DATOS MASIVOS I

UNIDAD IV ALGORITMOS PARA FLUJOS DE DATOS

CONTEO

29 de Marzo de 2023

Conteo de Elementos Distintos

Objetivo: contar el número de elementos distintos que han ocurrido en un flujo desde algún punto específico en el tiempo.

Conteo de Elementos Distintos

- Dado un flujo de elementos x_1, x_2, ...,x_n, encontrar el número de elementos distintos n, donde $n = |\{x_1, x_2, ..., x_n\}|$
- También conocido como el problema de estimación de la cardinalidad.
- Ocurren cuando se desea encontrar el número de elementos únicos.

Conteo de Elementos Distintos

Por ejemplo:

- Direcciones IP que pasan a través de un router,
- visitantes a un sitio web,
- secuencias de ADN,
- dispositivos IoT, etc.

Conteo de Elementos Distintos

Ejemplo.

- Dado el flujo a, b, a, c, d, b, d
- Entonces $n = |\{a, b, c, d\}| = 4$

Conteo de Elementos Distintos: Forma de Abordarlo

- Una solución simple es guardar los elementos que vayan llegando en una tabla *hash* o árbol de búsqueda.

Conteo de Elementos Distintos: Forma de Abordarlo

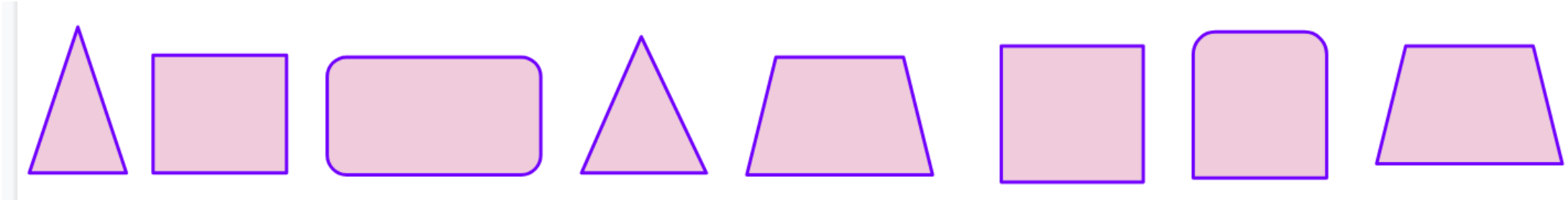
- Cuando el número de elementos distintos es demasiado grande, sería necesario usar múltiples máquinas o guardar la estructura en disco.

Conteo de Elementos Distintos: Forma de Abordarlo

- Una forma más eficiente es estimar el número de elementos distintos, evitando el desborde a través de algoritmos como el de Flajolet-Martin.

Conteo de Elementos Distintos: Forma de Abordarlo

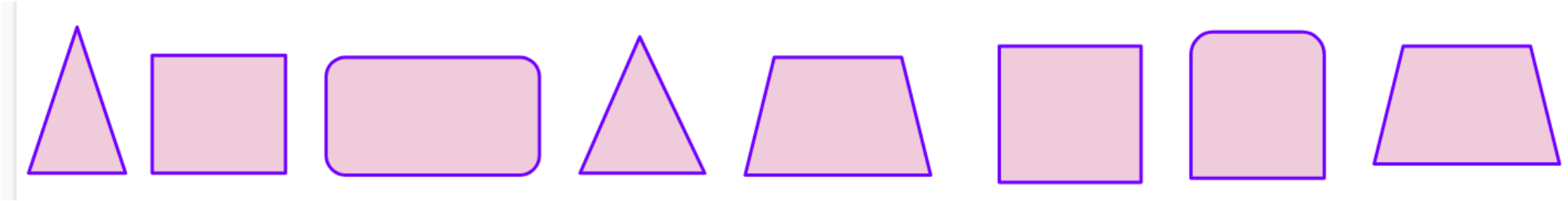
Supongamos que tenemos un flujo de datos con *m* elementos:



¿Cuántos elementos distintos tenemos?

Conteo de Elementos Distintos: Forma de Abordarlo

Supongamos que tenemos un flujo de datos con *m* elementos



Si *m* es pequeña:

- Solución: generar un diccionario.
- Costo de memoria: O(*m*).
- Costo computacional: O(*log(m)*) para almacenamiento y para búsqueda.

Conteo de Elementos Distintos: Forma de Abordarlo

Si *m* es grande:

- Almacenamiento de todos los elementos sería inviable.

Conteo de Elementos Distintos: Forma de Abordarlo

Si *m* es grande:

- Requerimientos de memoria y costo computacional muy altos.
- Sería necesario usar múltiples máquinas o guardar la estructura en disco.

Algoritmo de Flajolet– Martin

- Es un algoritmo para aproximar el número de elementos distintos en un flujo de datos[1].

  - Utiliza múltiples funciones *hash* para mapear los elementos del conjunto universal a una cadena de bits.

[1]Philippe Flajolet and G. Nigel Martin. Probabilistic Counting Algorithms for Data Base Applications, 1985.

