Anthony Ventresque
anthony.ventresque@ucd.ie

Big Data Programming
COMP47470

Streaming

School of Computer Science, UCD Scoil na Ríomheolaíochta, UCD

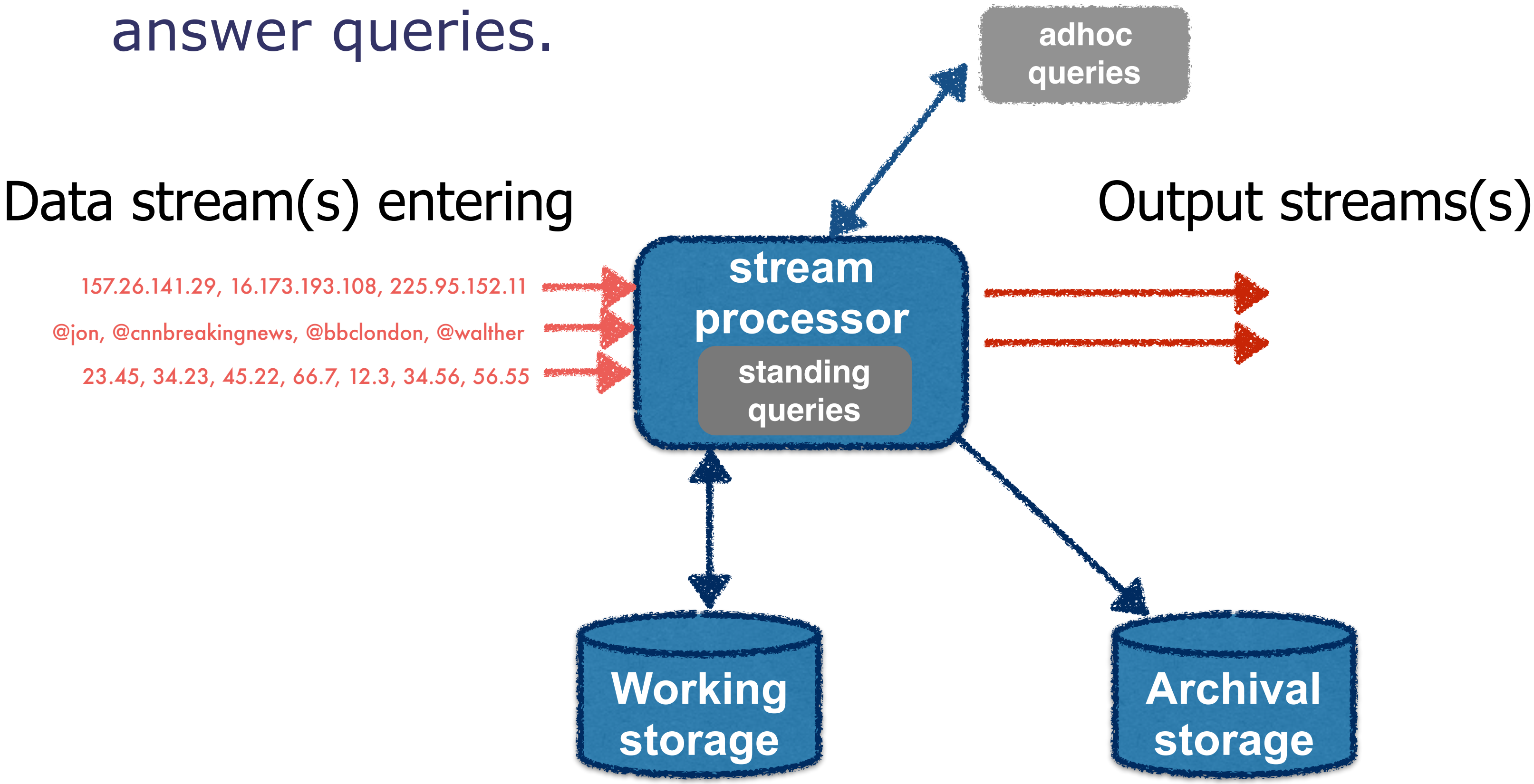
Outline

- Explain the limiting factors of data streaming & describe the different data stream models
- Describe sampling approaches for data streams
  - RESERVOIR sampling
  - MIN-WISE sampling
- Describe counter-based frequent item estimation approaches
  - MAJORITY
  - FREQUENT
  - SPACE-SAVING
- Describe BLOOM filters

DATA STREAMING

Streaming Architecture

- Maintain a summary (sketch) of the stream to answer queries.



