Mining Data Streams

First 7 slides are from below

Mining of Massive Datasets
Jure Leskovec, Anand Rajaraman, Jeff Ullman
Stanford University
http://www.mmds.org

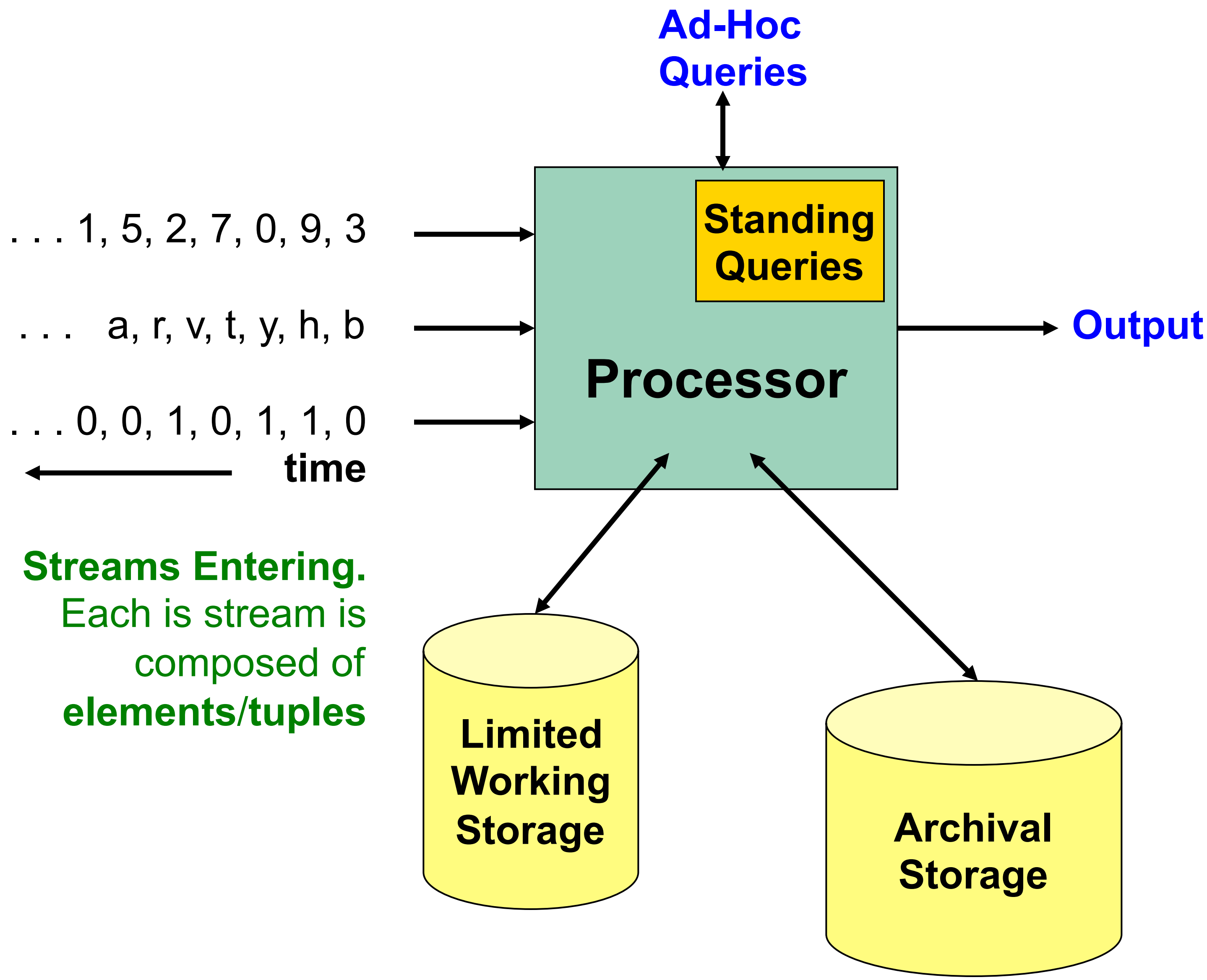
Data Streams

- **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 the **data** as **infinite** and **non-stationary** (the distribution changes over time)

