operator.)
most_common_elems	anyarray	A list of non-null element values most often appearing within values of the column. (Null for scalar types.)
most_common_elem_freqs	real[]	A list of the frequencies of the most common element values, i.e., the fraction of rows containing at least one instance of the given value.
elem_count_histogram	real[]	A histogram of the counts of distinct non-null element values within the values of the column, followed by the average number of distinct non-null elements. (Null for scalar types.)

The end.

Preguntas?

lucio.torre@gmail.com
@luciotorre