Data Streaming Scenario

- Continuous and rapid input of data
- Limited memory to store the data (less than linear in the input size)
- Limited time to process each element
- Sequential access (no random access)
- Algorithms have one ($p=1$) or very few passes ($p=\{2,3\}$) over the data

SAMPLING

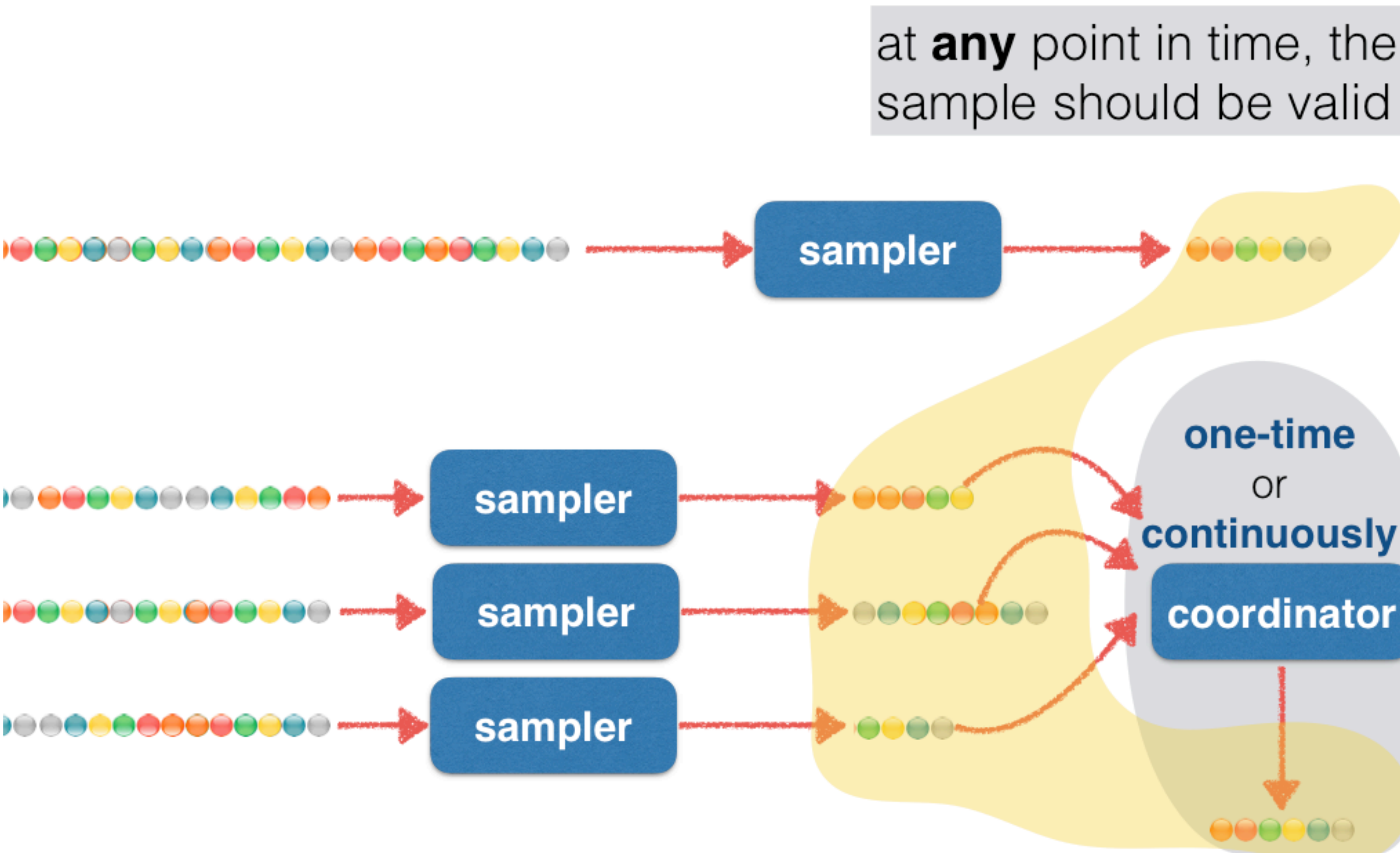
Overview

- Sampling: selection of a subset of items from a large data set
- Goal: sample retains the properties of the whole data set
- Important for drawing the right conclusions from the data

Sampling Framework

