Data Stream

A data stream is a countably infinite sequence of elements and is used to represent data elements that are made available over time.
Examples are readings from sensors in an environment monitoring application, stock quotes in financial application etc.

Data Streams are dynamic data that is generated on a continual basis.
This allows you to analyze data in real-time and gain insights on a wide range of scenarios.

By using stream processing technology, data stream can be processed, stored, analyzed, and acted upon as it's generated in real-time.

Types of Data-Stream Processing

There are two types of data stream processing —

1. Batch Processing Stream.
2. ~~Sotor~~ Real-Time Stream.

Batch Processing methods requires data to be downloaded as batches before it can be processed, stored, or analyzed, whereas.

Streaming data flows in continously, allo-wing that data to be processed simultaneously, in real-time the second it's generated.

Examples of Stream Sources.

1. Sensors Data

- Sensor data are the data produced by the sensors placed at different places.
- Different sensors such as temperature sensor, GPS sensors and more are installed at different places for capturing the temperature, height, and many other information of that particular place.
- The data produced by sensor is a stream of real numbers.
- This data given by the sensor is stored in main memory. These sensors send large amount of data every 10th of second.

2. Image Set

- Satellites often send down to earth streams consisting of many terabytes of images per day.
- Surveillance cameras produce images with lower resolution than satellites, but there can be many of them, each producing a stream of images at interval like 1 second

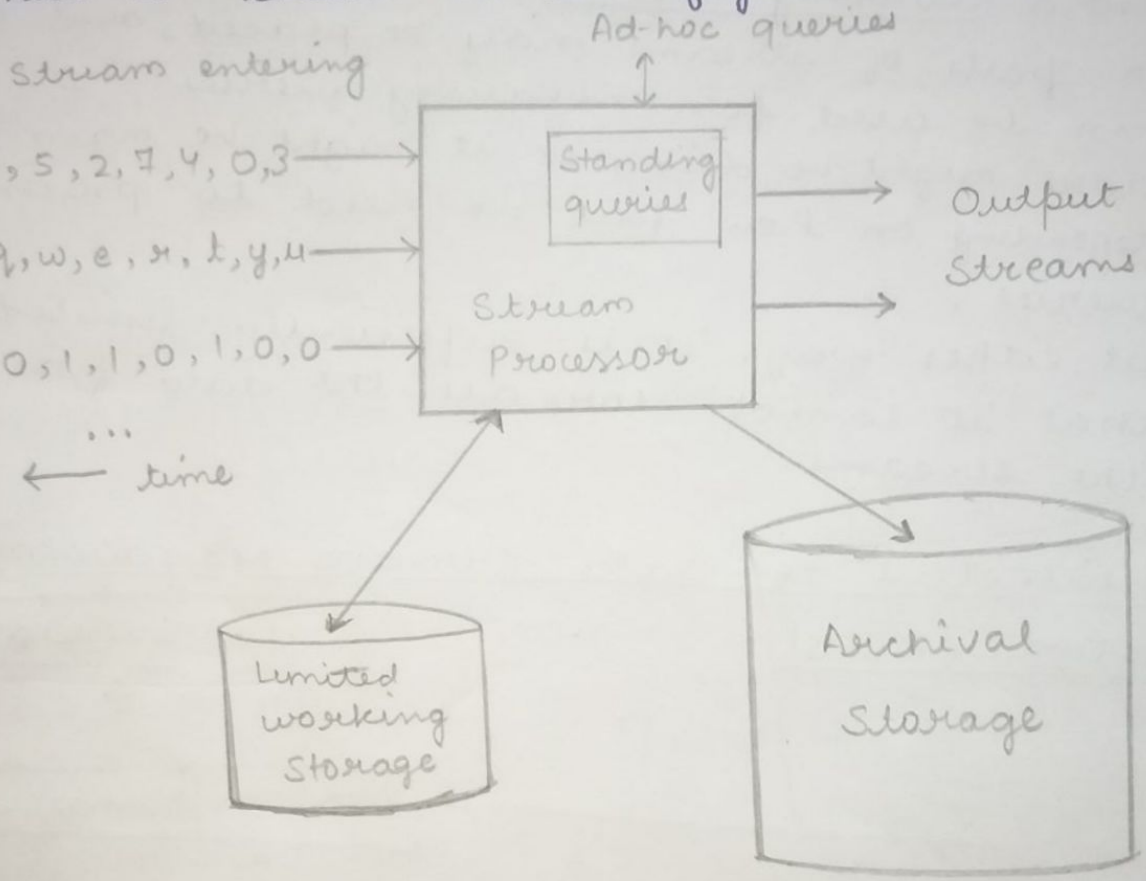
3. Internet and web traffic

- A switching node in the middle of the internet receives streams of IP Packets from many inputs and routes them to its outputs.
- The job of the switch is to transmit data and not to retain it or query it and provide more capability into the switch.

- Websites receive streams of various types. For eg, google receives several hundred million search queries per day. Yahoo accepts billions of clicks per day on its various sites.
- Many things can be learnt or extract from streams of data.

The Stream Data Model :-

In analogy to a database management system (DBMS), we can view a stream processor as a kind of data-management system, the high level organization of which is shown in the figure below.