The Stream Model

- Input **elements** enter at a rapid rate,
  at one or more input ports (i.e., **streams**)
  - **We call elements of the stream tuples**
- **The system cannot store the entire stream accessibly**
- **Q: How do you make critical calculations about the stream using a limited amount of (secondary) memory?**

General Stream Processing Model



Sources of this kind of data

- **Sensor data**
  - E.g., millions of temperature sensors deployed in the ocean
- **Image data from satellites, or even from surveillance cameras**
  - E.g., London
- **Internet and Web traffic**
  - Millions of streams of IP packets
- **Web data**
  - Search queries to Google, clicks on Bing, etc.

Problems on Data Streams

- **Types of queries one wants on answer on a data stream:**
  - **Filtering a data stream**
    - Select elements with property ***x*** from the stream
  - **Counting distinct elements**
    - Number of distinct elements in the last ***n*** elements of the stream
  - **Estimating moments**
    - Estimate avg./std. dev. of last ***n*** elements
  - **Finding frequent elements**

Applications (1)

- **Mining query streams**
  - 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 unusual number of hits in the past hour
- **Mining social network news feeds**
  - E.g., look for trending topics on Twitter, Facebook

Applications (2)

- **Sensor Networks**
  - Standard deviation of temperature
- **IP packets monitored at a switch**
  - Gather information for optimal routing
  - Detect denial-of-service attacks

Model

- Input: sequence of T elements $a_1, a_2, \ldots a_T$ from a known universe U, where $|U|=u$.

Goal: perform a computation on the input, in single left to right pass using

- Process elements in real time
- Can't store the full data => minimal storage requirement to maintain working "summary".

What can we compute over a stream ?

32, 112, 14, 9, 37, 83, 115, 2,

Some functions are easy: min, max, sum, …

We use a single register s, simple update:

- Maximum: **Initialize** $s \leftarrow 0$

  For element x, $s \leftarrow \max s, x$
- Sum: **Initialize** $s \leftarrow 0$

  For element x, $s \leftarrow s + x$

Today

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

- Heavy hitters: keys that occur lots and lots of times
- The number of *distinct* keys in the stream
  - Application of MinHash to computation of document similarity
- Second frequency moment.

Themes

- Cool applications of hashing
- Can compute interesting global properties of a long stream, with only one pass over the data, while maintaining only a small amount of information about it. We call this small amount of information a **sketch**

Heavy hitters: keys that occur many times

Some applications:

- Determining popular products
- Computing frequent search queries
- Identifying heavy TCP flows
- Identifying volatile stocks

Heavy hitters: keys that occur many times

Special case: an array of integers A[1..T] with **a majority element.**

Find majority element in single pass over data using sublinear auxiliary space?

Heavy hitters: a majority element.

guaranteed to occur > T/2 times

```
counter:= 0; current := NULL
for i := 1 to n do
    if counter == 0, then
        current := A[i];
        counter++;
    else if A[i] == current then
                Counter ++
            Else counter - -
return current
```

Heavy hitters: beyond majority

Find all elements that occur at least ϵT times.

provably impossible in sublinear auxiliary space

So what do we do?

Counting Distinct Elements

32, 12, 14, 32, 7, 12, 32, 7, 6, 12, 4,

Applications:

- IP Packet streams: Number of distinct IP addresses or IP flows (source+destination IP, port, protocol)
  - Anomaly detection, traffic monitoring
- Search: Find how many distinct search queries were issued to a search engine (on a certain topic) yesterday
- Web services: How many distinct users (cookies) searched/browsed a certain term/item
  - advertising, marketing, trends

Second frequency moment [AMS]

Measures how uneven the distribution of elements in the stream is

In database context: the size of a “self-join” – the size of the table you get when you join a relation with itself on a particular attribute.

Takeaway

- Can do amazing things with randomness
- Can implement many of those amazing things with limited randomness