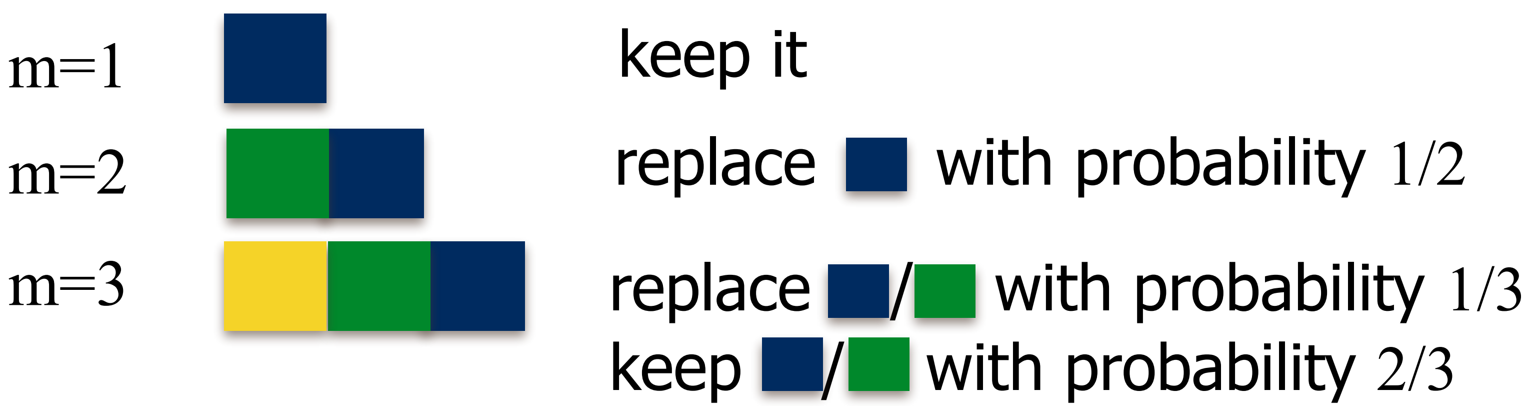
- Algorithm *A* chooses every incoming element with a certain probability
- If the element is sampled, *A* puts it into memory, otherwise the element is discarded
- Algorithm *A* may discard some items from memory after having added them
- For every query, *A* computes some function $\phi(\sigma)$ only based on the in-memory sample

Single Machine vs. Distributed



Reservoir Sampling

- Task: Given a data stream of unknown length, randomly pick *k* elements from the stream so that each element has the same probability of being chosen.