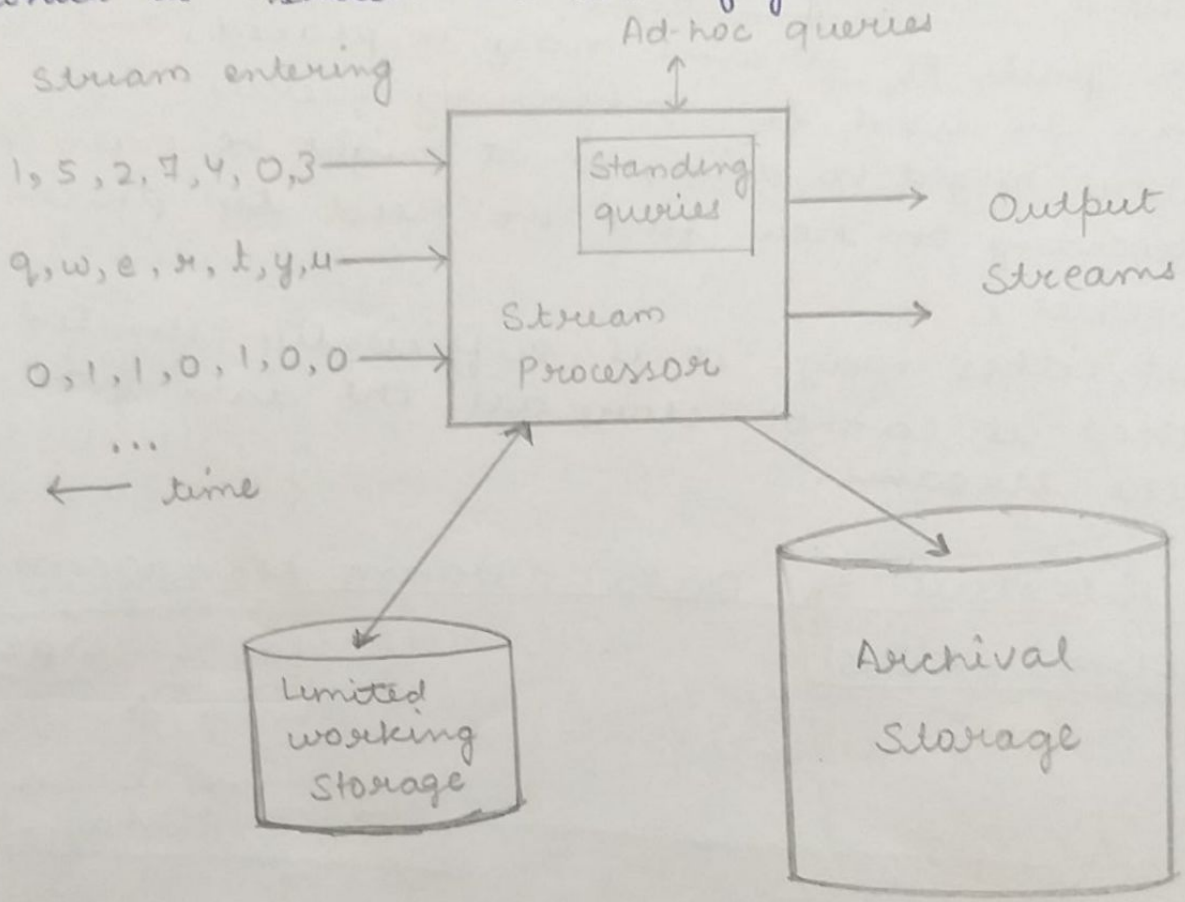


Any number of streams can enter the system. Each stream can provide elements at its own schedule; they need not have the same data rates or data types, and the time between elements of one stream need not to be uniform.

The fact that the rate of arrival of stream elements is not under the control of the system distinguishes stream processing from the processing of data that goes on within a database-management system. The latter system controls the rate at which data is read from the disk, and therefore never has to worry about data getting lost as it attempts to execute queries.

The Stream Data Model :-

In analogy to a database management system (DBMS), we can view a stream processor as a kind of data-management system, the high level organization of which is shown in the figure below.



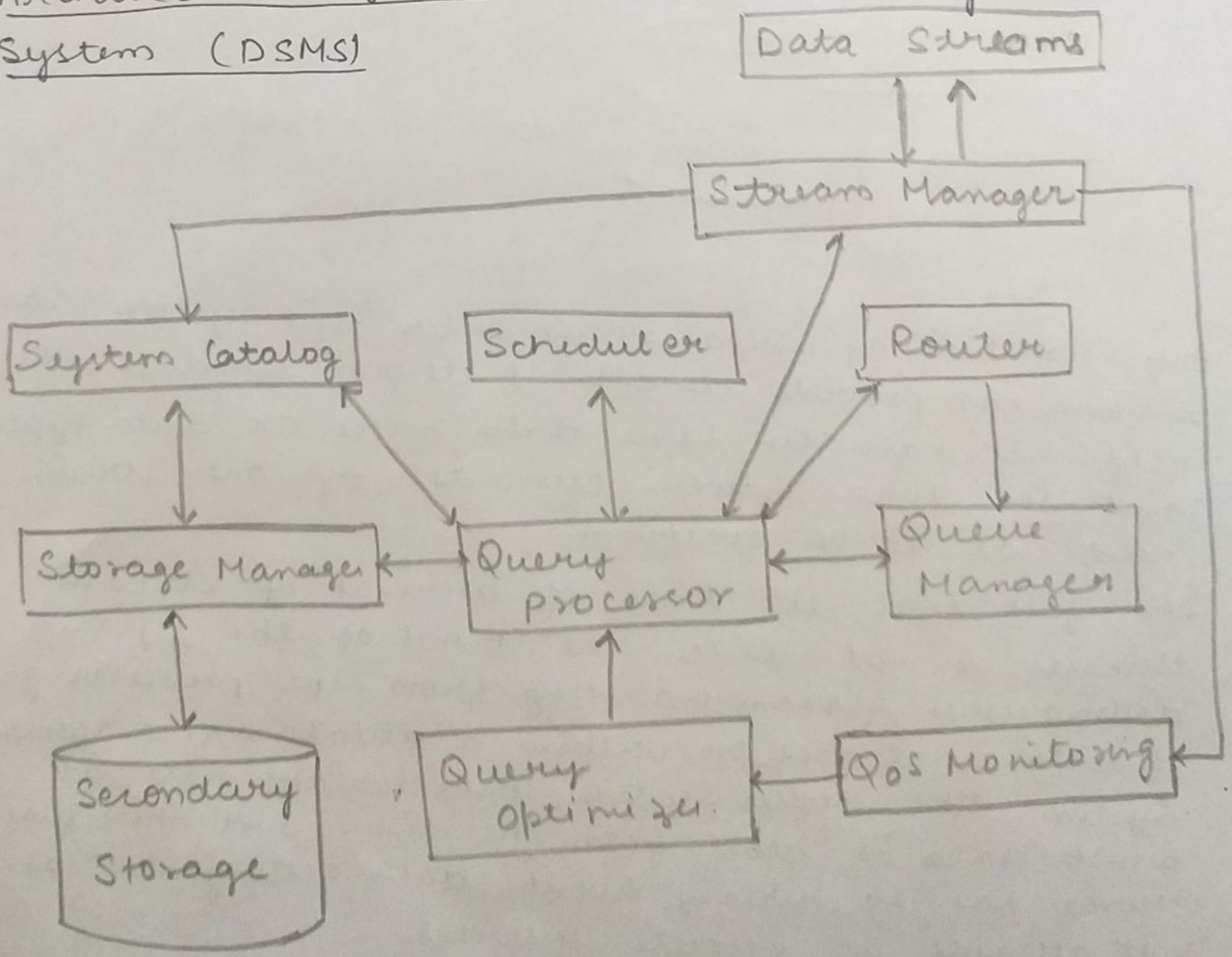
Any number of streams can enter the system. Each stream can provide elements at its own schedule; they need not have the same data rates or data types, and the time between elements of one stream need not to be uniform.

