Sampling Techniques

COMPCSI 753: Algorithms for Massive Data

Instructor: Ninh Pham

University of Auckland

Parts of this material are modifications of the lecture slides from

http://mmds.org

Designed for the textbook **Mining of Massive Datasets**

by Jure Leskovec, Anand Rajaraman, and Jeff Ullman.

Auckland, Aug 17, 2020

Outline

- Sampling from a data stream
  - Sampling a fixed proportion
  - Sampling a fixed-size (reservoir sampling)

Uniformly sampling from a stream

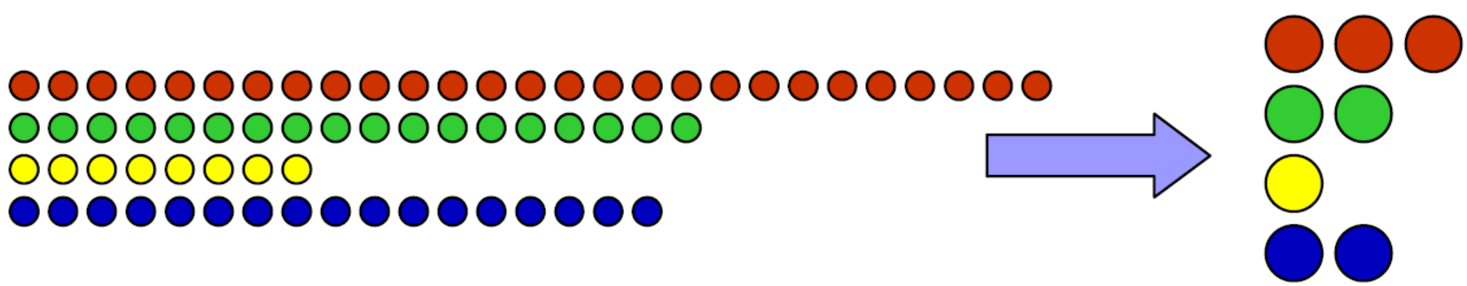
- Motivation:
  - Since we cannot store the entire stream, we store a uniform sample of the stream to answer queries.
- Goal:
  - Out of the large number of elements $\mathbf{a_1, \ldots, a_m}$, we choose a small number of elements to keep in memory.
  - Sample from the stream $\mathbf{a_1, \ldots, a_m}$ a single element $\mathbf{x}$ uniformly at random, i.e. the probability of being the sample for each element is the same:

$$\mathbf{Pr}\,[\,\mathbf{a_i} \text{ is the sample } \mathbf{x}] = \mathbf{1/m}$$

  - We draw a uniform sample over the stream, not over the universe $\mathbf{U}$.

Two different problems

- Problem 1 (easy):
  - Sample a fixed proportion of elements in the stream (**1/10**).
  - The sample size is **m/10** given the stream of **m** elements.
- Problem 2 (hard):
  - Maintain a random sample of fixed size over an infinite stream.
  - The sample size is fixed, i.e. **s** for at any time **k**.
- Both problems: **Pr [a_i is sampled] = 1/m**



Sampling a fixed proportion

- Scenario:
  - Search engine receives a stream of tuples (user, query, time).
  - Query: How many users run the same query in a single day?
  - Memory: We have enough space to store **1/10** of the stream.

- Naïve solution to keep **1/10** stream by:
  - Generate a random integer in **[0, … , 9]** for each query.
  - Store the query if the integer is **0**; otherwise, discard.

- Can we approximately answer the query?
  - Yes

Problem with naïve approach

- Scenario:
  - Search engine receives a stream of tuples (user, query, time).
  - Each user sends **x** queries once and **d** queries twice (total of **x + 2d**).
  - Query: What fraction of queries by an average user are duplicates?
  - Memory: We have enough space to store **1/10** of the stream.
- Solution: **d/(x + d)**
- Can we approximately answer the query using previous approach?
  - No

Problem with naïve approach

- Naïve solution to keep **1/10** stream for each query:
  - Sample contains **x/10** singleton queries and **2d/10** duplicate queries at least once.
  - Sample contains only **d/100** pairs of duplicates.
    - A query is sampled once with prob. **1/10**.
    - A query is sampled twice with prob. **1/10*1/10 = 1/100**.
  - Of **d** duplicates, **18d/100** appear exactly once in the sample.
    - A query is sampled at the first time with prob. **1/10**.
    - A query is not sampled at the second time with prob. **9/10**.