Toy example with k=1

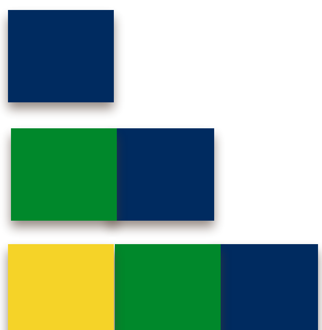
Reservoir Sampling

- Task: Given a data stream of unknown length, randomly pick *k* elements from the stream so that each element has the same probability of being chosen.

m=1

m=2

m=3



Toy example with k=1

$$P({\color{navy}\blacksquare}) = 1 \times \frac{1}{2} \times \frac{2}{3} = \frac{1}{3}$$

$$P({\color{green}\blacksquare}) = \frac{1}{2} \times \frac{2}{3} = \frac{1}{3}$$

$$P({\color{yellow}\blacksquare}) = \frac{1}{3}$$

Reservoir Sampling

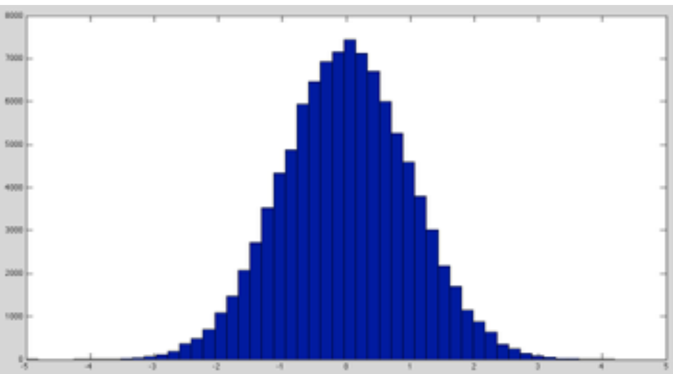
1. Sample the first *k* elements from the stream
2. Sample the *ith* element ($i>k$) with probability k/i (if sampled, randomly replace a previously sampled item)

- Limitations:
  - Wanted sample has to fit into main memory
  - Distributed sampling is not trivial

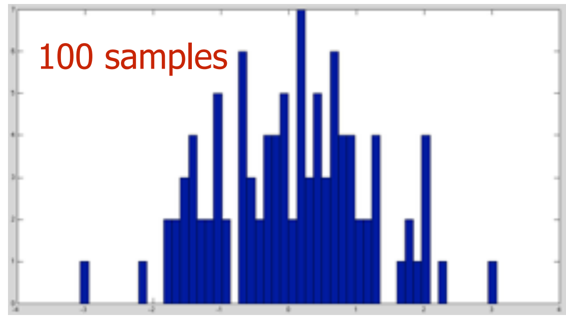
Reservoir Sampling Example

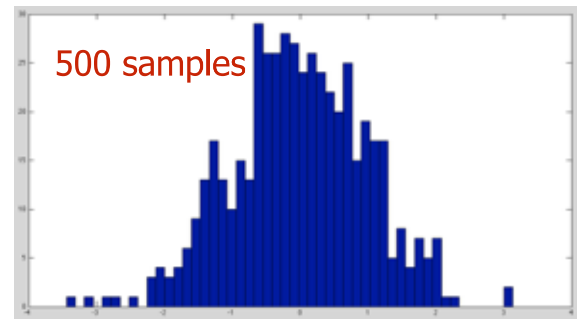
- Stream of numbers with a normal distribution *N(0,1)*
  - $|S| = 100000$
  - $k = \{100, 500, 1000, 10000\}$
- Samples are plotted in histogram form
- Expectation: with larger *k*, the histograms become more similar to the full stream histogram

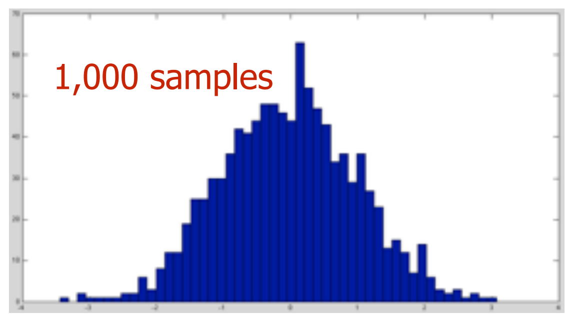
Reservoir Sampling Example

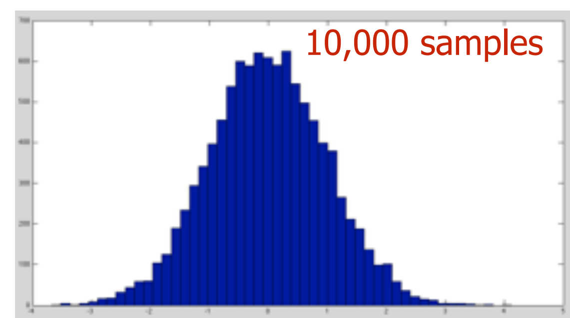


Histogram of entire stream (100,000 items)