The fact that the rate of arrival of stream elements is not under the control of the system distinguishes stream processing from the processing of data that goes on within a database-management system. The latter system controls the rate at which data is read from the disk, and therefore never has to worry about data getting lost as it attempts to execute queries.

Streams may be archieved in a large archival store, but we assume it is not possible to answer queries from the archival store. It could be examined only under special circumstances using time-consuming retrieval processes. There is also a working store, into which summaries or parts of streams may be placed, and which can be used for answering queries. The working store might be disk, or it might be main memory, depending on how fast we need to process queries.

But either way, it is sufficiently limited capacity that it cannot store all the data from all the streams.

Architecture of Data-Stream Management System (DSMS)



- Also, it is required when loading meta-information about queries, query plans, stream, inputs and outputs.
- These are held in a system catalog in secondary storage.

Schedular:

- Schedular determines which operator is executed next
- The schedular interacts closely with a query processor.

Query Processor:

- It helps to execute the operator by interacting with scheduler.

QoS Monitoring

- Many systems also include some kind of monitor which gathers statistics about performance, operator output rate, or output delay.
- These statistics can be used to optimize the system execution in several ways.

Query optimizer.

- The throughput of a system can be increased by a load shedder which is stream element selected by a sampling method.
- The load shedder can be a part of query optimizer, a single component, or part of the query execution plan.
- The statistics can be used to re-optimize the current query execution plan and record the operators. For this purpose a query optimizer can be included.

Data Stream :-

- DSMS gets data stream as input.
- Data Stream elements are represented as tuples, which adhere to a relational schema with attributes and values.

Stream Manager. :-

- Wrappers are provided which can receive raw data from its source, buffer and order it by timestamp.
- The task of stream manager is to convert the data to the format of the data stream management system.

Router

- It helps to add tuples or data stream to the queue of the next operator according to the query execution plan.

Queue Manager.

- The management of queues and their corresponding buffers handled by a queue manager.
- The queue manager can also be used to swap data from the queues to a secondary storage, if main memory is full.

System catalog and storage Manager.

- To enable access to data stored on disk many system employ a storage manager which handles access to secondary storage.
- This is used, when persistent data is combined with data from stream sources.

Sampling Data in a Stream

Sampling is the process of extracting reliable samples from a stream.

Sampling methods obtaining a representative Sample

1. Probabilistic sampling is a statistical technique
2. Non-Probabilistic sampling uses arbitrary or purposive (biased) sample selection instead of sampling based on a randomized selection.

Reservoir Sampling Method.

A random sampling method, choosing a sample of limited data items from a list containing a very large number of items randomly.

The list is larger than one that upholds in the main memory.

Concise Sampling

Concise Sampling like a reservoir sampling method, with a difference that a value that appears once is stored as a singleton, whereas a value ~~more~~ that appears more than once is stored as a (value, count) pair.

Inserts a new data item in the sample with a probability of $1/n$.

Filtering Stream

Another common process on streams is selection, or filtering.

We want to accept those tuples in the stream that meet a criteria. Accepted tuples are passed to another process as a stream, while other tuples are dropped.

If the selection criteria is a property of the tuple that can be calculated, then the selection is easy to do.

The problem becomes harder when the criteria involves lookup for membership in a set.

It is especially hard, when that ~~freedom~~ set is too large to store in main memory.

Bloom filtering is the way to eliminate most of the tuples that do not meet criteria.

Bloom Filtering (Bloom Filter)

A bloom filter consists of :-

1. An array of n bits, initially all 0's.
2. A collection of hash functions $h_1, h_2, \ldots, h_k$. Each hash function maps "keys" values of n buckets, corresponding to the n bits of the bit array.
3. A set S of m key values.