- Solution: $\dfrac{\frac{d}{100}}{\frac{x}{10}+\frac{d}{100}+\frac{18d}{100}} = \dfrac{\boldsymbol{d}}{\mathbf{10}\boldsymbol{x}\mathbf{+19}\boldsymbol{d}}$

Fix

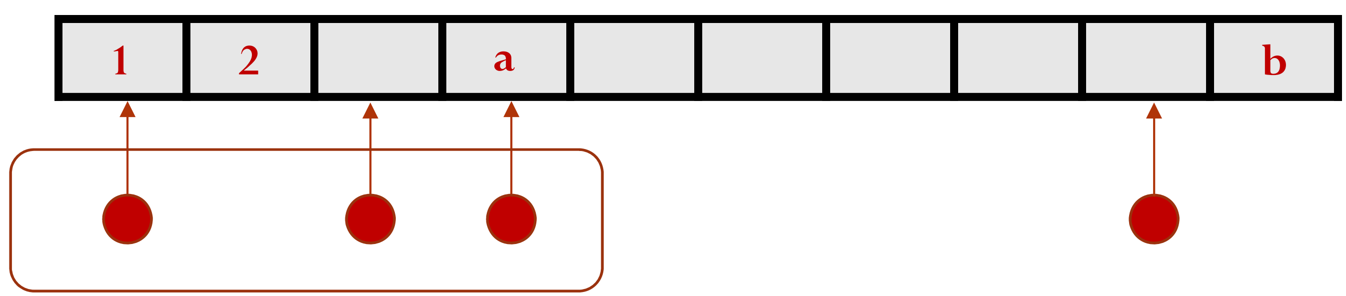
- Scenario:
  - Search engine receives a stream of tuples (user, query, time).
  - Each user sends **x** queries once and **d** queries twice (total of **x + 2d**).
  - Query: What fraction of queries by an average user are duplicates?

- Solution:
  - Pick **1/10** of users (not queries) and take all their searches in the sample.
  - Use a hash function to hash the user ID uniformly into **10** buckets and pick user IDs on the first bucket.

Generalized solution

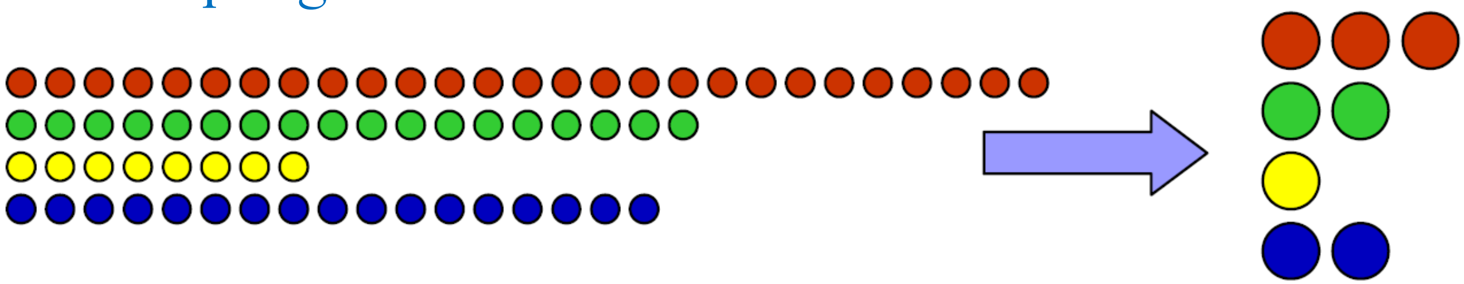
- Stream of tuples with keys:
  - Key is some subset of each tuple's components.
  - E.g. tuple(user, query, time) with user as a key.
  - Choice of key depends on application.

- To get a sample of **a/b** fraction of the stream:
  - Hash each tuple's key uniformly into **b** buckets.
  - Pick the tuples that hashed into the first **a** buckets.

