Datos Masivos I

4. Algoritmos para flujos de datos

Profa. Alondra Berzunza

4. Algoritmos para flujos de datos

4.1 El modelo de datos en flujo

4.2 Muestreo

4.3 Filtrado

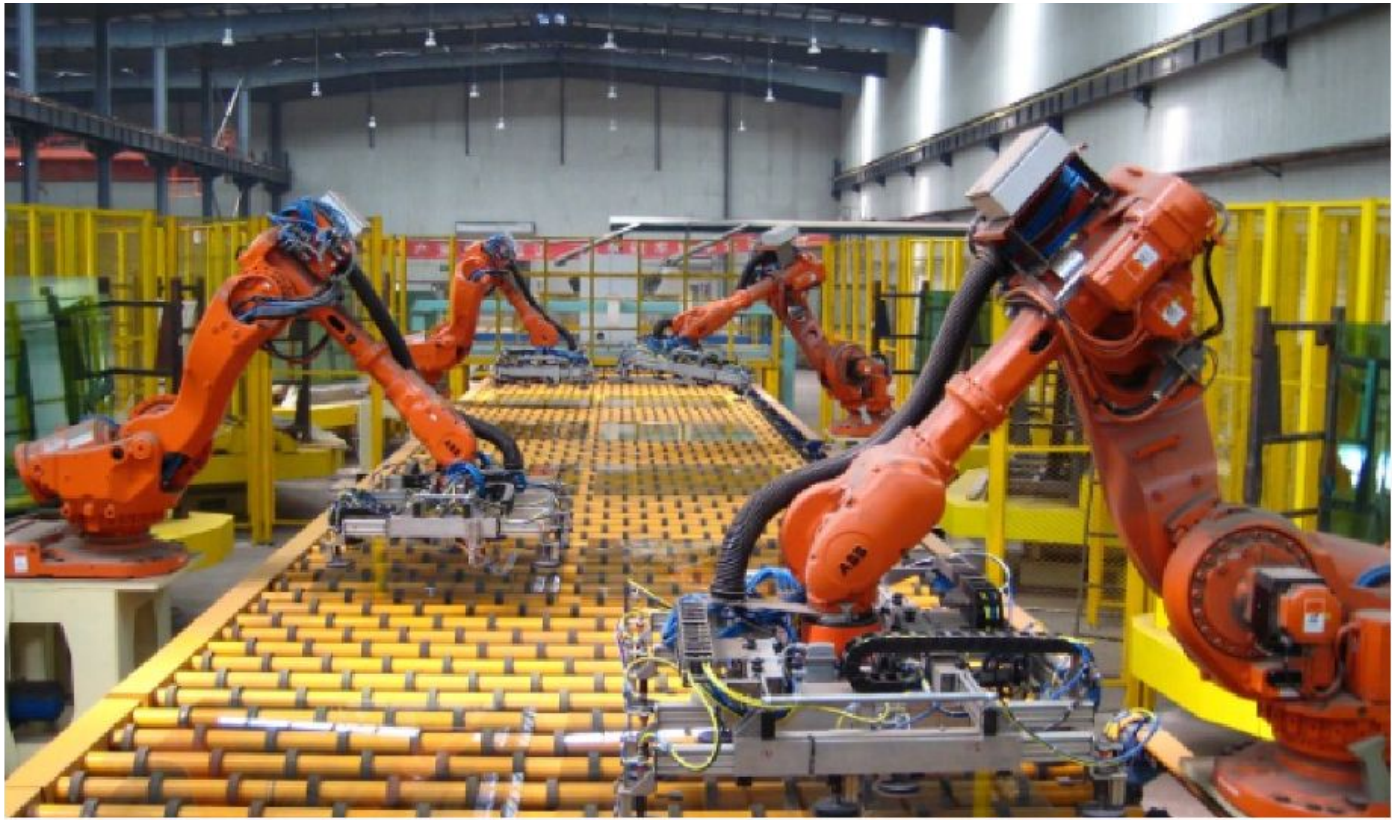
4.4 Conteo

4.5 Estimación de momentos

4.6 Ventanas deslizantes

Flujos de Datos

Flujos de Datos

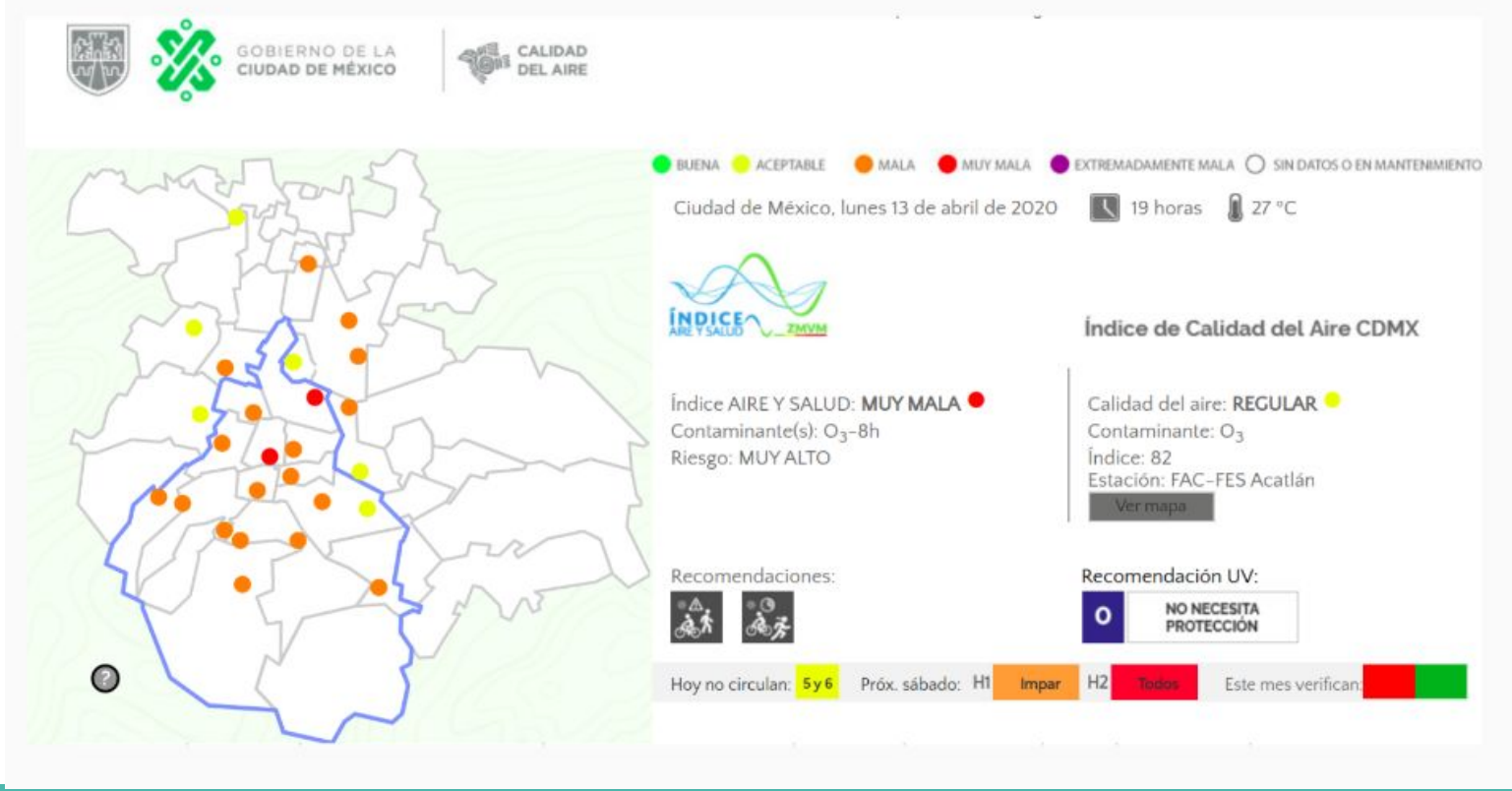


Los sensores industriales pueden capturar grandes cantidades de datos

Imagen tomada de commons.wikimedia.org

Flujos de Datos

Estaciones de monitoreo de la calidad del aire



Flujos de Datos

Estaciones de monitoreo de la calidad del aire

O_3

Fecha de consulta: 2020-04-13

Unidad de Medida: Partes por Millón (ppm)