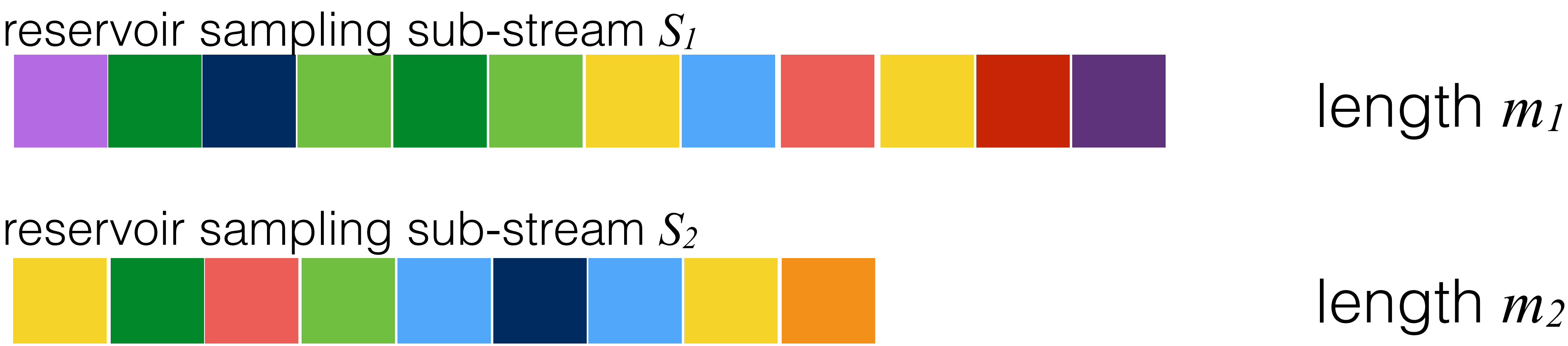








Distributed Reservoir Sampling for One-Time Sampling

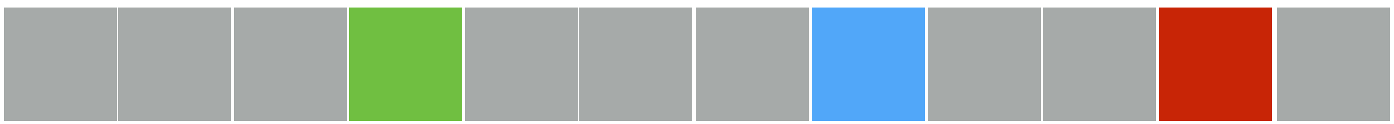


- Goal: sample sub-streams in parallel, combine with the same guarantee as the non-distributed version.
- Sub-stream output: k samples and length of sub-stream