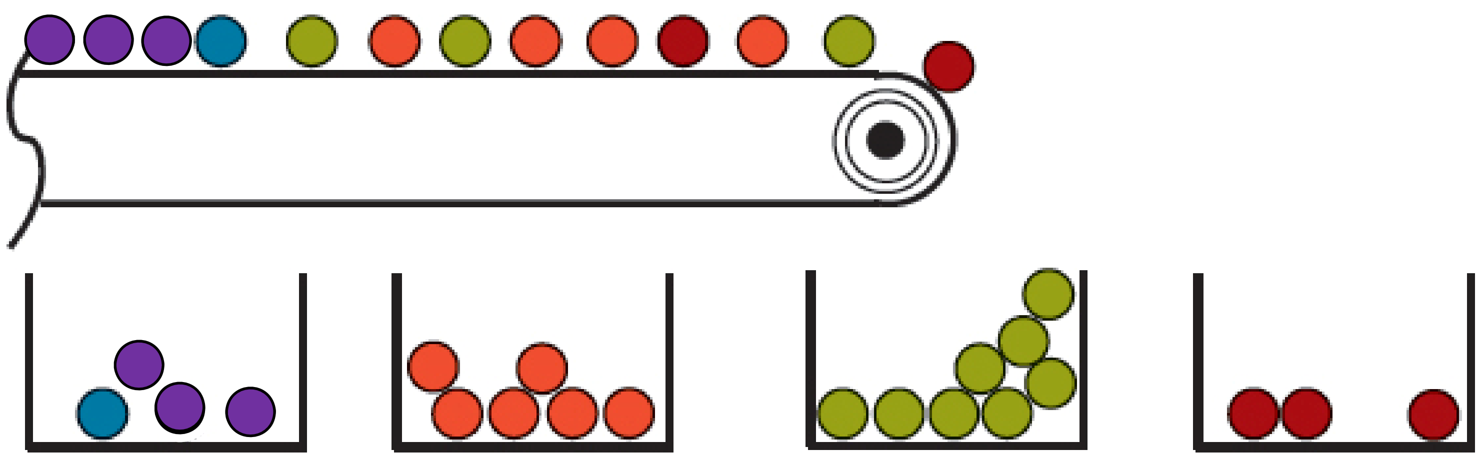


Sampling vs. hashing

- Sampling:



- Hashing:



Source: Graham Cormode, Marios Hadjieleftheriou, CACM 2009

Outline

- Sampling from a data stream
    - Sampling a fixed proportion
    - Sampling a fixed-size (reservoir sampling)