Hora	AJM	ATI	BJU	CAM	CCA	CHO	CUA	CUT	FAC	FAR	GAM	HGM	IZT	LLA	LPR	MER	MGH	NEZ	PED	SAC	SAG	SFE	TAH	TLA	TLI	UIZ	UAX	VIF	XAL
1																													
2																													
3																													
4	0.043	0.017	0.014	0.010	0.018			0.002	0.019	0.026	0.016		0.013	0.005	0.014	0.004	0.020	0.023	0.025	0.033	0.024	0.039	0.020	0.016		0.016	0.023	0.004	
5	0.042	0.007	0.012	0.004	0.018			0.002	0.014	0.021	0.005		0.010	0.003	0.010	0.005	0.016	0.018	0.022	0.018	0.014	0.035	0.009	0.008		0.005		0.003	
6	0.038	0.001	0.008	0.005	0.007			0.002	0.006	0.011	0.004		0.003	0.003	0.001	0.004	0.004	0.014	0.010	0.007	0.007	0.022	0.016	0.002		0.004	0.012	0.002	
7	0.033	0.000	0.005	0.004	0.004		0.035	0.002	0.003	0.009	0.004	0.008	0.002	0.003	0.004	0.003	0.002	0.005	0.010	0.003	0.006	0.004	0.011	0.003		0.005	0.007	0.003	
8	0.035	0.001	0.006	0.007	0.011		0.034	0.002	0.005	0.011	0.013	0.014	0.008	0.004	0.007	0.008	0.008	0.004	0.016	0.005	0.006	0.007	0.011	0.004		0.009	0.008	0.010	
9	0.041	0.020	0.021	0.021	0.029		0.040	0.010	0.010	0.028	0.027	0.022	0.019	0.013	0.017	0.016	0.021	0.015	0.020	0.014	0.016	0.034	0.026	0.012		0.016	0.027	0.022	
10	0.049	0.031	0.038	0.034	0.050		0.042	0.035	0.022	0.042	0.043	0.035	0.037	0.035	0.022	0.036	0.034	0.034	0.046	0.035	0.027	0.044	0.050	0.026		0.037	0.043	0.044	
11	0.064	0.039	0.052	0.049	0.064		0.054	0.050	0.038	0.058	0.063	0.060	0.050	0.040	0.036	0.060	0.053	0.051	0.060	0.060	0.047	0.056	0.066	0.037		0.060	0.058	0.057	
12	0.079	0.046	0.061	0.062	0.077		0.070	0.058	0.047	0.068	0.076	0.073	0.062	0.048	0.056	0.075	0.069	0.057	0.081	0.066	0.050	0.072	0.077	0.051		0.066	0.068	0.057	
13	0.093	0.052	0.080	0.081	0.093		0.090	0.060	0.066	0.081	0.095	0.089	0.080	0.066	0.076	0.093	0.083	0.063	0.094	0.072	0.065	0.092	0.086	0.059		0.072	0.073	0.055	
14	0.097	0.060	0.086	0.097	0.100		0.099	0.062	0.078	0.093	0.107	0.110	0.091	0.082	0.064	0.103	0.107	0.083	0.103	0.095	0.081	0.118	0.077	0.078		0.095	0.089	0.065	
15	0.100	0.069	0.090	0.116	0.107		0.057	0.069	0.092	0.104	0.114	0.112	0.101	0.088	0.080	0.112	0.116	0.090	0.106	0.074	0.094	0.098	0.053	0.086		0.107	0.099	0.074	
16		0.072	0.094	0.121	0.105		0.073	0.085	0.073	0.111	0.117	0.113	0.100	0.098	0.094	0.112	0.110	0.063	0.100	0.063		0.072	0.054	0.099		0.087	0.096	0.087	
17	0.076	0.068	0.079	0.116	0.084		0.073	0.086	0.071	0.077	0.106	0.101	0.075	0.098	0.086	0.099	0.085	0.060	0.077	0.062	0.093	0.073	0.068	0.092		0.074	0.071	0.092	
18	0.069	0.059	0.062	0.082	0.073		0.061	0.062	0.086	0.059	0.067		0.064	0.062	0.056	0.071	0.069	0.055	0.065	0.063	0.052	0.062	0.079	0.077		0.071	0.070	0.080	
19	0.065	0.036	0.051	0.065	0.065		0.051	0.052	0.050	0.064	0.059		0.060	0.051	0.050	0.063	0.049	0.057	0.057	0.068	0.055	0.056	0.072	0.052		0.065	0.065	0.052	

Consulta el índice por zonas

Interpretación del Índice AIRE Y SALUD	
Concentraciones	Condición
0-0.051	Buena
>0.051 y 0.095	Aceptable
>0.095 y 0.135	Mala
>0.135 y 0.175	Muy Mala
>0.175	Extremadamente Mala
M	Mantenimiento

Flujos de Datos

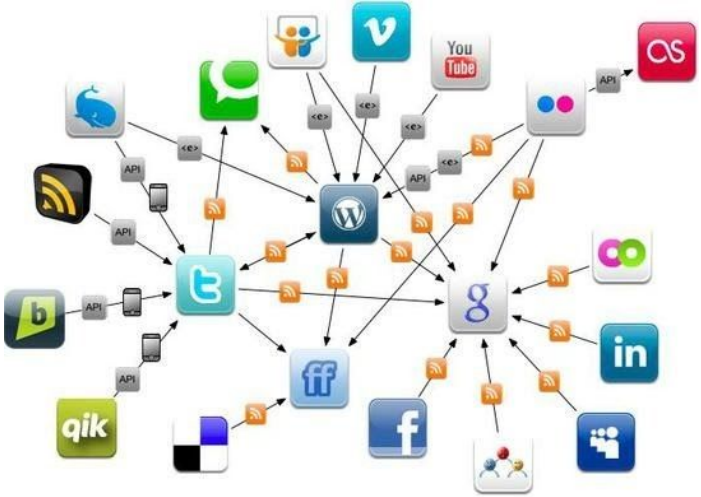
¿Qué es un flujo de datos?

Son datos que se generan constantemente (tiempo real) a partir de miles de fuentes de datos.

- Los datos son enviados simultáneamente en conjuntos de tamaño pequeño (kbs).
- Si los datos no se almacenan o se procesan rápido, estas se perderían.
- Esquema dinámico.

Flujos de Datos

- Atributos
- Marca de tiempo
- Dato crudo



Ejemplos

- Monitoreo
- Dispositivos IoT
- Internet y tráfico web
- Transacciones financieras
- Videojuegos en línea
- Videos

Flujos de Datos

Un sensor en el océano envía cada hora la temperatura del agua a una estación hidrológica (tasa de envío baja 4kb)

Flujos de Datos

Aproximadamente existen en órbita **5,000 satélites** que captan imágenes multiespectrales de la Tierra de **resolución media y alta**.
Aproximadamente capturan y envían: millón y medio de imágenes diarias

Cámaras de vigilancia generalmente producen imágenes de baja resolución (en comparación con los satélites), sin embargo el intervalo de envío es de 1 segundo.
Londres tiene alrededor de 6 millones de cámaras.

Internet y tráfico web

- Google procesa 81,226 búsquedas por segundo.
- 3.5 miles de millones de búsquedas por día.
- En 1999, a Google le tomó un mes indexar aprox. 50 millones de páginas. En 2012 le tomó un minuto.
- Cada pregunta viaja 1500 millas (hacia el centro de datos y de regreso).
- La respuesta a una consulta tarda 2 segundos (usando 1000 computadoras).

Aplicaciones

- Sencillas
  - Implementación de mínimo - máximo.
  - Generación de informes básicos.
  - Emitir alertas.
- Complejas
  - Uso de aprendizaje máquina
  - Procesamiento de eventos y transmisiones.

Retos

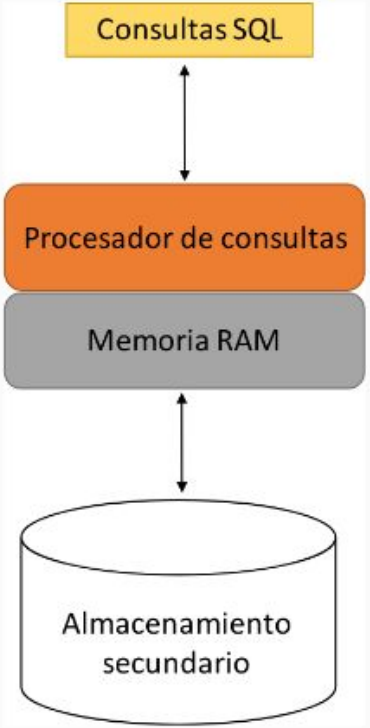
1. Memoria limitada para almacenar datos.
2. Debido a la vasta cantidad de datos, no es siempre posible generar respuestas exactas.
3. Se espera que la calidad de la respuesta sea confiable.
4. ¿Cómo trabajar los datos?

¿Cómo se procesan los flujos de datos?

