

ANÁLISIS DE DATOS MASIVOS

CONTEO

Blanca Vázquez

1 de abril de 2025

CONTANDO ELEMENTOS DISTINTOS (1)

- Objetivo: contar el número de elementos distintos que han ocurrido en un flujo desde algún punto específico en el tiempo
 - Dado un flujo de elementos x_1, x_2, \dots, x_n , encontrar el número de elementos distintos n , donde $n = |\{x_1, x_2, \dots, x_n\}|$
 - También conocido como el problema de estimación de la cardinalidad
- Ocurren cuando se desea encontrar el número de elementos únicos
 - Direcciones IP que pasan a través de un router, visitantes a un sitio web, secuencias de ADN, dispositivos IoT, etc.
- Ejemplo
 - Dado el flujo a, b, a, c, d, b, d , entonces $n = |\{a, b, c, d\}| = 4$

CONTANDO ELEMENTOS DISTINTOS (2)

- Una solución simple es guardar los elementos que vayan llegando en una tabla *hash* o árbol de búsqueda.
- Cuando el número de elementos distintos es demasiado grande, sería necesario usar múltiples máquinas o guardar la estructura en disco.
- Una forma más eficiente es estimar el número de elementos distintos, a través de algoritmos como el de Flajolet-Martin.

PROBLEMA DEL CONTEO DE ELEMENTOS DISTINTOS

- Encontrar el número de elementos distintos en un flujo de datos con elementos repetidos
- También conocido como problema de estimación de la cardinalidad

Dado un stream s de elementos x_1, x_2, \dots, x_n , encontrar el número de elementos distintos n , donde $n = |\{x_1, x_2, \dots, x_n\}|$

CONTANDO ELEMENTOS DISTINTOS (2)

- Supongamos que tenemos un flujo de datos con m elementos



¿Cuántos elementos distintos tenemos?

CONTANDO ELEMENTOS DISTINTOS (3)

- Supongamos que tenemos un flujo de datos con m elementos



¿Cuántos elementos distintos tenemos?

- Si m es pequeña:
 - Solución: generar un diccionario
 - Memoria: $O(m)$
 - Costo computacional: $O(\log(m))$ para almacenamiento y para búsqueda

CONTANDO ELEMENTOS DISTINTOS (4)

- Supongamos que tenemos un flujo de datos con m elementos



¿Cuántos elementos distintos tenemos?

- Si m es grande:
 - Almacenamiento de todos los elementos sería inviable
 - Requerimientos de memoria y costo computacional muy altos
 - Sería necesario usar múltiples máquinas o guardar la estructura en disco.

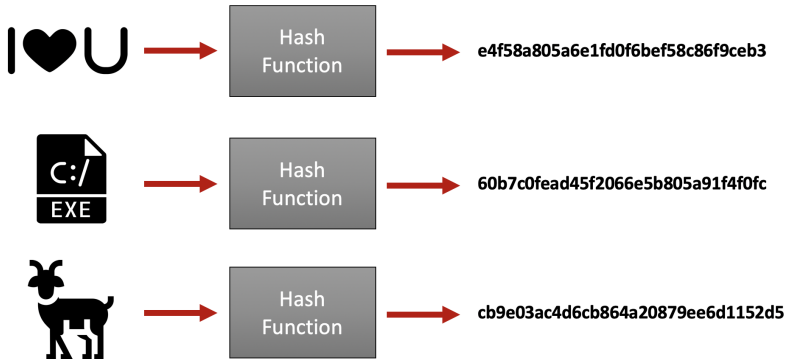
ALGORITMO DE FLAJOLET–MARTIN (1)

- Es un algoritmo para aproximar el número de elementos distintos en un flujo de datos¹
 - Utiliza múltiples funciones *hash* para mapear los elementos del conjunto universal a una cadena de bits
 - La cadena debe ser suficientemente grande como para que haya más posibles valores *hash* que elementos en el conjunto universal
 - Entre más elementos distintos haya en el flujo, mayor número de valores *hash* distintos deberían ocurrir.

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

INTUICIÓN: ALGORITMO DE FLAJOLET–MARTIN

- Transformar los elementos de entrada sobre una función hash binaria con distribución uniforme e independiente de probabilidad.



ALGORITMO DE FLAJOLET–MARTIN (2)

- 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.
- Por lo tanto, si una función *hash* genera un valor que termina en k ceros, el conjunto universal tiene cerca de 2^k elementos únicos.
- Se usan múltiples funciones hash para obtener varias estimaciones.

- La idea general del algoritmo de Flajolet-Martin es que cuantos más elementos diferentes veamos en el flujo de datos, más valores hash diferentes tendremos.
- A medida que observemos valores hash más diferentes, es más probable que uno de estos valores sea inusual.
- Un valor inusual será aquel que termine en muchos ceros.

ANÁLISIS DE LAS ESTIMACIONES (1)

- Sea R el número de 0s al final de la cadena correspondiente al valor hash de algún elemento, se mantiene el R más grande de los elementos del flujo y se toma 2^R como un estimador del número de elementos distintos
- La probabilidad de que un elemento (dato) tenga un valor hash que termine en al menos r 0s es 2^{-r}
- 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 $(1 - \epsilon)^{1/\epsilon} = \frac{1}{e}$

- 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
 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

- 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