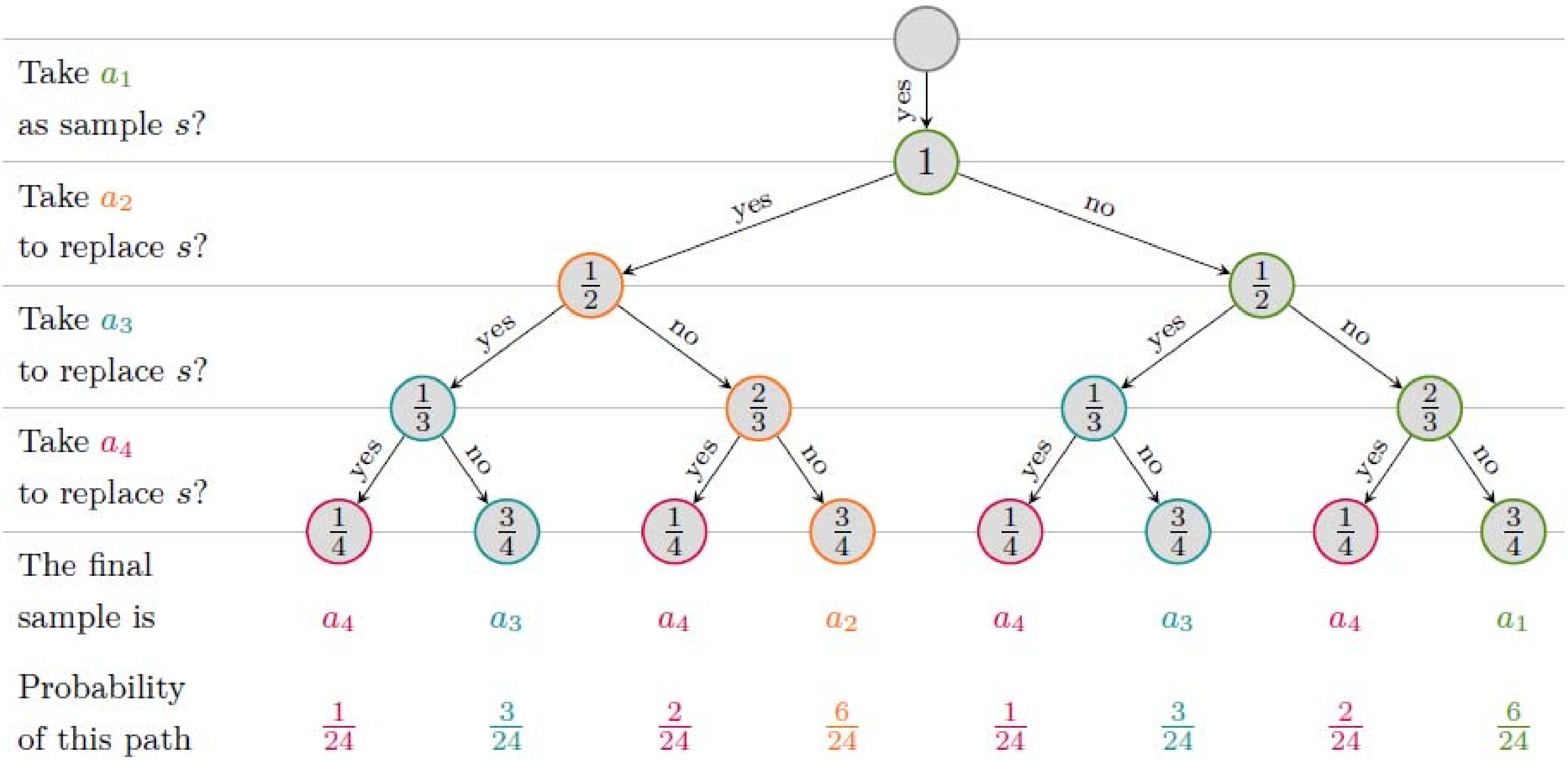
Sample a fixed size

- Challenge:
  - The sample set **S** has fixed size **s** regardless the length of the stream.
  - Uniformly sampling element from the stream

- Reservoir Sampling ($\mathbf{s} = \mathbf{1}$) [Vitter'85]:
  - $\mathbf{s} = \mathbf{a_1}$
  - $\mathbf{a_2}$ is picked as **s** with prob. $\mathbf{1/2}$
  - $\mathbf{a_3}$ is picked as **s** with prob. $\mathbf{1/3}$
  - …
  - $\mathbf{a_n}$ is picked as **s** with prob. $\mathbf{1/n}$
  - …

Decision tree of reservoir sampling

- Reservoir sampling a stream a_1, a_2, a_3, a_4



Proof

- For a stream $\mathbf{a_1, a_2, \ldots, a_L, a_{L+1}, \ldots, a_m}$, we need to prove

  $$\mathbf{Pr}\,[\,\mathbf{a_L} \text{ is sampled }] = \mathbf{1/m}$$

- Proof:
  - What is the probability that some $\mathbf{a_L}$ is the final $\mathbf{s}$ for $\mathbf{1 \le L \le m}$?
  - This happen if $\mathbf{a_L}$ is chosen as $\mathbf{s}$ and the rest $\mathbf{a_{L+1}, \ldots, a_m}$ are not.

$$
\begin{aligned}
Pr[\,a_\ell \text{ is the final } s\,] &= Pr[\,a_\ell \text{ is chosen as } s\,] \cdot \prod_{i=\ell+1}^{m} Pr[\,a_i \text{ does not replace } a_\ell \text{ as } s\,] \\
&= \frac{1}{\ell} \cdot \prod_{i=\ell+1}^{m} \left(1 - \frac{1}{i}\right) \\
&= \frac{1}{\ell} \cdot \prod_{i=\ell+1}^{m} \frac{i-1}{i} \\
&= \frac{1}{m}
\end{aligned}
$$

Generalized solution

- Reservoir Sampling ($s > 1$) [Vitter'85]:
  - Store all the first **s** elements of the stream to **S**.
  - Suppose we have seen **m-1** elements, and now the m^{th} element arrives ($m > s$).
    - With prob. **s/m**, keep m^{th} element, else discard it.
    - If we keep m^{th} element, then it replaces one of the **s** elements in **S**, picked uniformly at random.

- Theorem:
  - After **m** elements, the sample **S** contains each element seen so far with the same probability **s/m**.

Proof by induction

- Proof:
  - Assume that after **m** elements: **Pr** [$\mathbf{a_L}$ is sampled] = **s/m, 1 ≤ L ≤ m**.
  - When the (**m+1**)th element arrives, we need to show

    Pr [$\mathbf{a_L}$ is sampled] = **s/(m+1), 1 ≤ L ≤ m+1**.
- Base case:
  - When we see the first **s** elements, **Pr** [$\mathbf{a_L}$ is sampled] = **1, 1 ≤ L ≤ s**.
- (**m+1**)th element $\mathbf{a_{m+1}}$ arrives:
  - **Pr** [$\mathbf{a_L}$ is sampled] = **s/m, 1 ≤ L ≤ m** (hypothesis)
  - For $\mathbf{a_L}$ already in **S**, the probability that $\mathbf{a_L}$ is still in **S** is:
  - $\left(1 - \frac{s}{m+1}\right) + \frac{s}{m+1}\frac{s-1}{s} = \frac{m}{m+1}$
  - **Pr** [$\mathbf{a_L}$ is sampled] = **(s/m)*(m/m+1) = s/m+1, 1 ≤ L ≤ m**.
  - The new element $\mathbf{a_{m+1}}$ is sampled with prob. **s/(m+1)**

Homework

- Implement the uniformly sampling algorithms on the dataset from Assignment 1:
  - Consider each line of data as a streaming transaction in e-commerce
    - doc ID is a user ID
    - word ID is an item ID

  - Queries:
    - What is the most frequent items has been bought?
    - What is an average number of items bought by a user?