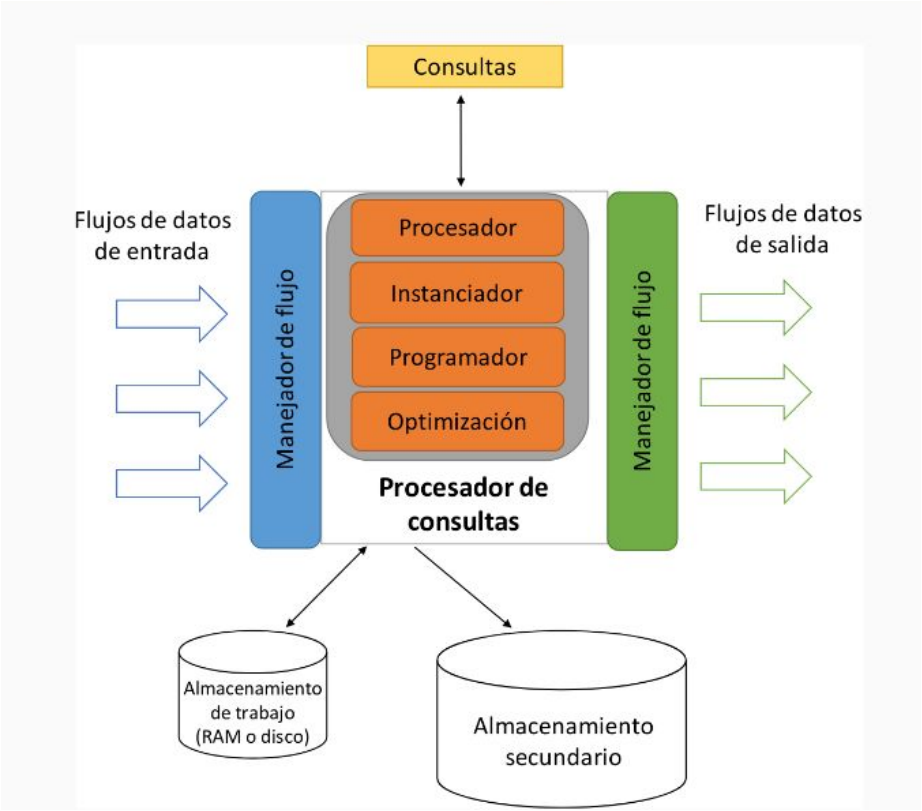
Un procesador de flujos de datos es un tipo de sistema de administración de datos (DSMS)

- Cualquier número de flujos puede ingresar al DSMS.
- Los flujos que se reciben no necesariamente deben tener la misma tasa de datos o tipo de datos.
- El tiempo entre flujos no necesita ser uniforme.
- Los algoritmos para procesar los flujos pueden involucrar resumen, filtrado o uso de ventanas.

Modelo general de un DBMS



Modelo general de procesamiento de flujos de datos



Consultas sobre flujos de datos

- Las consultas son frecuentes:
  - los flujos de datos son evaluados a medida que se van recibiendo.
- Actualizaciones constantes.
- Las consultas con complejas:
  - Preprocesamiento de atributos y extracción de datos crudos.

Existen formas generales para hacer consultas sobre los flujos de datos.

Consultas sobre flujos de datos

Consultas permanentes.

Están almacenadas dentro del procesador, son ejecutadas permanentemente y producen salidas.

Un sensor en el océano envía cada hora la temperatura del agua a una estación hidrológica (tasa de envío baja 4kb)

Consultas sobre flujos de datos

Consultas Ad - Hoc

Se realiza una sola vez sobre el flujo o los flujos actuales.

Un enfoque común es almacenar una ventana deslizante de cada flujo en el working storage.

Ventanas deslizantes

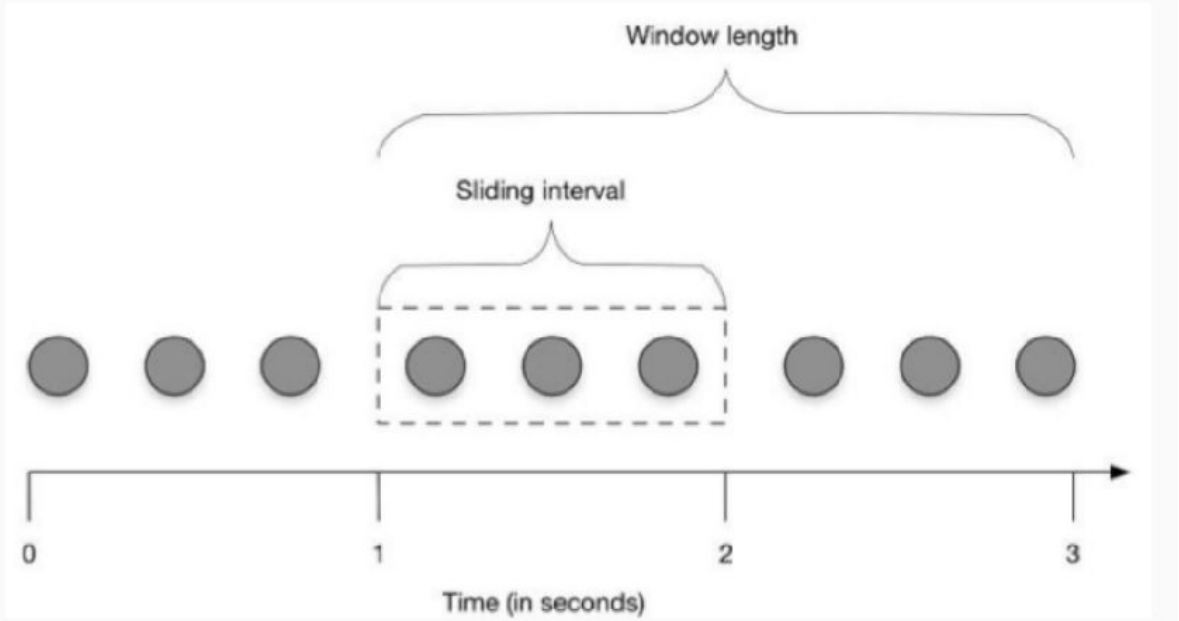
Técnica para el procesamiento de flujos de datos el cual divide dicho flujo basándose en dos parámetros:

- Longitud de la ventana (window lenght).
- Intervalo (sliding interval).

Ventanas deslizantes

Ejemplo.

Supongamos que debemos actualizar cada segundo con el valor de la mayor compra de los últimos dos segundos.



DBMS vs DSMS

DBMS	DSMS
Almacenamiento persistente	Almacenamiento transitorio
Acceso aleatorio	Acceso secuencial
Baja tasa de actualización	Tasas de múltiples Gbs
Servicios no de tiempo real	Servicios de tiempo real
Almacenamiento en disco	Memoria principal limitada

Capas en el procesamiento del flujo de datos

Capa de almacenamiento

- Ordenar registros
- Alta coherencia (lecturas y escrituras)
- Económica
- Rápida

Capa de procesamiento

- Consume los datos (capa de alm)
- Realiza las operaciones necesarias
- Notifica a la capa de almacenamiento que elimine los datos que ya no son necesarios.

Una capa adicional se añade para cada una de las capas:

- Planificar la escalabilidad
- Durabilidad de los datos
- Tolerancia a fallos

Plataformas Open Source

Actualmente existen numerosas plataformas que soportan el procesamiento de flujo de datos.

- Amazon Kinesis Streams
- Amazon Kinesis Firehose
- Apache Kafka
- Apache Flume
- Apache Spark Streaming
- Apache Storm

Muestreo

Muestreo

En muchos casos no es posible almacenar todos los datos de un flujo, por lo que es necesario realizar muestro.

Objetivo:

Seleccionar un subconjunto de datos del flujo de tal manera que sea representativo de todo el flujo de datos.

Ventaja:

Costo computacional más bajo debido a que estamos usando solo una porción del flujo.

Muestreo

Retos:

- ¿Cómo sabemos que tan largo es el flujo de datos?
- ¿Cada cuánto tiempo debemos realizar muestro?
- ¿Cómo hacemos el muestreo?