Algoritmo de Flajolet– Martin

- Es un algoritmo para aproximar el número de elementos distintos en un flujo de datos[1].

  - La cadena debe ser suficientemente grande como para que haya más posibles valores *hash* que elementos en el conjunto universal.

Algoritmo de Flajolet– Martin

- Es un algoritmo para aproximar el número de elementos distintos en un flujo de datos[1].

  - Intuición: entre más elementos distintos haya en el flujo, mayor número de valores *hash* distintos deberían ocurrir y es más probable que uno de estos valores sea inusual (que termine en muchos ceros).

Algoritmo de Flajolet– Martin

- La propiedad de uniformidad de la función hash permite estimar que la mitad de los valores de los elementos terminarán en 0, una cuarta parte de los elementos terminarán en 00, una octava parte terminarán en 000 y en general $\frac{1}{2^k}$ terminarán en *k* ceros.

Algoritmo de Flajolet– Martin

	Number	Binary Repr
Uniform Distribution	0	000
	1	001
	2	010
	3	011
	4	100
	5	101
	6	110
	7	111

$$Pr[\rho = 0] = \frac{1}{2}$$

$$Pr[\rho = 1] = \frac{1}{4}$$

$$Pr[\rho = 2] = \frac{1}{8}$$

Algoritmo de Flajolet– Martin

- Por lo tanto, si una función hash genera un valor que termina en *k* ceros, una estimación del número de elementos distintos es $\frac{2^{k}}{\phi}$.

Algoritmo de Flajolet– Martin

- Por lo tanto, si una función hash genera un valor que termina en k ceros, una estimación del número de elementos distintos es $\frac{2^{k}}{\phi}$

- Donde $\phi \approx 0.77351$ es un factor de corrección.

Algoritmo de Flajolet– Martin

- Se usan múltiples funciones *hash* para obtener varias estimaciones, las cuales se combinan.

Combinando Estimaciones

- Se pueden promediar estimaciones de múltiples funciones *hash.*
- Pero la estimación es muy sensible a valores atípicos.

Combinando Estimaciones

- Una mejor estrategia es unir las estimaciones.
- Obtener la mediana de cada grupo (estimación).
- Posteriormente promediar las medianas.

Procedimiento General

- Sea *R* el número de *0s* al final de la cadena correspondiente al valor *hash* de algún elemento.
- Se inicializa un arreglo de bits de tamaño *L* con ceros y se pone a 1 el bit de la posición *R* de cada elemento del flujo.

Ejemplo de Estimación del Número de Elementos Distintos

x	h(x)	Rem	Binary	r(a)
1	7	2	00010	1
3	19	4	00100	2
2	13	3	00011	0
1	7	2	00010	1
2	13	3	00011	0
3	19	4	00100	2
4	25	0	00000	5
3	19	4	00100	2
1	7	2	00010	1
2	13	3	00011	0
3	19	4	00100	2
1	7	2	00010	1

Ejemplo de Estimación del Número de Elementos Distintos

$S = 1, 3, 2, 1, 2, 3, 4, 3, 1, 2, 3, 1$

$h(x) = (6x + 1) \ mod \ 5$

R = max(r(a)) = 5

No. of distinct elements = $2^R = 2^5$ = 32 / 0.77351 = 41.369

x	h(x)	Rem	Binary	r(a)
1	7	2	00010	1
3	19	4	00100	2
2	13	3	00011	0
1	7	2	00010	1
2	13	3	00011	0
3	19	4	00100	2
4	25	0	00000	5
3	19	4	00100	2
1	7	2	00010	1
2	13	3	00011	0
3	19	4	00100	2
1	7	2	00010	1

Ejercicio de Estimación del Número de Elementos Distintos

Ejercicio.

Dado el flujo 3, 1, 4, 1, 5, 9, 2, 6, 5 y las funciones *hash* $h_1(x) = (2 \cdot x+1)$ **mod** 32, $h_2(x) = (3 \cdot x+7)$ **mod** 32 y $h_3(x) = 4 \cdot x$ **mod** 32.

Determina el número de *0s* en los que termina el valor de cada elemento y estima el número de elementos distintos usando 5 bits.

Ejercicio de Estimación del Número de Elementos Distintos

$Z = 2$ $Z = 3$ $Z = 2$

The estimate for F_0 is $\left\lfloor \frac{2^{(2+3+2)/3}}{.77351} \right\rfloor = 6.$

Procedimiento General

- Sea *r* la primera posición del arreglo de bits cuyo valor es cero.

- Y el estimador del número de elementos distintos en el flujo de datos es $\frac{2^k}{\phi}$, donde $\phi \approx 0.77351$ es un factor de corrección.