The purpose of the Bloom filter is to allow through all stream elements whose keys are in S, while rejecting most of the stream elements whose keys are not in S.

To initialize the bit array, begin with all bits 0. Take each key value in S and hash it using each of the k hash function.

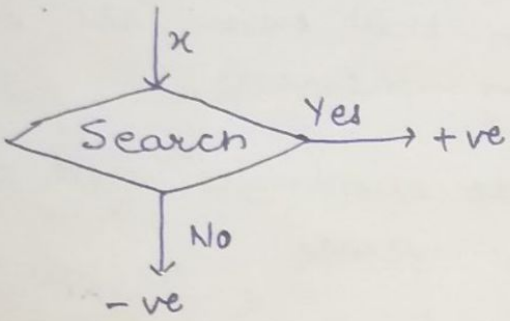
Set to 1 each bit that is $h_i(K)$ for some hash function h_i and some key value K in S.

To test a key K that arrives in the stream, check that all of

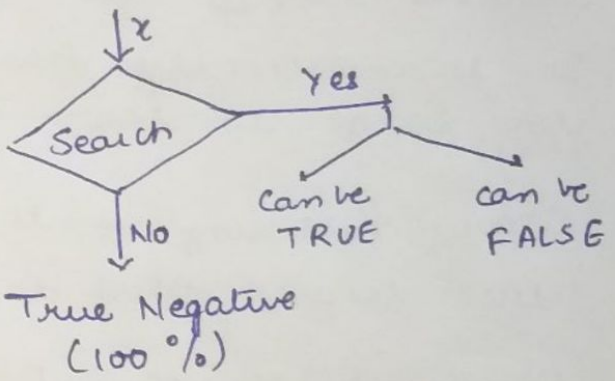
$$h_1(K), h_2(K), \ldots, h_k(K)$$

are 1's in the bit-array. If all are 1's, then let the stream element through, If one or more of these bits are 0, then K could not be in S, so reject the stream element.

Normal Search



Bloom filter



- It was given by Buster-Howard Bloom in 1970.
- a space efficient - probabilistic data structure
- uses to check whether an element is a number of the set.
- False positive matches possible.
- False negative never possible

Example :-

- Facebook user exist check
- G-mail uses exist check.
- Cache exist.
- Packet Routing
- Database query.

Example :-

$m = 5$ — size of array.

$h1(x) = x \bmod 5$

$h2(x) = 2x + 3 \bmod 5$ } hash function

Soln :-

0	0	0	0	0
0	1	2	3	4

Insert 9

$h1(x) = 9 \bmod 5 = 4$

$h2(x) = 2(9)+3 \bmod 5 = 1$.

Now we will put 1 at the 4th and 1st index of the array.

0	1	0	0	1
0	1	2	3	4

Insert 11

$h1(x) = 11 \bmod 5 = 1$

$h2(x) = 2(11)+3 \bmod 5 = 0$

Now we will put 1 at index 0 and 1 of the array, but 1st index already has 1.
So the array will be :-

1	1	0	0	1
0	1	2	3	4

Query - 15 :-

Now let's check this filter for 15.

$h1(15) = 15 \bmod 5 = 0$

$h2(15) = 2(15) + 3 \bmod 5 = 3$.

Now check the position or 0 and 3 index of the array, if all are 1, then it is a +ve case, else it is a negative case.

1	1	0	0	1
0	1	2	3	4

X

As we see we have 1 at 0th index, but not at the 3rd index, so it is a negative case.

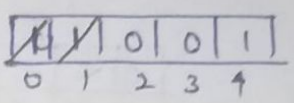
So, surely, 15 is not present.

Now let's check for another element.

Query - 16.

$h_1(16) = 16 \bmod 5 = 1$

$h_2(16) = 16(2) + 3 \bmod 5 = 0$.



As we can see that at 0th and 1st index, we have 1, so it is a positive case

But we know that 16 was not in the samples when we created the filter. Hence, it is a false positive case.

Try it for other examples :-

Q:- $m = 10$

$h_1(x) = (3x + 7) \bmod 8$.

$h_2(x) = (2x + 3) \bmod 5$.

Samples :-

8, 10

Query :-

48, 7

Check whether 48 and 7 are present or not.

Counting Distinct Elements in a Stream

In this we find out how many different elements have appeared in the stream, counting either from the beginning of the stream or from some known time in the past.

We can count the number of distinct element in a stream by using Flajolet-Martin Algorithm :-

Flajolet Martin Algorithm.

- Flajolet - Martin Algorithm approximate the number of unique objects in the stream or a database in one pass.
- If the stream consist n elements with m of them unique, this algorithm runs in $O(n)$ time and needs $O(\log(m))$ memory..
- Space-consumption logarithmic in the maximal number of possible distinct elements in the stream.

Let us take an example to under F-M algorithm :-

Q:. Determine the distinct element in the stream using FM.

Input stream of integers X =
{1, 3, 2, 1, 2, 3, 4, 3, 1, 2, 3, 1}

Hash function, $h(x) = 6x + 1 \bmod 5$.

Soln.

Step-1:-
Calculate hash function $h(x)$
for the given input stream =
{1, 3, 2, 1, 2, 3, 4, 3, 1, 2, 3, 1}.

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

$h(1) = 2$
$h(3) = 4$
$h(2) = 3$
$h(1) = 2$
$h(2) = 3$
$h(3) = 4$

$h(4) = 0$
$h(3) = 4$
$h(1) = 2$
$h(2) = 3$
$h(3) = 4$
$h(1) = 2$

Step-2.

Evaluate binary bits.

- For every hash function calculated, write the binary equivalent for the same.