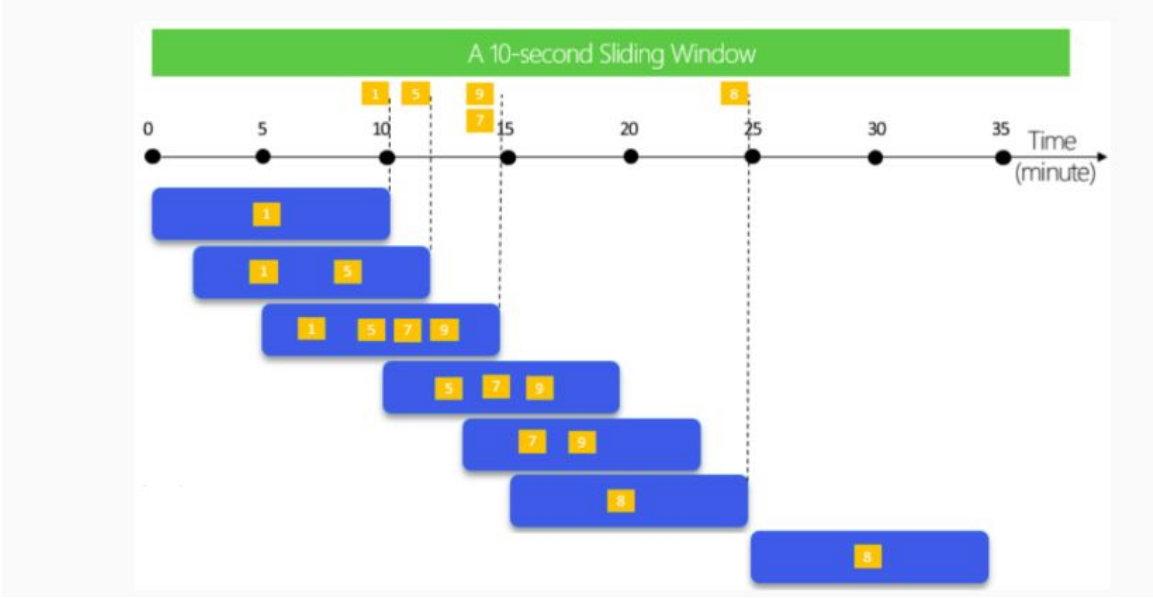
Estrategias de muestreo

- Ventanas deslizantes
- Muestreo de tamaño fijo
- Muestreo aleatorio

Ventanas deslizantes

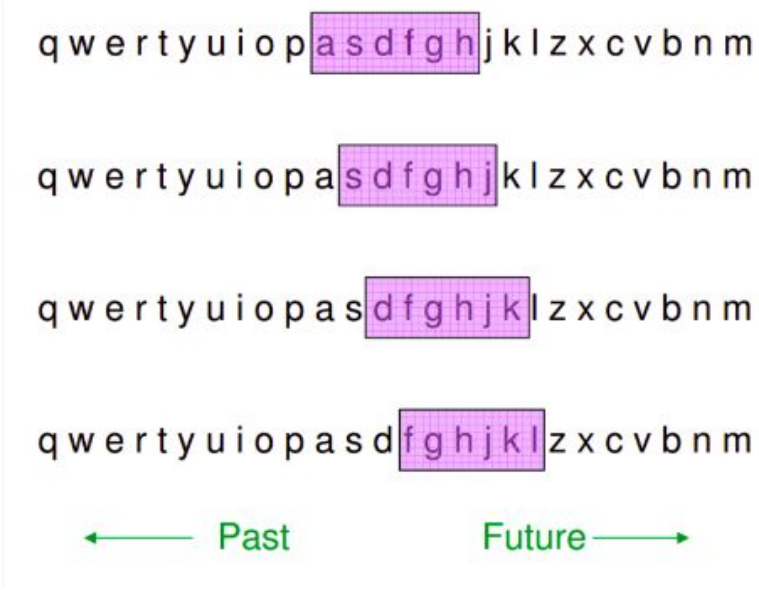
Las consultas se realizan sobre una ventana de tamaño *w*.

Si un elemento llega en el tiempo *t*, expira en el tiempo *t* + *w*.



Ventanas deslizantes

En este ejemplo, el tamaño de la ventana deslizante deslizante es 6, observamos el traslape entre datos.



Ventanas de promedio deslizantes

Calificaciones:

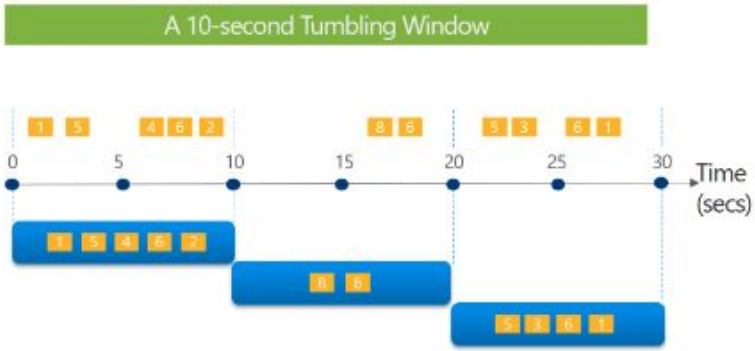
10	7.8	6.8	8.0	9.2	9.0

Tamaño de la ventana: 3

Ventanas de saltos de tamaño constante

Se divide el flujo de datos en segmentos de tiempo sin traslape.

Tell me the count of tweets per time zone every 10 seconds

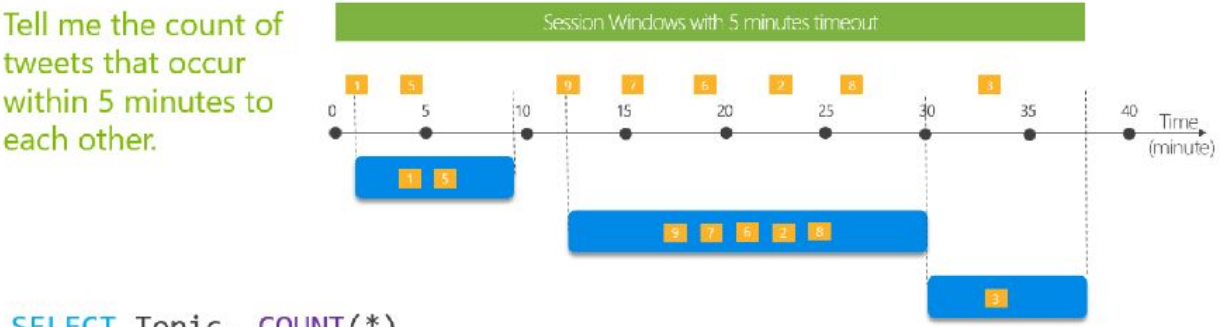



```
SELECT TimeZone, COUNT(*) AS Count
FROM TwitterStream TIMESTAMP BY CreatedAt
GROUP BY TimeZone, TumblingWindow(second,10)
```

Ventanas de Sesión

Agrupan eventos que llegan en tiempos similares, filtrando los periodos en los que no se recibe ningún dato.

Los parámetros de este tipo de ventana son el tiempo de espera y la duración máxima.




```
SELECT Topic, COUNT(*)
FROM TwitterStream TIMESTAMP BY CreatedAt
GROUP BY Topic, SessionWindow(minute, 5, 10)
```

Muestreo Aleatorio

Los elementos tienen la misma probabilidad de ser seleccionados.

Ejemplo:

Cobertura de la vacuna anti-sarampión entre 1,200 niños de la escuela “Benito Juárez”

Muestra: 60 niños

- Hacer una lista de todos los niños.
- Numerarlos del 1 al 1,200.
- Selección aleatoria de 60 números.

Muestreo determinista

Los elementos se seleccionan en base a criterios o reglas específicas.

Ejemplo:

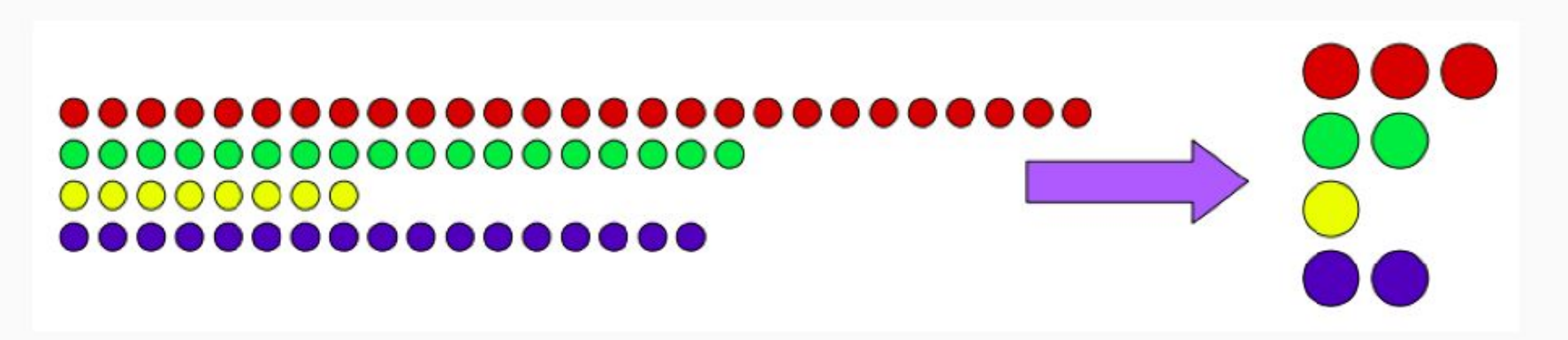
Únicamente se registrarán a los pacientes que acudan a la clínica en cierto día u horario en particular.

Nota: Es posible que los elementos seleccionados sean poco representativos de todos los conjuntos generados.

Muestreo de tamaño fijo

Consiste en muestrear una porción fija de los elementos recibidos.

(digamos 1 de 10)



Muestreo de tamaño fijo

Ejemplo:

Flujo de datos: consultas de usuarios.

Entrada: flujos de datos en forma de tupla.

User IP	Query	Time

¿Qué fracción de las consultas de un usuario son duplicadas?

Obtención de Muestras representativas

Hacer el muestreo tomando una muestra de cada usuario, puede arrojar resultados poco confiables.

¿Y si en lugar de tomar 1/10 de las búsquedas de cada usuario, todas las búsquedas de 1/10 de los usuarios?

Es decir, vamos a almacenar todas las búsquedas descartando las consultas del resto de los usuarios.

Tomando la IP del usuario como ID, muestreamos a/b usuarios usando una función hash que almacena las IPs en b cubetas, agregando las consultas del usuario si su valor hash es menor a a.

Como resultado obtendríamos una muestra más representativa.

Muestreo de presa (Reservoir Sampling)

Consiste en muestrear los primeros *m* datos recibidos y los mantiene en memoria (presa).

Cada nuevo elemento recibido tiene una probabilidad de *m/n* de reemplazar un elemento actual.

