# Streaming

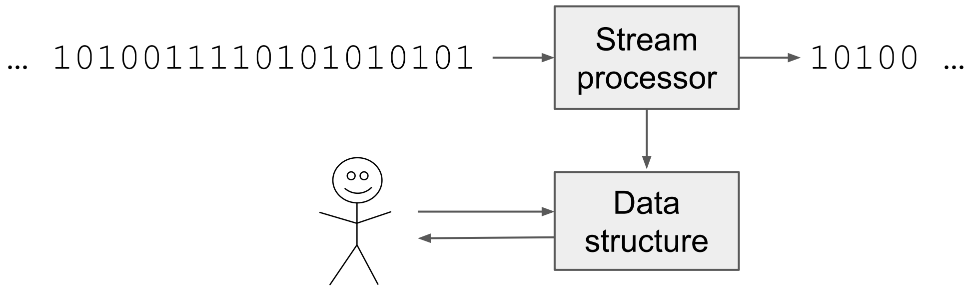
02807 Computational Tools for Data Science

# Today

- The streaming model
- Computing the majority element of a stream
- Simple random sampling in a stream

# The streaming model

- Elements of the stream arrive one at a time
- Stream is infinite
- Stream processor updates a data structure
- Queries are sent to data structure



# The streaming model (cont.)

- The data structure must use limited space
  - We cannot store all elements we see!
- The processor has limited time to update the data structure
  - Typically, we want to update x<<k places, where k is the size of the data structure
- We allow approximate answers from the data structure
  - “How many distinct elements have we seen?”
  - “What are the most frequent elements seen?”
  - “Did element **x** occur in the stream so far?”

# Examples of streaming

- Internet traffic
- Sensor data
- Social media feeds
- Search queries
- Bank transactions
- Reading large files

# Computing the majority element of a stream

**Problem**

- Process a stream and maintain a data structure that can answer if some element occurred more times than total number of elements seen so far in the stream. If such an element exists, report it.

**Example**

Stream: `abacbccbcbabaaaababababacaaaaabaaa`

Query answer: `a` occurred more than half of the time

# Simple algorithm

**Data structure**

- Maintain a co
- Maintain a c

**Processing an**


- Increment counter for a
- Increment total counter

**Query**

- Find highest counter and report corresponding element if counter > total/2

# Streaming algorithm

**Data structure**

- Maintain one counter/element pair, initialized to (c, e) = (0, _)

**Processing an element** `a`

- If c == 0 then set (c, e) = (1, `a`)
- Else if e == `a`, then increment c by 1, else decrement c by one

**Query**

- Report e if c > 0, else report none

**If there is a majority element, it is e! But the algorithm could be wrong.**

# Simple random sample

A subset of (exactly) k elements selected at random from a larger population.

**Main property**: Each subset of k elements has the same probability of being selected.

# Simple random sample

Given the numbers 1 to 20, what is the probability of getting each of the following samples with Simple Random Sampling for k=5?

- 1,5,9,13,17
- 1,2,3,4,5
- 5,2,13,8,20
- 3,5,6,10,11,19

# Sampling in a stream

**Problem**

- Process a stream and maintain a sample of size k such that at any point in time the main property is satisfied.

**Why sampling?**

- Insights into data
- Entire stream can’t be stored
- Arbitrary queries to data (requires representative sample)

# Offline simple random sampling

**Algorithm 1:** Randomly permute your data and select the first k elements.

**Algorithm 2**: Let $i = 0$. For each element $j = 1..n$, select element $j$ with probability $(k-i)/(n-j+1)$. If element $j$ is selected, then increment $i$ by 1.

# Reservoir sampling

**Data structure**

- Maintain list of k elements
- Populate the list with the first k elements of the stream

**Processing the i-th element**

- We keep the element with probability k/i
- If we keep the element, replace it with an element selected uniformly at random

# Reservoir sampling (cont.)

**Analysis**

- Each element has the same probability of being in the sample
- The probability of an element being selected decreases with the number of elements we see
- The probability of being evicted from the sample increases over time
- Things balance: if you are selected early, your probability of staying is small

# Reservoir sampling (cont.)

**Analysis**

- i is in S if i is selected and i is not evicted in round j = i+1..m

$$P[i \in S] = P[i \text{ is selected}] \cdot \prod_{j=i+1}^{m} P[i \text{ is not evicted in round } j]$$

$$P[i \in S] = \frac{k}{i} \cdot \prod_{j=i+1}^{m}((P[j \text{ is selected}] \cdot P[i \text{ is not evicted}]) + P[j \text{ is not selected}])$$

$$P[i \in S] = \frac{k}{i} \cdot \prod_{j=i+1}^{m}(\frac{k}{j} \cdot \frac{k-1}{k} + (1 - \frac{k}{j})) = \frac{k}{m}$$