$h(1) = 2 = 010$
$h(3) = 4 = 100$
$h(2) = 3 = 011$
$h(1) = 2 = 010$
$h(2) = 3 = 011$
$h(3) = 4 = 100$

$h(4) = 0 = 000$
$h(3) = 4 = 100$
$h(1) = 2 = 010$
$h(2) = 3 = 011$
$h(3) = 4 = 100$
$h(1) = 2 = 010$

Step-3.

Counting the trailing zeroes.

Now we will count the number of trailing zeros in each hash function bit.

$h(1) = 2 = 010 = 1$
$h(3) = 4 = 100 = 2$
$h(2) = 3 = 011 = 0$
$h(1) = 2 = 010 = 1$
$h(2) = 3 = 011 = 0$
$h(3) = 4 = 100 = 2$

$h(4) = 0 = 000 = 0$
$h(3) = 4 = 100 = 2$
$h(1) = 2 = 010 = 1$
$h(2) = 3 = 011 = 0$
$h(3) = 4 = 100 = 2$
$h(1) = 2 = 010 = 1$

Step-4:

Calculate the number / count of distinct elements.

- From the binary equivalent trailing zero values, write the number of (maximum) of trailing zeros.
- The value of $r = 2$.
- The distinct value $R = 2^r$

$$\boxed{R = 2^2 = 4}.$$

Therefore, $R = 4$. means there are 4 distinct elements in the given input stream.

Try it for other examples:-

Q-1:-

I/P stream = {1, 4, 2, 1, 2, 4, 4, 4, 1, 2, 4, 1, 7}

$h(x) = (3x + 1) \bmod 5$.

Q-2:-

I/P stream = {4, 2, 5, 9, 1, 6, 3, 7}

$h(x) = (x + 6) \bmod 32$.

Estimating moments

Estimating moments is a generalization of the problem of counting distinct elements in a stream.
The problem, called computing "moments" involves the distribution of frequencies of different elements in the stream.

Goal :- Computing distribution of frequencies of different elements in stream.

Example :-

- 1st moment = Sum of all m_i = count the no. of elements.
  = length of the stream
- 2nd moment = Sum of all m_i^2
  = Surprise number (S)
  (a measure of how uneven the distribution is)
- 0th moment = number of distinct elements

$$K^{th} \text{ moment} = \sum_{i \in A} (m_i)^K$$

Example :-

Consider input stream
{5, 5, 5, 5, 5}.

0th moment = $(5)^0 = 1$ (No of distinct element)

1st moment = $(5)^1 = 5$ (length of the stream)

2nd moment = $5 \times 5^2 = 5 \times 25 = 125$ (Surprise Number)

- {10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9}.

2nd moment = $10^2 + 10(9)^2 = 10^2 + 10 \times 81 = 910$.

Alon-Matias Szegedy Algorithm (AMS)

– for second moments.

1. Let us assume that a stream has a particular length n.
2. Suppose we do not have enough space to count all the m_i's for all the element of the stream.
3. We can still estimate the 2nd moment of the stream using limited amount of space, the more space we use, the more accurate the answer will be. We compute some number of variables.
4. For each variable X, we store:
   - A particular element of the universal set, which we refer do as X. element.
   - An integer X. value, which is the value of the variable. To determine the value of a variable X, we choose a position in the stream between 1 and n, uniformly and at random.
     Set X. element to be the element found there, and initialize X. value to 1. As we read the stream, add 1 to X. value each time we encounter another occurance of X. element.

In Simple words, AMS Algorithm.

- N observation in the stream
- choose K random positions pj∈[1, 2, ... N]
- When reaching position pj :
  - → store object at position
  - → store counting occurences of object.
- Estimate :- M2 = N/K (sum of (2mi - 1)

For example :-

Consider this stream

{a, b, c, b, d, a, c, d, a, b, a, c, a, a, b}

By using 2nd moment :-

2nd moment = $a^2 + b^2 + c^2 + a^2$

= $5^2 + 4^2 + 3^2 + 3^2$ = **59**

Problem solved by AMS Algorithm.

Given stream = {a, b, c, b, d, a, c, d, a, b, d, c, a, a, b}

Soln. :-

length of the stream n = 15.

choose 3 random positions with different values say c, d, a

{a, b, (c), b, d, a, c, (d), a, b, d, c, (a), a, b}.

↑ X_1. element ↑ X_2. element ↑ X_3. element.

Count the number of times of occurances from that positions

for X_1. element (c)

{a, b, ⓒ, b, d, a, c, ⓓ, a, b, d, c, ⓐ, a, b}

↓

X_1 value = 1

Set X. value = 1, now increase value by 1 each time we encounter another occurence of c.

{a, b, ⓒ, b, d, a, c, d, a, d, c, a, a, b}

↓ X_1. value = 1

↓ X_1. value = 2

↓ X_1. value = 3

We we repeat the same process for all 3 elements c, d, a.

{a, b, ⓒ, b, ⓓ, a, ⓒ, ⓓ, a, d, c, ⓐ, a, b}.

↓ X_1. value = 1

↓ X_1. value = 2

↓ X_2. value = 2

↓ X3. value = 1

↓ X_2. value = 1

↓ X_1. value = 3

↓ X3 value = 2.

So, we get.

$X_1 = (c, 3)$

$X_2 = (d, 2)$

$X_3 = (a, 2)$

Calculate the estimate

Estimate = (n x (2 * X-value - 1))

X_1 estimate = 15 x (2 * 3 - 1)

= 75

X_2 estimate = 15 x (2 * 2 - 1)

= 45.

X_3 estimate = 15 * (2 * 2 - 1)

= 45.

Calculate the average of X_1, X_2, X_3 :-

Avg = { Sum of estimates } / 3

$= \frac{75 + 45 + 45}{3}$

= 55

55 is somewhat close to the answer 59.