Procedimiento general:

1. Tomas los primeros k-elementos de flujo como muestra.
2. Supongamos que hemos visto n - 1 elementos, y ahora recibimos el n-ésimo elemento ($k < n$).
3. Con probabilidad k/n, mantenemos el elemento n-ésimo, reemplazando uno de los k elementos en la muestra.

Muestreo de presa (Reservoir Sampling)

Análisis

Las muestras son aleatorias y uniformemente distribuídas.

Supongamos que una muestra $x_\ell, 1 \leq \ell \leq n$ es la final, lo cual ocurre su se elige y ninguna de las otras muestras posteriores

$$x_{\ell+1}, x_{\ell+2}, \ldots, x_n$$

lo reemplaza.

$$P(x_\ell \text{ es muestra final}) = \frac{1}{k}$$

Filtrado

Filtrado de flujo de datos

Seleccionar elementos del flujo que cumplan con cierto criterio y descartar el resto.

Ejemplo:

Dado un flujo de números reales,

Flujo: 33, 71, 58, 12, 41, 56, 3, 89

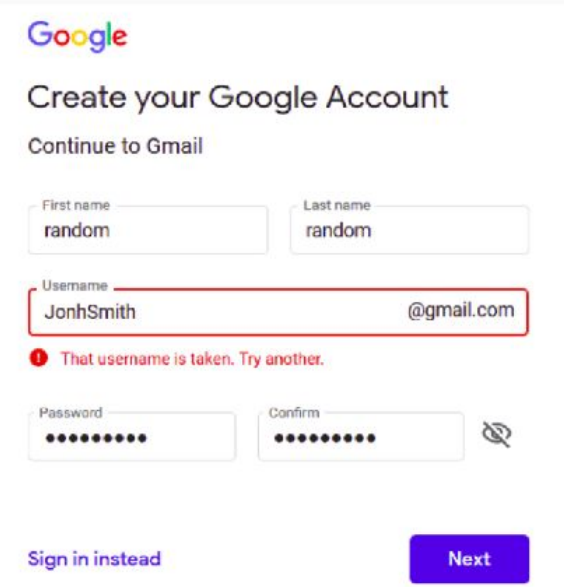
filtrar los datos que sean mayores a 50.

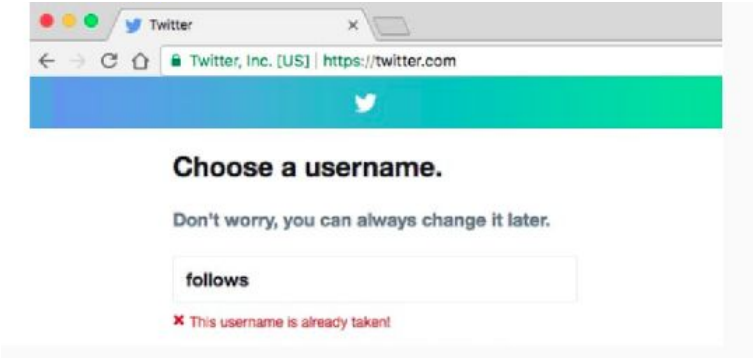
Esta tarea se vuelve más difícil cuando el criterio requiere verificar si el elemento pertenece a un conjunto dado, especialmente si este conjunto es tan grande que no cabe en memoria.

Pertenencia a un conjunto

Ejemplo:

¿Cómo podemos revisar rápidamente la disponibilidad de un nombre dentro de cientos de millones existentes?





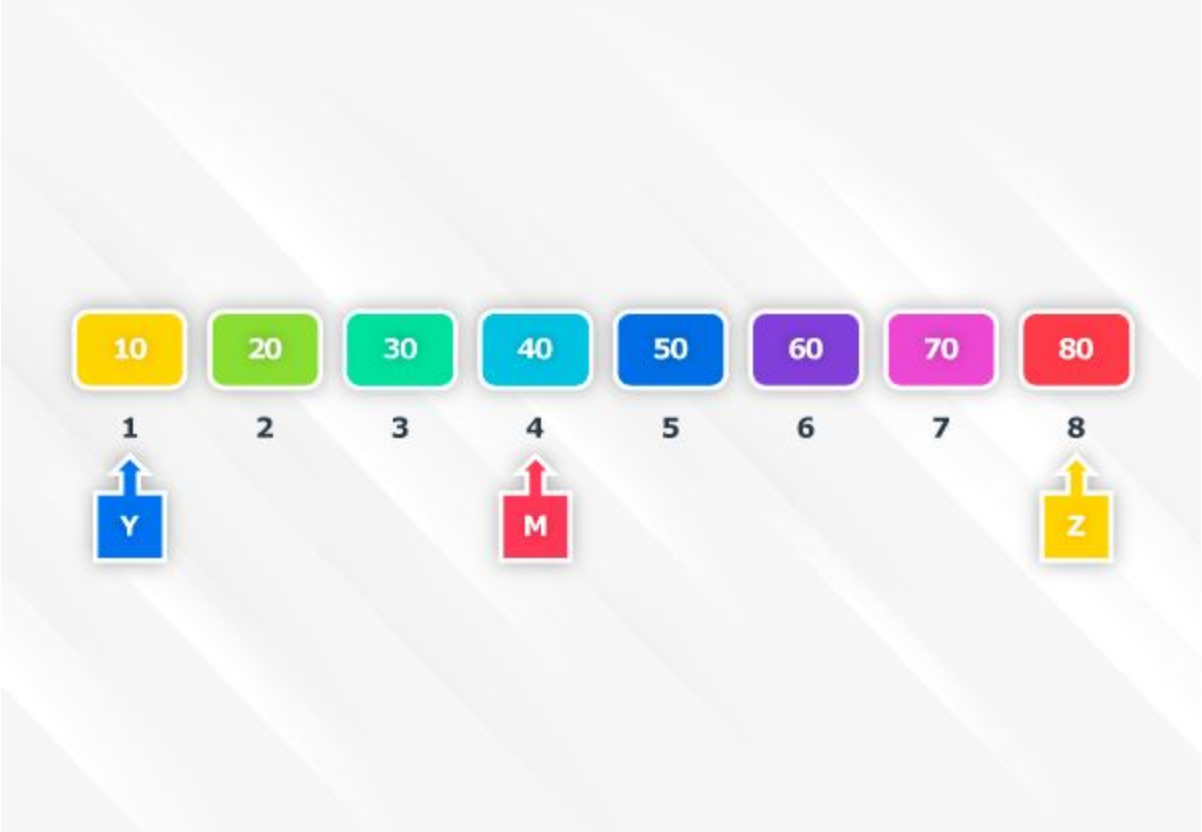
Búsqueda Binaria

Supongamos que almacenamos todos los nombres de las personas del salón, comparamos el nuevo nombre con el que está a principio de la lista y continuamos con el siguiente y con todos en orden.

En el peor de los casos, tendríamos que examinar todos los nombres para encontrar el que estamos buscando.

SI los nombres estuvieran ordenados alfabéticamente, la búsqueda binaria reduce significativamente el número de comparaciones.

Búsqueda Binaria



Búsqueda Binaria

Adivinar el número - Algoritmo

1. Sea *min* = 1 y *max* = n.
2. Calcula el promedio de *max* y *min* redondeando a entero hacia abajo.
3. Si adivinaste el número, lo encontraste. Podemos detenerlos.
4. Si el intento fue demasiado bajo, hacer *min* uno más grande que el intento.
5. Si el intento fue demasiado alto, hacer *max* uno más pequeño que el intento.
6. Regresar al paso 2.

Filtro Bloom

Es una estructura de datos probabilística y se emplea para evaluar si un elemento pertenece a un conjunto.

Desarrollado por Burton Howard Bloom en 1970.

Elimina la mayoría de los elementos que no pertenecen al conjunto.

Es muy eficiente en memoria, ya que no requiere mantener el conjunto en memoria.

Tiene falsos positivos.

Filtro Bloom

Un filtro Bloom consta de:

1. Una matriz de *n* bits, inicialmente 0.

2. Una colección de funciones hash $h_1, h_2, \ldots, h_k$. Cada función hash mapea valores “clave” para *n* cubos, correspondiente a los *n* bits.

3. Un conjunto *S* de *m* valores clave.

Filtro Bloom

El propósito es permitir el paso de todos los elementos de flujo cuyas claves están en *S*, mientras rechaza la mayoría de los elementos del flujo cuyas claves no están en *S*.

1. Para inicializar la matriz de bits, comience con todos los bits 0.
2. Tomar cada valor clave en *S* y hashearlo usando cada una de las *k* funciones hash.
3. Establecer en 1 cada bit que sea hi(k) para cada función hash h_i y algún valor clave *k* en *S*.

