Data Stream Mining

1. Data Streams

- In many data mining situations, we do not know the entire data set in advance

- Stream management is Important when the input rate is controlled externally:
  + Google queries
  + Twitter or Facebook status updates

- We can think of data as infinite and non-stationary (the distribution changes over time)

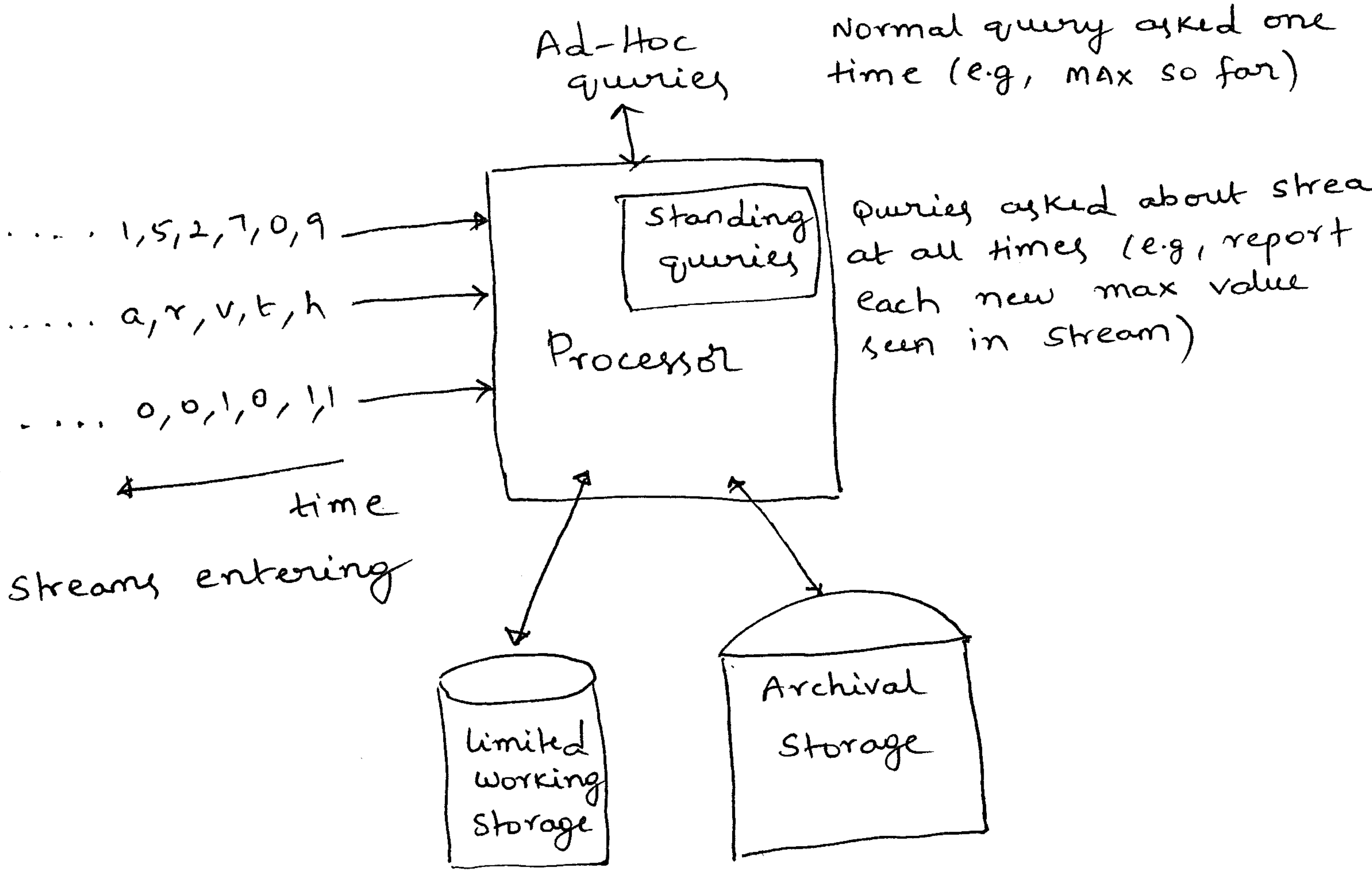
2. The Stream model

- Input elements enter at a rapid rate, at one or more ports (i.e, streams)
  + We call elements of the stream tuples

- The system cannot store the entire stream accessibly

- Question: How do you make critical calculations about the stream using a limited amount of memory?