Advantages of AMS Algo

→ Simple

→ Needs to store only K-counters

→ Larger the value of K → Accuracy increases

Disadvantages of AMS Algo.

→ N is not known.

Sliding windows

A useful model of stream processing is that queries are about a window of length of N - the N most recent elements received.

Interesting case :- N is so large it cannot be stored in main memory, or even on disk.
Or there are so many streams that windows for all cannot be stored.

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

← Past Future →

Counting bits

Problem :- Given a stream of 0's and 1's, be prepared to answer queries of the form " how many 1's in the last K bits? " where $K \leq N$.

Obvious solution :- Store the most recent N bits
When new bit comes in, discard the N+1st bit.

You can't get an exact answer without storing the entire window.

Real Problem:- what if we can't afford to store N-bits?

eg. we are processing 1 billion streams and N = 1 billion, but we're happy with an approximate answer.

GIM Algorithm.

(Counting the number of 1's in the data streams)

→ Each bit ~~ba~~ stream has a timestamp, for the position at which it arrives.

→ The first bit has time stamp 1, the second bit has time stamp 2, and so on.

→ The position are recognized with the window size N.

→ The timestamp is represented with modulo N, and are represented as $\log_2 N$ bits.

→ The windows are divided into buckets consisting of :-

- The time stamp at its right end.
- The number of ones must be in the power of 2, which are referred to as size of bucket.
- Eg 1001011 → the bucket size is 4.

→ The right side of the bucket should always start with 1.

→ Every bucket should have at least one 1, else no bucket can be formed.

→ All bucket sizes should be a power of 2.

→ The buckets cannot decrease in siz as we move to the left

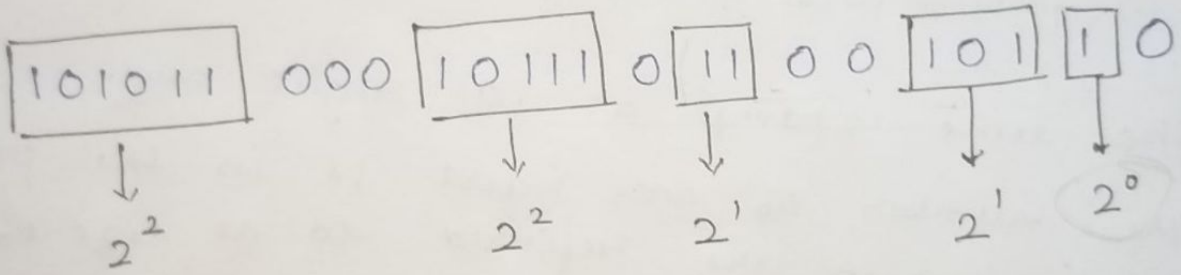
→ There are one or two buckets of any given size, up to maximum size.

- This algorithm uses $O(\log^2 N)$ bit to repres a window of N bit.
- Allows to estimate the number of 1's in the window.
- With an error of no more than 50%.
- Each bit that arrive has a timestamp.
- The windows are divided into buckets consisting of 1's and 0's.

Example. 1:-

Input Stream :- 10101100010111011001010

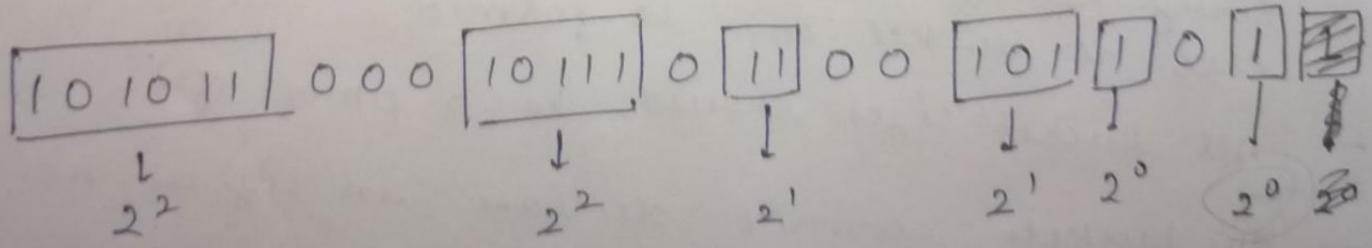
N = 24 (window size)



Example - 2:

Given I/P stream :- 10101100010111011001010

Input bit stream :- 1



Updating buckets

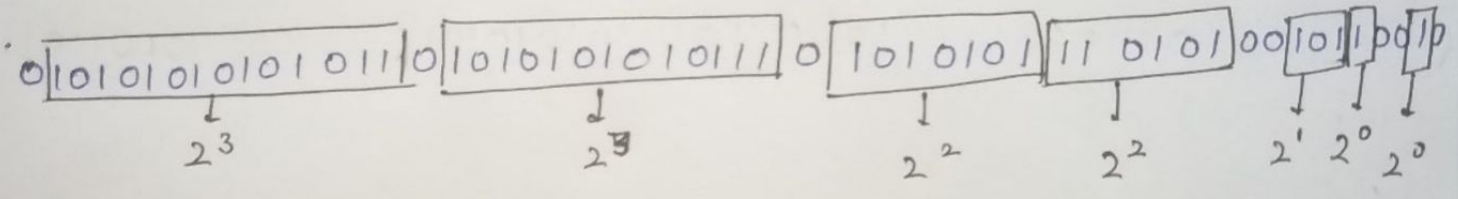
When a new bit comes in, drop the last (oldest) bucket if its end time is prior to N bits times unit before the current time