Análisis de las Estimaciones

- La probabilidad de que un elemento (dato) tenga un valor hash que termine en al menos *r* *0s* es 2^{-r}

Análisis de las Estimaciones

- Si tenemos *m* elementos distintos en el flujo, la probabilidad de que ninguno tenga (al menos) *r* *0s* es $(1-2^{-r})^m = ((1-2^{-r})^{2^r})^{m \cdot 2^{-r}}$
- Cuando *r* es suficientemente grande este valor se aproxima a $(1-\epsilon)^{1/\epsilon} = \frac{1}{e}$

Análisis de las Estimaciones

La probabilidad de que ninguno de los valores *hash* de los *m* elementos distintos termine en al menos *r 0s* es $e^{-m \cdot 2^{-r}}$ por lo que:

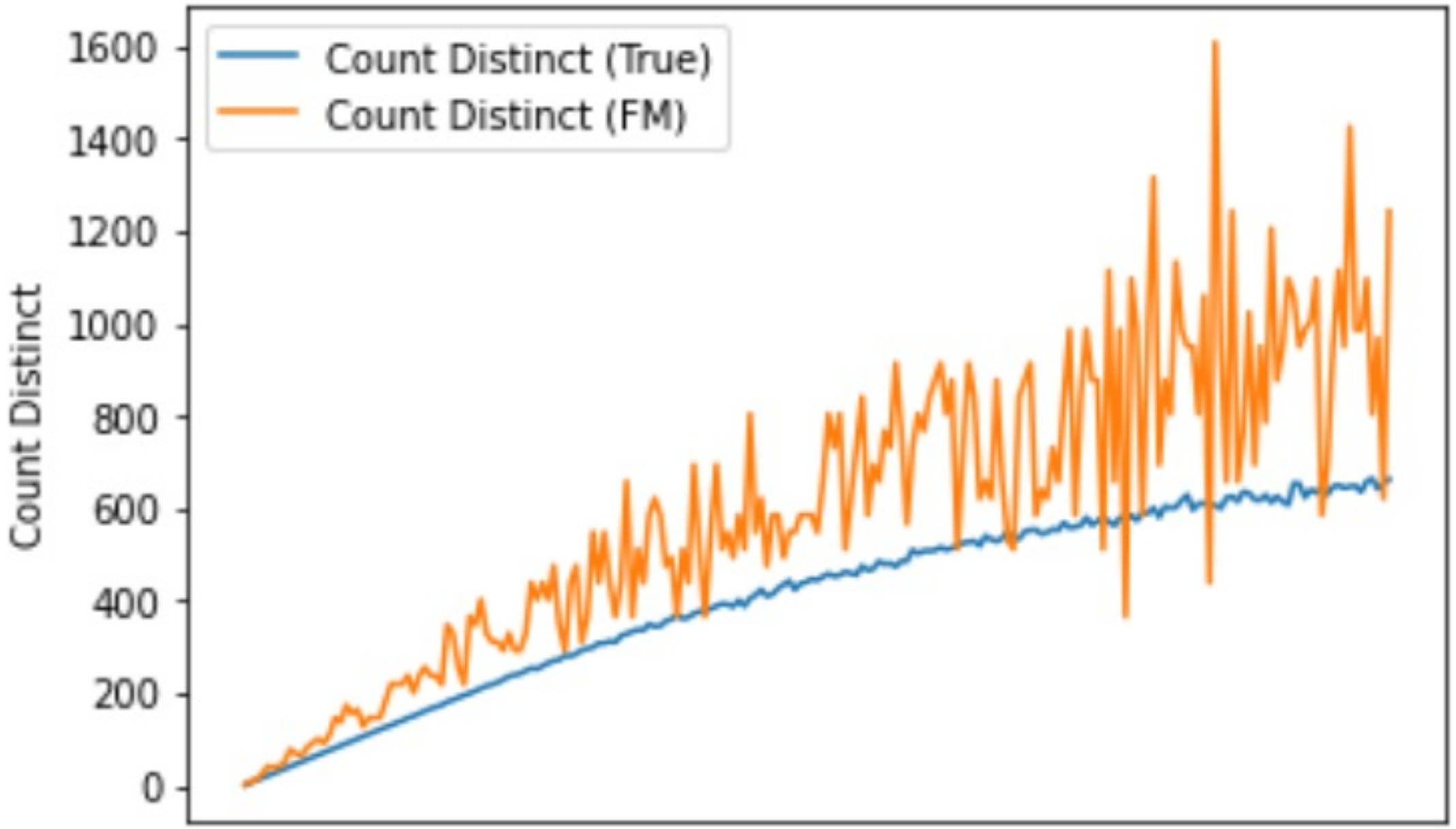
1. Si *m* es mucho más grande que 2^r, la probabilidad de que alguno de los valores termine en al menos *r 0s* se aproxima a *1*.

Análisis de las Estimaciones

La probabilidad de que ninguno de los valores *hash* de los *m* elementos distintos termine en al menos *r 0s* es $e^{-m \cdot 2^{-r}}$ por lo que:

2. Si *m* es mucho más pequeño que 2^r, la probabilidad de que alguno de los valores termine en al menos *r 0s* se aproxima a *0.*

Ejercicio de Estimación del Número de Elementos Distintos



Ventaja y Desventaja

- Usa menos cantidad de memoria para aproximar el número de elementos únicos.

- Una de las desventajas del algoritmo de Flajolet – Martin es la suposición de la generación de claves *hash* totalmente aleatorias y uniformes.