Distributed Reservoir Sampling for One-Time Sampling

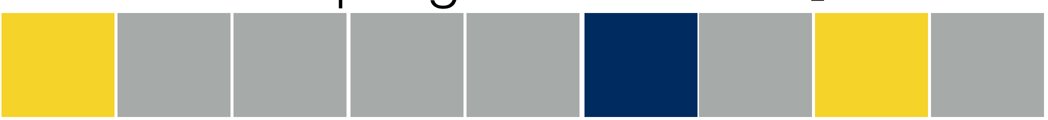
k=3

reservoir sampling sub-stream S_1



length m_1

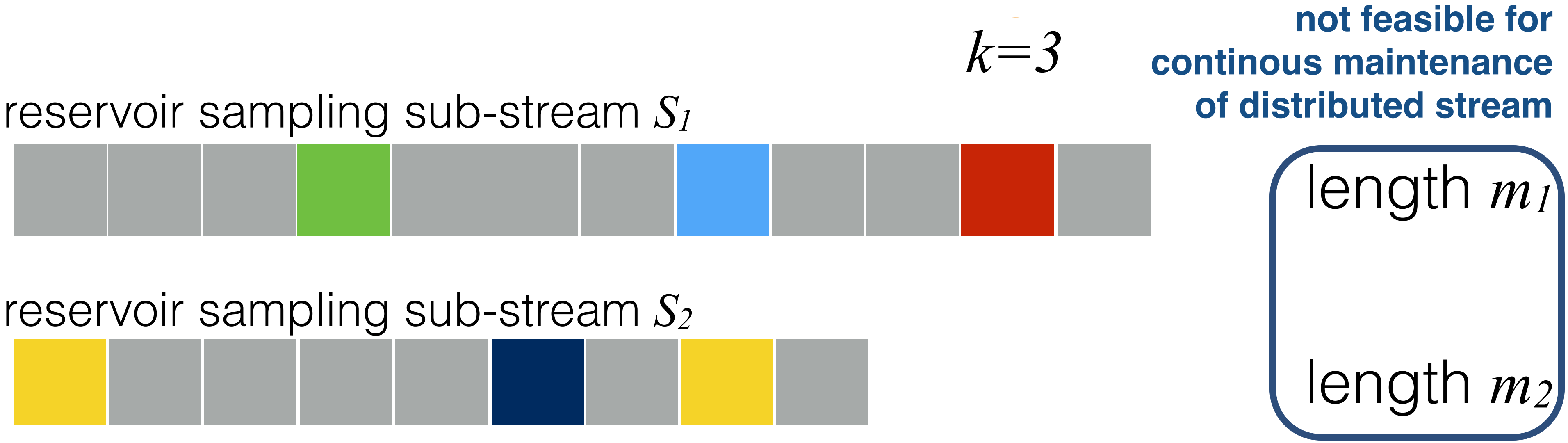
reservoir sampling sub-stream S_2



length m_2

- Combining sub-stream pairs in 2. sampling phase *k* iterations:
  - with probability p = m1/(m1 + m2) pick a sample from *S1*,
  - with (1 - p) pick a sample from *S2*

Distributed Reservoir Sampling for One-Time Sampling



- Combining sub-stream pairs in 2. sampling phase *k* iterations:
  - with probability p = m1/(m1 + m2) pick a sample from *S1*,
  - with (1 - p) pick a sample from *S2*

Min-wise Sampling

- Task: Given a data stream of unknown length, randomly pick *k* elements from the stream so that each element has the same probability of being chosen.

1. For each element in the stream, tag it with a random number in the interval [0,1].
2. Keep the *k* elements with the smallest random tags.

Min-wise Sampling

- Task: Given a data stream of unknown length, randomly pick *k* elements from the stream so that each element has the same probability of being chosen.
  - Can easily be run in a distributed fashion with a merging stage (every subset has the same chance of having the smallest tags)
  - Disadvantage: more memory/CPU intensive than reservoir sampling ("tags" need to be stored as well)

Sampling: Summary

- Advantages:
  - Low cost
  - Efficient data storage
  - Classic algorithms can be run on it (all samples should fit into main memory)
- In practical applications, we have complicating factors:
  - Time-sensitive window: only the last *x* items of the stream are of interest (e.g. in anomaly detection)
  - Sampling from databases through their indices from non-cooperative providers (e.g. Google, Bing)
    - How many car repairs does Google Places index?
    - How many documents does Google index?