→ If the current bit 0, no other changes are needed

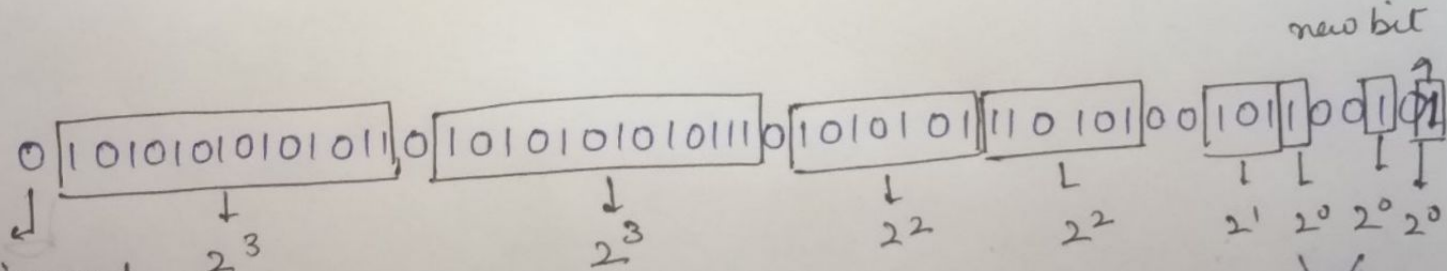
→ <u>If the current bit is 1</u> :-

- Create a new bucket of size 1, for first this bit.
  End timestamp = current timestamp.
- If there are now three buckets of size 1, combine the oldest two buckets into a size 2.
- If there are now three buckets of size 2, combine the oldest two into a bucket into a size 4.
- And so on.

<u>Example 3.</u>

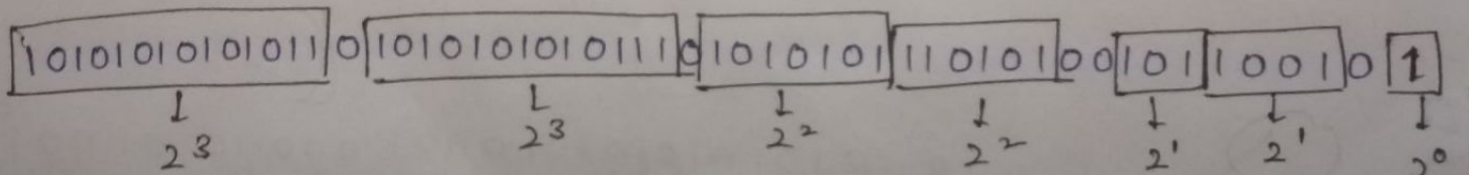


Adding 1 at the end.



discard.

merge.
As we have
3(2)^0



Now, no rule is violated, and that's how we have added 1 to the input data stream.

Decaying window algorithm

This algorithm allows you to identify the most popular elements (trending) in an incoming data stream.

The decaying window algorithm not only tracks the most recurring elements in an incoming data stream, but also discounts any random spikes or random requests that might have boosted an element's frequency.

In a decaying window, you assign a score or weight to every element of the incoming data stream.

Further, you need to calculate the aggregate sum for each distinct element by adding all the weights assigned to that element. The element with the highest total score is listed as trending or the most popular.

1. Assign each element with a weight / score.
2. Calculate aggregate sum for each distinct element by adding all the weights assigned to that element.

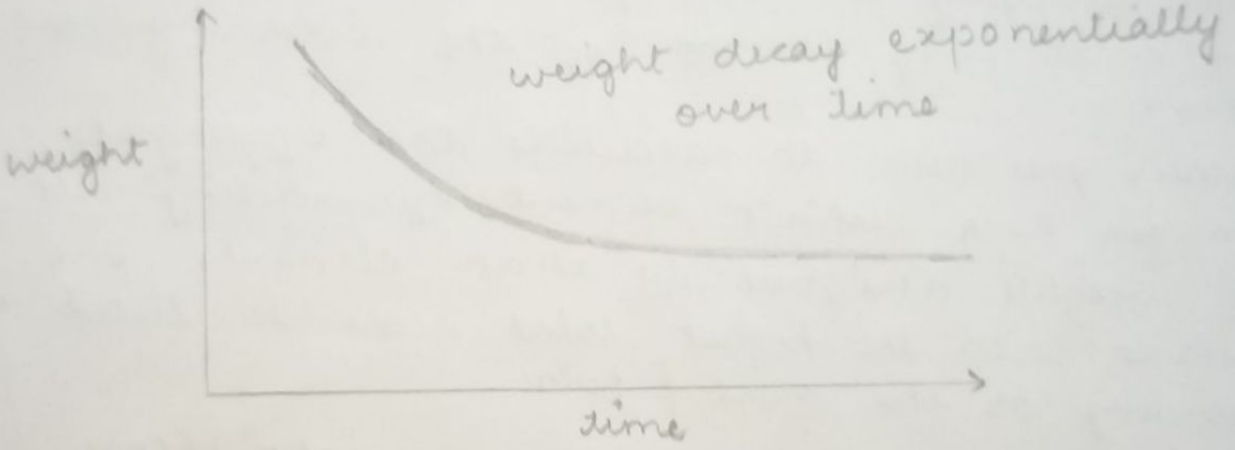
In decaying window algorithm, you assign more weight to newer elements.

For a new element you first reduce the weight of all the existing elements by a constant factor k and the assign the new element with a specific weight. The aggregate sum of the decaying exponential weights can be calculated using the following formula:-

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

Here, c is usually a small constant of the order 10^{-6} or 10^{-9}, whenever a new element, say a_{t+1}, arrives in the data stream you perform the following step to achieve an updated sum :-

1. Multiply the current sum / score by the value (1-c).
2. Add the weight corresponding to the new element.



In a data stream consisting of various elements, you maintain a seperate sum for each distinct element. For every in coming element, you multiply the sum of all the existing elements by a value of (1-c). Further, you add the weight of the in coming element to its corresponding aggregate sum.