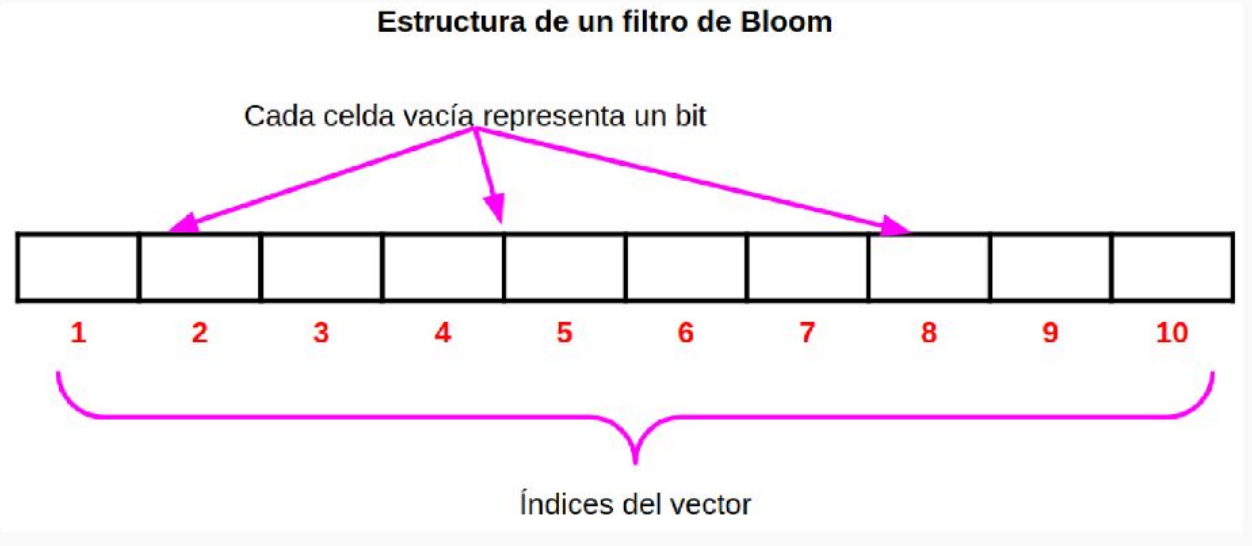
Filtro Bloom

Para probar una clave k que llega al flujo, verificar que todas las

$$h_1(k), h_2(k), \ldots, h_k(k)$$

son 1's en el arreglo de bits. Si todos son 1, entonces se deja pasar el elemento del flujo, si uno o más de estos bits son cero, entonces *K* no podría estar en *S*, así que se rechaza ese elemento del flujo.

Filtro Bloom - Estructura



Filtro de Bloom vacío: es un vector de *m* bits donde todos los valores son ceros.

0	0	0	0	0	0	0	0	0	0
1	2	3	4	5	6	7	8	9	10

Filtro Bloom - Llenado

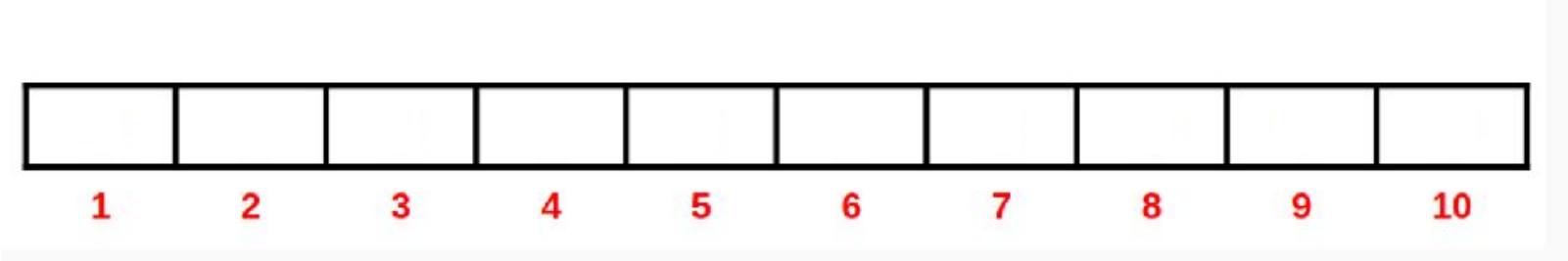
Para añadir un elemento *x* al filtro *S*

- *x* debe transformarse a un conjunto de bits a través de *k* funciones hash. El resultado de cada función indica el índice dentro del filtro, que debe cambiarse de 0 a 1.

Antes de empezar, debemos definir:

- ***m*** tamaño del vector
- ***n*** número de elementos a agregar
- ***k*** número de funciones hash

Filtro Bloom - Ejemplo



- $m = 10$
- $n = 2$
- $k = 3$

Filtro Bloom - Falsos Positivos

¿Cómo evitar / reducir los falsos positivos?

- Más espacio (incrementar el tamaño del vector)
- Incrementar k (número de funciones hash)

La probabilidad de que un bit no sea puesto a 1 durante el registro de un elemento es

$$1-\frac{1}{m}$$

Filtro Bloom - Falsos Positivos

- Para k funciones hash esto es

$$\left(1-\frac{1}{m}\right)^{k}$$

- Usando la identidad de e^{-1}

$$\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^{k}=\frac{1}{e}$$

- Esto es, cuando *m* es muy grande

$$\left(1-\frac{1}{m}\right)^{k}=\left(\left[1-\frac{1}{m}\right]^{m}\right)^{\left(\frac{k}{m}\right)}\approx e^{\frac{-k}{m}}$$

Filtro Bloom - Falsos Positivos

- Si ya han sido registrados en n elementos en el arreglo de bits, la probabilidad de un bit dado esté en 0 es

$$\left(1-\frac{1}{m}\right)^{(k-n)} \approx e^{\frac{-kn}{m}}$$

- La probabilidad de que ese bit esté en 1 es

$$\left(1-\frac{1}{m}\right)^{kn} \approx 1-e^{\frac{-kn}{m}}$$

- Para que exista un falso positivo, todos los bits de las funciones hash deben estar en 1 la probabilidad es

$$\left(\left[1-\frac{1}{m}\right]^{(kn)\,k}\right) \approx \left(1-e^{\frac{-kn}{m}}\right)^{k}$$

Selección de funciones Hash para el Filtro Bloom

Las funciones hash en el filtrado de Bloom deben ser:

- Independientes
- Uniformemente distribuídas
- Deben ser rápidas para su cálculo
- No criptográficas

Cuando el número de funciones hash incrementa, el filtrado se vuelve lento.

Ejemplos de funciones con bajas tasas de colisiones y no criptográficas.

- MURMUR (2011)
- FNV (1991)
- Jenkins o HashMix (1997)

Aplicaciones del Filtrado de Bloom

- Medium
  - servicio de publicación de blogs, creado por los fundadores de Twiter en 2012.
  - Usa filtros de Bloom para recomendar publicaciones a los usuarios filtrando las publicaciones que ya ha visto el usuario.

- Quora
  - Red social de preguntas y respuestas creada en junio de 2004
  - Usa filtros de Bloom para filtrar historias no vistas.

- Google Chrome
  - Usa filtros de Bloom para detectar URLs maliciosas

Conteo

Contando elementos distintos

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

- Dado un flujo de elementos $x_1, x_2, \ldots, x_n$ encontrar el número de elementos distintos n, donde $n = |\{ x_1, x_2, \ldots, x_n \}|$
- 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 a un sitio web, secuencias de ADN, dispositivos IoT, etc.

Contando elementos distintos

Ejemplo:

Dado el flujo a, b, a, c, d, b, d encontrar n.

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, evitando a través de algoritmos como Flajolet - Martin

Contando elementos distintos

Supongamos que tenemos un flujo de datos con m elementos.

¿Cuántos elementos distintos tenemos?

Si m es pequeña:

- Solución: generar un diccionario
- Costo computacional

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.

Flajolet - Martin

1984 - Philippe Flajolet & Nigel Martin

“Probabilistic Counting Algorithms for DataBase Applications”

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

Flajolet - Martin

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).

La propiedad de uniformidad de las funciones 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.

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 $\phi=0.77351$ es un factor de corrección.

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: un 0 de más y se duplica.

La mediana es menos sensible pero solo obtiene estimaciones que son potencias de 2.

Una mejor estrategia es agrupar las estimaciones, obtener la mediana de cada grupo y 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.

Sea r la primera posición del arreglo de bits cuyo valor es cero, un estimador del número de elementos en el flujo de datos es $\frac{2^k}{\phi}$ donde $\phi=0.77351$ es un factor de corrección.

Análisis de las Estimaciones

La probabilidad de que un elemento dado 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 $(1-2^{r})^{m}=((1-2^{-r})^{(2^{r})})^{(m-2-r)}$. Cuando r es suficientemente grande este valor se aproxima a $(1-\varepsilon)^{(\frac{1}{\varepsilon})}=\frac{1}{e}$

La probabilidad de que ninguno de los valores hash de los *m* elementos distintos termine en al menos *r* 0s es $e^{-m2^{r}}$

Análisis de las Estimaciones

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.

Ejemplo de estimación del número de elementos distintos

Ejemplo:

Flujo: 1, 4, 2, 1, 2, 4, 4, 4, 1, 2, 4, 1, 7

$h(a) = (3x + 1) \mod 5$

Ventajas y Desventajas

Ventajas	Desventajas
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.

Práctica (Individual)

Dado el flujo:

3, 1, 4, 1, 5, 9, 2, 6, 5 y las funciones hash

$h_1(x) = (2x + 1) \bmod 32$

$h_2(x) = (3x + 7) \bmod 32$

$h_3(x) = 4x \bmod 32$

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

Estimación de Momentos

Estimación de momentos en Flujos de datos

Generalización del conteo de elementos distintos.

