Streaming Model

- Input consisting of N items $e_1, \dots, e_N$
  examples: numbers, points in metric space, rows of a matrix, edges of a graph
- Items arrive sequentially
- N is not known a priori
- Goal: compute some (problem-specific) function $f(e_1, \dots, e_N)$
  - using small amount of memory $B \ll N$ bits
- For now: Consider only "additions" of items, but model in principle also allows removals

Sampling from a Stream

Basic Question: Sample one item from the stream uniformly at random

↳ Solution: Reservoir Sampling

Algorithm: Reservoir Sampling ($k=1$)

Initialize sample to be empty

Initialize counter $t \leftarrow 0$

For each item x in the stream do

$t \leftarrow t+1$

With probability $\frac{1}{t}$:

Sample $\leftarrow x$ ← "Sample is updated"

return sample

Thm: After N items have been processed, the sample returned by the algorithm is a uniform sample from the stream, i.e., for any $1 \leq i \leq N$, $\Pr[e_i \text{ is the returned sample}] = \frac{1}{N}$

Proof: Idea: Proof by induction on length t of the stream so far

Claim: After the first t items have been processed, for any $1 \leq i \leq t$, $\Pr[e_i \text{ is the current sample}] = \frac{1}{t}$

Base Case ($t=1$) e_1 is stored with probability 1 ✓

Inductive Step ($t \to t+1$)

Assume that claim holds for t, i.e., for any $1 \le i \le t$,

$\Pr[e_i \text{ is sample (after } t \text{ items have been processed)}] = \frac{1}{t}$

Consider now the $(t+1)$-st item being processed by the algorithm

prove claim for item e_{t+1}

prove claim for item e_i, $i \le t$

$\Pr[e_{t+1} \text{ is the sample at the end of the loop}] = \frac{1}{t+1}$

$\Pr[e_i \text{ is the sample at the end of the loop}] =$

$= \Pr[e_i \text{ is the sample at beginning of the loop}] \cdot \Pr[\text{sample not updated}]$ (= independence!)

$= \frac{1}{t}$ by IH $\qquad (1 - \frac{1}{t+1})$

$= \frac{1}{t}\left(1 - \frac{1}{t+1}\right) = \frac{1}{t} \cdot \left(\frac{t+1}{t+1} - \frac{1}{t+1}\right) = \frac{1}{t} \cdot \frac{t}{t+1} = \frac{1}{t+1}$ □

sampling k items with replacement: just run k independent instances of single-item reservoir sampling "in parallel"

sampling k items without replacement

↳ variant of algorithm above