FREQUENCY COUNTER ALGORITHMS

Examples

- Packets on the Internet
  - Frequent items: most popular destinations or most heavy bandwidth users
- Queries submitted to a search engine
  - Frequent items: most popular queries

MAJORITY Algorithm

- Task: Given a list of elements - is there an absolute majority (an element occurring > m/2 times)?

no absolute majority

blue wins

$c \leftarrow 0$; v unassigned;
for each i :
  if $c = 0$:
    $v \leftarrow i$;
    $c \leftarrow 1$;
  else if $v = i$:
    $c \leftarrow c+1$;
  else:
    $c \leftarrow c-1$;

MAJORITY Algorithm

- Task: Given a list of elements - is there an absolute majority (an element occurring > m/2 times)?

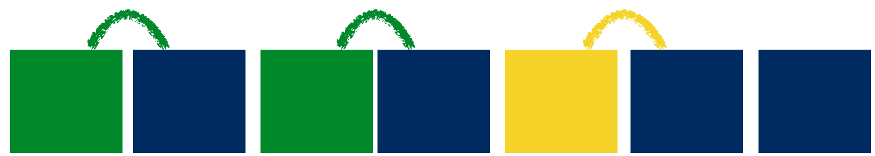
In this stream, the last item is kept.

v		b	b	b	b	b	b	b
c	0	1	0	1	2	1	0	1

- A second pass is needed to verify if the stored item is indeed the absolute majority item (count every occurrence of *b*).

MAJORITY Algorithm

- Task: Given a list of elements - is there an absolute majority (an element occurring > m/2 times)?



v		g	g	g	g	y	y	b
c	0	1	0	1	0	1	0	1

- Correctness based on pairing argument:
  - Every non-majority element can be paired with a majority one
  - After the pairing, there will still be majority elements left

FREQUENT Algorithm (Misra-Gries)

- Task: Find all elements in a sequence whose frequency exceeds 1/k fraction of the total count (i.e. frequency > m/k)

- Wanted: no false negatives, i.e. all elements with frequency > m need to be reported k

- Deterministic approach

$$
\begin{array}{l}
c[1,..(k-1)] = 0; T \leftarrow \varnothing; \\
\textbf{for each } i: \\
\quad \textbf{if } i \in T: \\
\qquad c_i \leftarrow c_i + 1; \\
\quad \textbf{else if } |T| < k-1: \\
\qquad T \leftarrow T \cup \{i\}; \\
\qquad c_i \leftarrow 1; \\
\quad \textbf{else for all } j \in T: \\
\qquad c_j \leftarrow c_j - 1; \\
\qquad \textbf{if } c_j = 0: \\
\qquad\quad T \leftarrow T \setminus \{j\};
\end{array}
$$

(k-1) counter-value pairs

FREQUENT Algorithm (Misra-Gries)

Blue and green have been estimated to each occur 3 times.

$k = 3$

$c = 0$

v_1	g	g	g	g	g	g	g	g	g	g	g	g
c_1	1	2	2	3	3	2	1	1	2	2	3	3
v_2	-	-	b	b	b	b	-	b	b	b	b	b
c_2	0	0	1	1	2	1	0	1	1	2	2	3

Stream with $m = 12$ elements; all elements with more than $\frac{m}{k}$ (i.e. $12/3 = 4$) occurrences should be reported.

FREQUENT Algorithm (Misra-Gries)

Green is estimated to have occurred once.

$k = 3$

$c = 0$

v_1	g	g	g	g	g	g	g
c_1	1	2	2	3	3	2	1
v_2	-	-	b	b	b	b	-
c_2	0	0	1	1	2	1	0

Stream with $m = 7$ elements; all elements with more than $\frac{m}{k}$ (i.e. $7/3 = 2.333$) occurrences should be reported.

FREQUENT Algorithm (Misra-Gries)

$k = 3$

$c = 0$

v_1	g	g	g	g
c_1	1	2	2	3
v_2	-	-	b	b
c_2	0	0	1	1

Recall: no false negatives wanted; blue is a **false positive** (possible, not as undesired as a false negative)

Streaming algorithms are **approximations** (estimates) of the correct answers!

Stream with $m = 4$ elements; all elements with more than $\frac{m}{k}$ (i.e. $4/3 = 1.333$) occurrences should be reported.