A threshold can be kept to, ignore elements of weight less than that.

Finally, the element with the highest aggregate score is listed as the most popular element.

example:-

Consider a sequence of twitter tags below:-

fifa, ipl, fifa, ipl, ipl, iplt, fifa

Also, let's say each element in sequence has weight of 1.

Let's c be 0.1

The aggregate sum of each tag in the end of the above stream will be calculated as below

fifa

fifa - 1 * (1-0.1) = 0.9

ipl - 0.9 * (1-0.1) + 0 = 0.81 (adding 0 because current tag is different than fifa)

fifa - 0.81 * (1-0.1) + 1 = 1.729 (adding 1 because current tag is fifa only)

ipl - 1.729 * (1-0.1) + 0 = 1.5561

ipl - 1.5561 * (1-0.1) + 0 = 1.4005

ipl - 1.4005 * (1-0.1) + 0 = 1.2605

fifa - 1.2605 * (1-0.1) + 1 = 2.135

iplt

fifa - 0 * (1-0.1) = 0

iplt - 0 * (1-0.1) + 1 = 1

fifa - 1 * (1-0.1) + 0 = 0.9 (different tag)

ipl - 0.9 * (1-0.1) + 1 = 1.81

iplt - 1.81 * (1-0.1) + 1 = 2.7919

ipl - 2.7919 * (1-0.1) + 1 = 3.7647

fifa - 3.7647 * (1-0.01) + 0 = 3.7264

In the end of the sequence, we can see the score of fifa is 2.135 but ipl is 3.7264.

So, ipl is more trending then fifa. Even though both of them occurred same number of times in input their score is still different.

Advantages of Decaying window

1. Sudden spikes or spam data is taken care.
2. New element is given more weight by this mechanism, to achieve right trending output.

Real Time Analytics

- Refers to the finding meaningful patterns in the data at the actual time of receiving.
- Real-Time Analytics Platform (RTAP) analyses the data, correlates, and predicts the outcomes in the real time.

Real Time Analytics Platform :-

- Manages and processes data and helps timely decision making.
- Helps to develop dynamic analysis applications.
- Leads to evolution of busines intelligence.

Widely Used RTAP's

1. Apache SparkStreaming :- a big data platform for data stream analytics in real time.
2. Cisco Connected Streaming Analytics (CSA) — a platform that delivers insights from high velocity streams of live data from multiple sources and enables immediate action.
3. Oracle Stream Analytics (OSA) - a platform that provides a graphical interface to "Fast Data".
4. SAP HANA — a streaming analytics tool which also does real-time analytics
5. SQL streamBlaze - an analytics platform, offering a real-time, easy-to-use and powerful visual development environment

for developers and analysts.

6. TIBCO StreamBase - streaming analytics, which accelerates action in order to quickly build applications.

7. Infamatica - a real-time data streaming tool which transforms a torrent of small messages and events into unprecedented business agility.

8. IBM Stream Computing - a data streaming tool that analyzes a broad range of streaming data - unstructured text, video, audio, geospatial, sensor - helping organizations spot the opportunities and risks and make decisions in real time.

RTAP Applications :-

1. Fraud detection systems for online transactions.
2. Log analysis for understanding usage pattern.
3. Click analysis for online recommendations.
4. Social media analytics
5. Push notifications to the customers for location-based advertisements for retails
6. Action for emergency services such as fires and accidents in an industry.
7. Any abnormal measurements require immediate reactions in healthcare monitoring.

RTAP Case Studies

→ Stock Market Prediction

- Trading is the process of buying and selling of financial instruments stock market for the trading
- One of the most important sources for companies to raise money allows business to go public, or raise additional capital for expansion.
- Predicting stock performance is a very large and profitable area of study.
- Many companies have developed stock predictors based on neural networks.
- This technique has proven successful in aiding the decision of investors.
- Can give an edge to beginning investors who don't have a lifetime of experience.

What is stock market and how it works?

- The stock market works like an auction where investors who buy and sell shares of stocks.
- These are a small piece of ownership of a public corporation.
- Stock prices usually reflect investors opinion of what the company's earnings will be.
- Stock trading involves buying and selling stocks frequently in an attempt to time the market
- The goal of stock traders is to capitalize on short-term market events to sell stocks for a profit, or buy stocks at a low.
- Investors who trade stocks do extensive research, often devoting hours a day to following the market.

Stock Market Prediction

- Collect a sufficient amount of historical stock data.
- Using the data train the neural network.
- Once trained, the neural network can be used to predict stock behavior
- Forecasting stock market price has always been challenging task for many business analyst and researchers.
- In fact, stock market price prediction is an interesting area of research for investors.
- For successful investment lot many investors are interested in knowing about future situation of market.
- Effective prediction system indirectly help traders by providing supportive information such as the future market direction.

What is Stock Market Prediction?

Stock market prediction is the act of trying to determine the future value of company stock or other financial instrument traded on an exchange The successful prediction of a stock's future price could yield significant profit.
Generally modeled as classification or regression task predict a binary or ordinal label features:

→ 'Negotiation is important.
→ Using all words (in naive bayes) works well for some tasks.
→ Finding subset of words may help in other tasks.
→ Hand-Built polarity lexicons.
→ Use seeds and semi-supervised learning to induce lexions.

The most important for predicting stock market price are neural networks because they are able to learn non-linear-mapping between inputs and outputs.

It may be possible to perform better than traditional analysis and other computer-aided / based methods with the neural networks ability to learn - nonlinear, chaotic - systems.