- Objetivo: Estimar los momentos en un flujo de datos, lo cual se obtiene mediante la distribución de frecuencias de diferentes elementos.

Sea m_e el número de ocurrencias del elemento e en el flujo, el momento i-ésimo está definido por

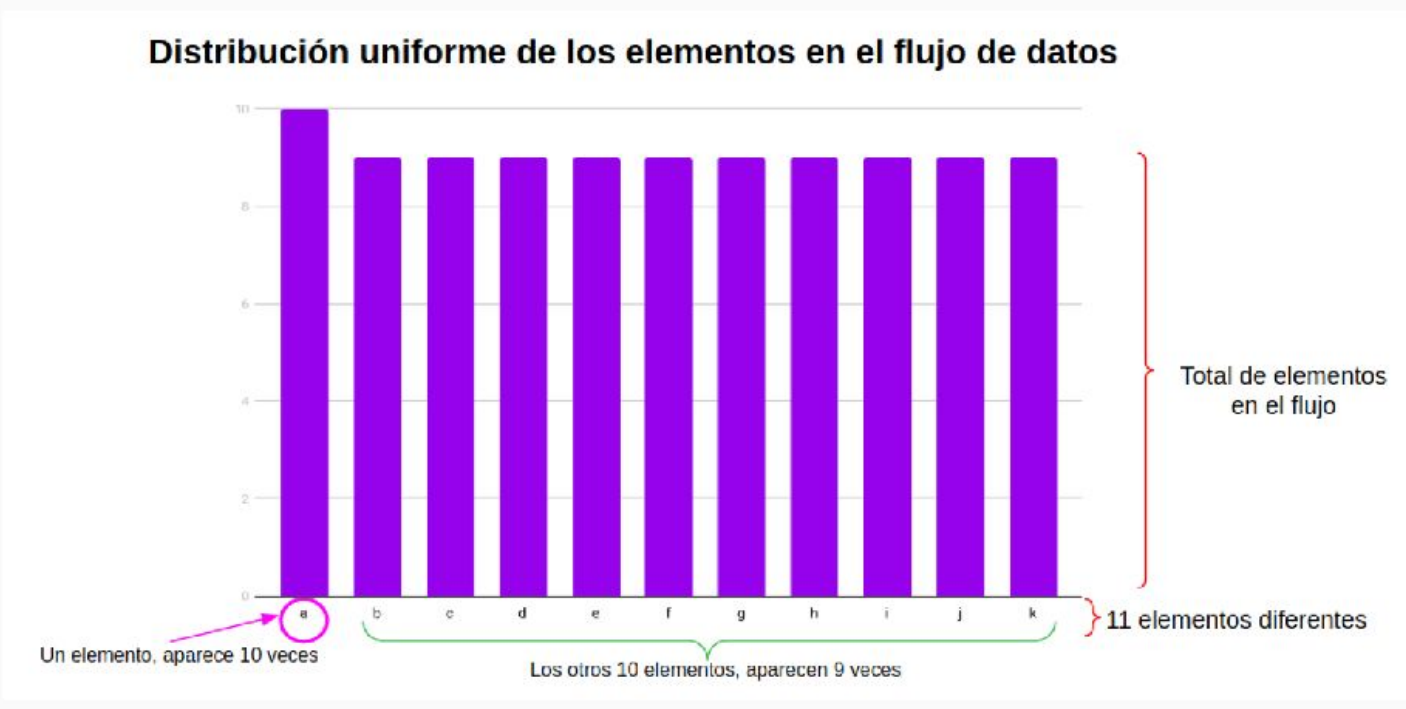
$$\sum_{e \cup \Omega} (m_e)^i$$

donde Ω es el conjunto universal.

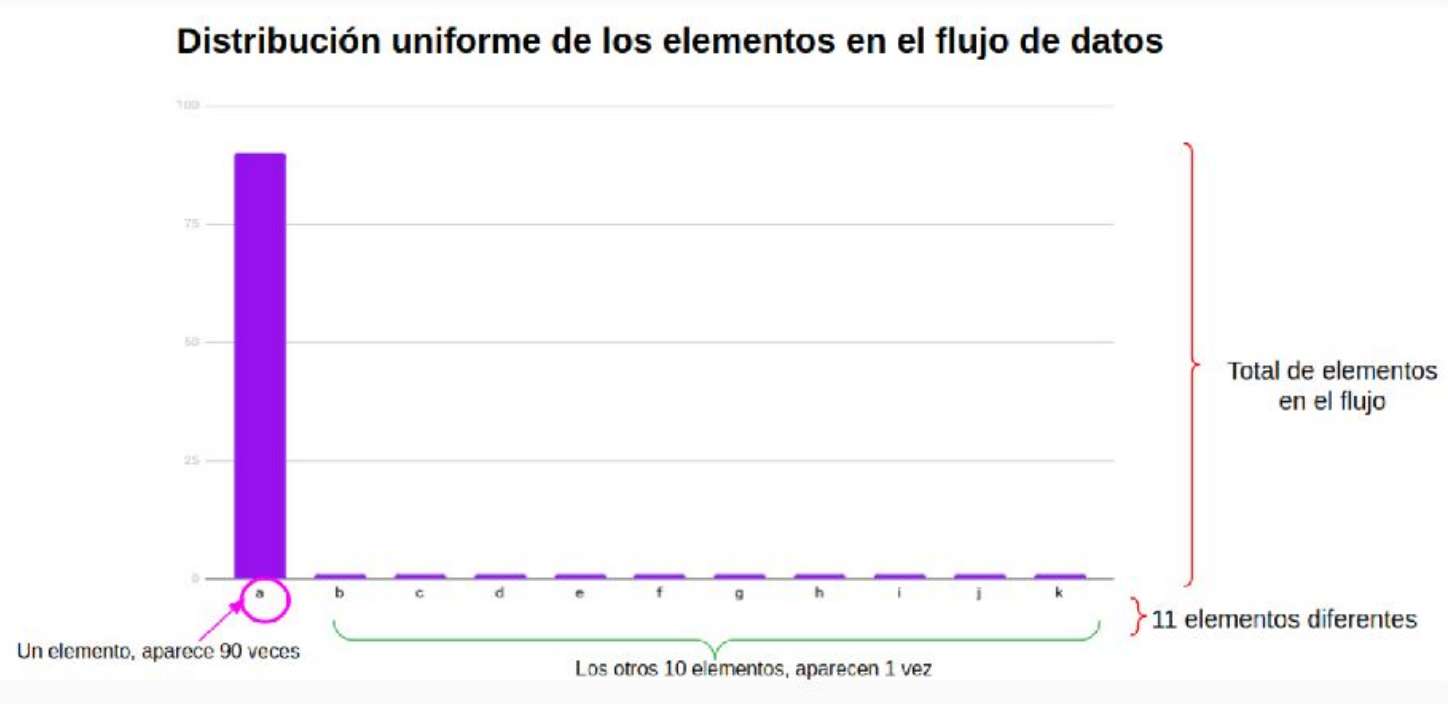
Estimación de momentos en Flujos de datos

- El momento 0 es el número de elementos distintos.
- El momento 1 es la suma de m_i, es decir, el tamaño del flujo de datos.
- El momento 2 es la suma de los cuadrados de m_i, también conocido como número sorpresa.

Estimación de momentos en Flujos de datos



Estimación de momentos en Flujos de datos



Algoritmo de Alon - Matías - Szegedy

- Algoritmo para el cálculo de momentos en flujos de datos, definido por Noga Alon, Yossi Matías y Mario Szegedy.

- Se enfoca en aproximar la suma de las entradas al cuadrado de un vector definido por un flujo de datos.

- Permite calcular cualquier momento aún si no es posible almacenar todas las cuentas m_i de todos los elementos.

Algoritmo de AMS para estimar el segundo momento

- Dado un flujo de tamaño *n* constante, se toman k variables $x_1, x_2, \ldots, x_k$, seleccionando *k* posiciones en el flujo de forma aleatoria y uniforme.

- Las variables almacenan el elemento de la posición correspondiente en $x_k \cdot$ elemento y un valor entero $x_k \cdot$ valor, el cual se inicializa con 1 y se incrementa en 1 cada vez que se encuentra una ocurrencia de $x_k \cdot$ elemento.

- El segundo momento de cualquier variable x_k se estima con $n \cdot (2 \cdot x_k \cdot \text{valor}-1)$

Ejemplo

Cálculo del 2do momento.

Considera el flujo a, b, c, b, d, a, c, d, a, b, d, c, a, a, b.

Calcula el primer y segundo momento.