3. General Stream processing Model



4. Applications

- Mining query streams → Google Trends
  + Google wants to know what queries are more frequent today than yesterday.
- Mining click streams
  + Yahoo wants to know which of its pages are getting an unequal no of hits in the past hour → Breaking news!
- Mining social network news feeds
  + looking for trending topics on Twitter, Facebook.

Filtering Data Streams:

Each element of the stream is a tuple
Given a list of Keys S
Determine which tuples of stream are in S

$\Longrightarrow$ Bloom Filters

Motivation:

- Consider a web crawler
- It keeps centrally a list of all the URL's it has found so far
- It assigns these URL's to a number of parallel tasks; these tasks stream back the URL's they find in the links they discover on a page.
- It needs to filter out those URL's it has seen before.

Role of the Bloom Filter:

- A Bloom filter placed on the stream of URL's will declare that certain URL's have been seen before.
- Others will be declared new, and will be added to the list of URL's that need to be crawled.

Unfortunately, the Bloom filter can have false positives.

- It can declare a URL has been seen before when it hasn't.
- But if it says "never seen" then it is truly new.

How a Bloom Filter works?

- A Bloom filter is an array of bits, together with a number of hash functions.
- The argument of each function (hash) is a stream element, and it returns a position in the array. (Important, hash functions are independent)
- Initially, all bits are 0.
- When input 'x' arrives, we set to 1 the bits h(x), for each hash function h.

Example : Bloom Filter :

use $N = 11$ bits for our filter.

Stream elements = integers

use two hash functions :

$h_1(x)$ = Take odd-numbered bits from the right in the binary representation of x

$\Downarrow$

Treat it as an integer i

$\Downarrow$

Result is i modulo 11

$h_2(x)$ = Same, but even-numbered bits.

Stream Element	h_1	h_2	Filter Contents
			00000000000
25 = 11001	5	2	00100100000
159 = 10011111	7	0	10100101000
585 = 1001001001	9	7	10100101010

Bloom Filter Lookup :

Suppose elemeny 'y' appears in the stream, and we want to know if we have seen 'y' before.

Comput $h(y)$ for each hash function

If all the resulting bit positions are "1", say we have seen 'y' before
(False positives : Some other combination of elements 'g' may have turned these bits ON)

If at least one of these positions is '0', say we have not seen 'y' before.
(NO, False negatives)

$\boxed{\text{Intersection of Two sets ??}}$

Example lookup :

Suppose we have the bloom filter as before, and we have set the filter to

1 0 1 0 0 1 0 1 0 1 0

Lookup element $y = 118 = 1110110$ (binary)

$h_1(y) = 14$ modulo $11 = 3$

$h_2(y) = 5$ modulo $11 = 5$

bit 5 is ON, but bit 3 is OFF
$\Rightarrow$ we are sure y is not in set.

Performance of Bloom Filters:

Prob of a false-positive depends on the density of 1's in the array and the number of hash functions.

$$= (\text{fraction of 1's})^{\#\text{ hash functions}}$$

The number of 1's is approximately the no of elements ~~times~~ inserted times the no of hash functions

- But collisions lower that number

Throwing Darts:

Turning random bits from 0 to 1 is like throwing "d" darts at "t" targets at random.

"d" = # Elements * # hash functions

"t" ⇓ one target for each bit of array

How many targets are hit by at least one dart ??

Prob a given target is hit by a given dart

$$= \frac{1}{t}$$

Prob that none of the "d" darts hits a given target is

$$(1 - 1/t)^d$$

Rewrite as

$$(1 - 1/t)^{t \cdot (d/t)}$$

$$= e^{-d/t}$$

[If t is large $(1 - 1/t)^t = 1/e \approx 0.37$

Example : Throwing Darts

Suppose we use an array of 1 Billion bits

5 Hash Functions

100 million elements

$t = 10^9 \qquad d = 5 * 10^8$

The fraction of 0's that will remain

$= e^{-1/2} = 0.607$

Density of 1's = 0.393

Prob of a false positive $= (0.393)^5$

$= 0.00937$