Data Streams

- Data that are *continuously* generated by many sources at very *fast* rates
- Examples:
  - Google queries
  - Twitter feeds
  - Financial markets
  - Internet traffic
- We do not have complete information (e.g., size) on the entire dataset
- Convenient to think about data as *infinite*
- Question: “How do you make critical calculations about the stream using limited amount of memory?”
Applications

- 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, etc.
Applications (cont’d)

- Sensor networks
  - Many sensors feeding into a central controller

- Telephone call records
  - Data feeds into customer bills as well as settlements between telephone companies

- IP packets monitored at a switch
  - Gather information for optimal routing
  - Detect denial-of-service attacks

From http://www.mmds.org
One Pass Model

- Given a data stream \( D = x_1, x_2, x_3 \ldots \)

- At time \( t \), we observe \( x_t \)

- For analysis, observed \( D_t = x_1, x_2, \ldots, x_t \) so far
  (don’t know how many points we will observe in advance)

- We have a limited memory budget, i.e., \( \ll t \)

- Task: at any point of time \( t \), compute some function of \( D_t \)
  (i.e., \( f(D_t) \))

- What is an approach to approximating \( f(D_t) \), given \( x_t, x_{t-1}, \ldots \)?
Basic Question

- If we can get a representative *sample* of the data stream, then we can do analysis on it

- How to sample a stream?

- Sampling is . . .?
Sampling (example 1)

- Suppose we have seen $x_1, \ldots, x_{1000}$
- Memory can only store sample size of 100
- Task: sample 10% of the stream
- How?
Sampling (example 1)

- Suppose we have seen $x_1, \ldots, x_{1000}$
- Memory can only store sample size of 100
- Task: sample 10% of the stream
- How?
  - Take every 10th element
  - $q \sim \{1, 2, \ldots, 10\}$, take every $q + 1$ element
- Issues?
Sampling (example 2)

- Dataset:
  - # of unique elements = $U$
  - # of (pairwise) duplicate elements = $2D$
  - total # of elements: $N = U + 2D$

- Fraction of duplicates: $\alpha = \frac{2D}{U + 2D}$

- Take 10% sample and estimate $\alpha$

- Questions:
  - What is the probability that a pair of duplicate items is in the sample?
  - What happens to the estimation?
Sampling From Stream

Task: sample $s$ elements from a stream; at element $x_t$, we want:

- Every element was sampled with probability $\frac{s}{t}$
- We have $s$ number of samples

Can this be accomplished? If yes, then how?

Let us think through this . . .
Reservoir Sampling

- Sample size $s$

- Algorithm:
  - observe $x_t$ from stream
  - if $t < s$, then add $x_t$ to reservoir
  - else with probability $\frac{s}{t}$:
    - uniformly select an element from reservoir
    - and replace it with $x_t$

- Claim: at any time $t$, any element in $x_1, x_2, \ldots, x_t$ has exactly $\frac{s}{t}$ chance of being sampled
Reservoir Sampling - Proof by Induction

- Inductive hypothesis: after observing $t$ elements, each element in the reservoir was sampled with probability $\frac{s}{t}$

- Base case: first $t$ elements in the reservoir was sampled with probability $\frac{s}{t} = 1$

- Inductive step: element $x_{t+1}$ arrives . . .

work on the board . . .
Weighted Reservoir Sampling

- Each element $x_i$ has a weight $w_i > 0$

- Task: sample elements from the stream, such that:
  - at time $t$, every element $x_i$ was sampled with probability
    $$\frac{w_i}{\sum_i w_i}$$
  - have $s$ elements

- Reservoir sampling is special case ($w_i = 1$)
Weighted Reservoir Sampling

Solution by (Pavlos S. Efraimidis and Paul G. Spirakis, 2006)

- Observe $x_i$

- Sample $r_i \sim \mathcal{U}(0, 1)$

- Set score $\sigma_i = \frac{1}{w_i} r_i$

- Keep elements $(x_i, \sigma_i)$ with highest $s$ scores as sample
Weighted Reservoir Sampling

- Implementation considerations:
  - Use heap to maintain top scores \((x_i, \sigma_i)\); \(O(\log(s))\) time complexity
  - \(\sigma_i \in (0, 1) \Rightarrow\) top scores get closer to 1, which becomes hard to distinguish
Weighted Reservoir Sampling

Lemma: Let $U_1$ and $U_2$ be independent random variables with uniform distributions in $[0, 1]$. If $X_1 = (U_1)^{1/w_1}$ and $X_2 = (U_2)^{1/w_2}$, for $w_1, w_2 > 0$, then

$$\Pr[X_1 \leq X_2] = \frac{w_2}{w_1 + w_2}.$$ 

Partial proof:

$$\Pr[X_1 \leq X_2] = \Pr[(U_1)^{1/w_1} \leq (U_2)^{1/w_2}]$$

$$= \Pr[(U_1) \leq (U_2)^{w_1/w_2}]$$

$$= \int_{U_2=0}^{1} \int_{U_1=0}^{U_2^{w_1/w_2}} dU_1 dU_2 = \ldots = \frac{w_2}{w_1 + w_2}.$$