Segundo Momento con el algoritmo AMS

1. Seleccionamos 3 variables aleatorias: x_1, x_2, x_3,
2. Supongamos que las posiciones para las tres variables son 3, 8, 13
   a. En la posición 3, encontramos el elemento c, al cual llamamos x_1· elemento = c
   b. En la posición 8, encontramos el elemento d, al cual llamamos x_2· elemento = d
   c. En la posición 13, encontramos el elemento a, al cual llamamos x_3· elemento = a

$$a, b, \overbrace{\mathbf{c}}^{3}, b, d, a, c, \overbrace{\mathbf{d}}^{8}, a, b, d, c, \overbrace{\mathbf{a}}^{13}, a, b$$

Segundo Momento con el algoritmo AMS

$$a, b, \overbrace{\mathbf{c}}^{3}, b, d, a, c, \overbrace{\mathbf{d}}^{8}, a, b, d, c, \overbrace{\mathbf{a}}^{13}, a, b$$

$X_1.elemento = c$	$X_1.valor = 1$	$X_1.valor = 2$	$X_1.valor = 3$
$X_2.elemento = d$	$X_2.valor = 1$	$X_2.valor = 2$	
$X_3.elemento = a$	$X_3.valor = 1$	$X_3.valor = 2$	

El valor final es $X_1.valor = 3$, $X_2.valor = 2$ y $X_3.valor = 2$

Segundo Momento con el algoritmo AMS

Estimamos el segundo momento con $n \cdot (2 \cdot x_k \cdot \text{valor}-1)$

- Para x_1: $15 \cdot (2 \cdot 3 - 1) = 75$
- Para x_2: $15 \cdot (2 \cdot 2 - 1) = 45$
- Para x_3: $15 \cdot (2 \cdot 2 - 1) = 45$

Promediando las estimaciones de cada variable tenemos

$$(75 + 45 + 45) / 3 = 55$$

El valor esperado de cualquier variable es el segundo momento del flujo del que fue generada.

Segundo Momento con el algoritmo AMS

Sea *e(i)* el elemento que aparece en la posición *i* en el flujo y *c(i)* el número de veces que aparece este elemento de la posición *i* a la *n*, el valor esperado del estimador del segundo momento es:

$$\mathrm{E}\left[n \cdot (2 \cdot X_k.valor - 1)\right] = \frac{1}{n}\sum_{i=1}^{n} n \cdot (2 \cdot c(i) - 1) = \sum_{i=1}^{n} (2 \cdot c(i) - 1)$$

Segundo Momento con el algoritmo AMS

Podemos poner el valor esperado de otra forma si agrupamos los términos de todas las posiciones que tienen el mismo elemento.

Para el ejemplo anterior, la letra a aparece m_a veces. Si tomamos los términos de la última posición hacía la primera tendríamos:

$$2 \cdot 1 - 1 = 1$$

$$2 \cdot 2 - 1 = 1$$

$$2 \cdot 3 - 1 = 5$$

$$\vdots$$

$$2 \cdot m_a - 1$$

Segundo Momento con el algoritmo AMS

Por lo tanto, podemos, reescribir el valor esperado para cada elemento $e \in \Omega$ de la siguiente manera

$$\mathrm{E}\left[n \cdot (2 \cdot X_k.valor - 1)\right] = \sum_{e \in \mathbb{U}} \left[1 + 3 + 5 + \cdots + (2 \cdot m_e - 1)\right]$$

$$\left[1 + 3 + 5 + \cdots + (2 \cdot m_e - 1)\right] = (m_e)^2$$

$$\mathrm{E}\left[n \cdot (2 \cdot X_k.valor - 1)\right] = \sum_{e} (m_e)^2$$

Segundo Momento con el algoritmo AMS

Para el estimador del segundo momento tenemos

$$n(2 \cdot x_k \cdot valor - 1) = n \cdot (x_k \cdot valor^2 - (x_k \cdot valor - 1)^2)$$

El estimador del tercer momento es

$$n(x_k \cdot valor^3 - (x_k \cdot valor - 1)^3) = n \cdot (3 \cdot x_k \cdot valor^2 - 3 \cdot x_k \cdot valor + 1)$$

En general, el estimador del i-ésimo momento es

$$n(x_k \cdot valor^i - (x_k \cdot valor - 1)^i)$$

Estimación para flujos Infinitos

Hasta ahora hemos considerado que n es constante, sin embargo en la práctica no lo es.

¿Cómo seleccionamos las posiciones para las variables?

Si se hace al inicio y no se actualiza, conforme el flujo crezca, habría un sesgo hacia las primeras posiciones y la estimación sería demasiado grande.

Si se espera demasiado y no mantenemos suficientes variables en las posiciones al principio del flujo, podemos tener una estimación poco confiable.

Estimación para flujos Infinitos

Estrategia de selección de posiciones.

1. Se toman las primeras s posiciones del flujo como variables.
2. Se elige la posición $n > s$ con probabilidad s/n
    a. SI es elegida, se selecciona de forma aleatoria y uniforme una de las s variables y se reemplaza por la de la posición n, en caso contrario se mantienen las posiciones de las s variables.

Ejercicio

Obtener el primer y segundo momento para el flujo:

3, 1, 4, 1, 3, 4, 2, 1, 2

¿Cuál es el tercer momento?

Problemas para conteo de flujos

Ahora dirigimos nuestra atención a los problemas de conteos para flujos.

Supongamos que tenemos una ventana de longitud N en un flujo binario.

Queremos responder consultas del tipo:

¿Cuántos 1's hay en las últimos k-bits?

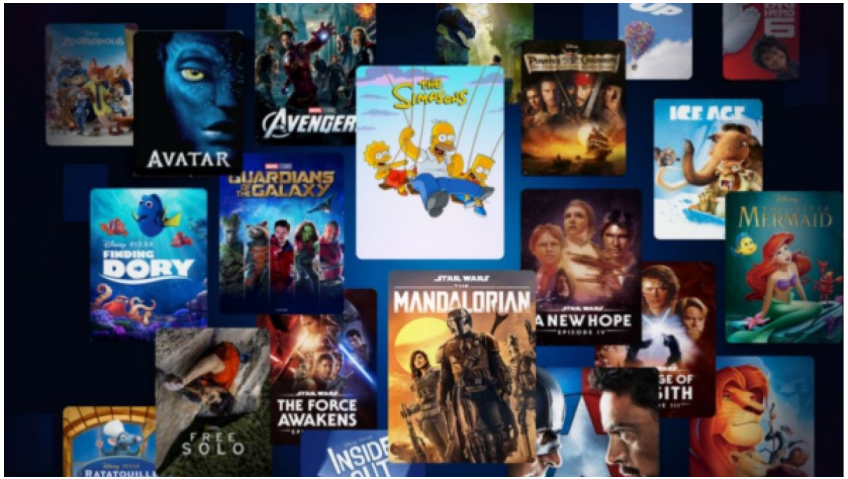
Para cualquier $k \leq N$

Ventanas deslizantes

Decaying Windows

Supongamos que tengamos un flujo cuyos elementos con las entradas de cine compradas en todo el mundo, con el nombre de la película como parte del elemento.

Se quiere mantener un resumen de la transmisión de las películas más populares “actualmente”.



Decaying Windows

Una solución sería imaginar un flujo de bits para cada película. El i-ésimo bit tiene valor 1 si la i-ésima entrada es para esa película y en 0 en caso contrario.

Elija un tamaño de ventana N, que es la cantidad de boletos más recientes que se considerarían en la evaluación de la popularidad.

1. Estimar la cantidad de boletos para cada película.
2. Clasificar las películas según sus conteos estimados.

Decaying Windows

Formalmente dejemos que una secuencia actualmente consista de los elementos $a_1, a_2, \ldots, a_t$ donde a_1 es el primer elemento en llegar y a_t es el elemento actual.

Sea c una pequeña constante como $10^{(-6)}$ o $10^{(-9)}$. Definir la ventana que decae exponencialmente para que esta corriente sea la suma.

$$\sum_{i=0}^{t-1} a_{(t-1)}(1-c)^i$$

Decaying Windows

El efecto de esta definición es distribuir los pesos de los elementos del flujo hacia atrás en el tiempo a medida que avanza el flujo.

En contraste una ventana fija pondría el mismo peso en cada uno de los elementos.

Elemento más popular

Nos aproximamos a una ventana deslizante que contiene las últimas mil millones de entradas vendidas.

Para cada película tenemos un flujo separado por 1’s y 0’s.

Establecemos un umbral, (ej. ½) para mantener las películas o no dentro del conteo.

El valor del umbral debe ser siempre < 1

Elemento más popular

Cuando llegue un ticket nuevo a la transmisión:

1. Para cada película cuyo puntaje mantengamos actualmente, multiplique su puntaje por (1 - c).

2. Suponga que el nuevo boleto es para la película M. Si no hay puntuación para M, cree una e inicializarla en 1.

3. SI algún puntaje está por debajo del umbral ½ , baje el puntaje.

Elemento más popular

Puede que no sea obvia la cantidad de películas cuyas puntuaciones se mantienen en cualquier momento es limitada. La suma de todas las puntuaciones es 1/c.

No puede haber más de 2/c películas con puntaje de ½ o más, de lo contrario la suma de los puntajes sumaría 1/c. Por lo tanto 2/c es un límite en el número de películas que se cuentan en cualquier momento, por lo que la cantidad de películas contadas activamente sería mucho menor que 2/c.

Ejercicio

Encontrar las 5 películas más populares de acuerdo al rating.

https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset