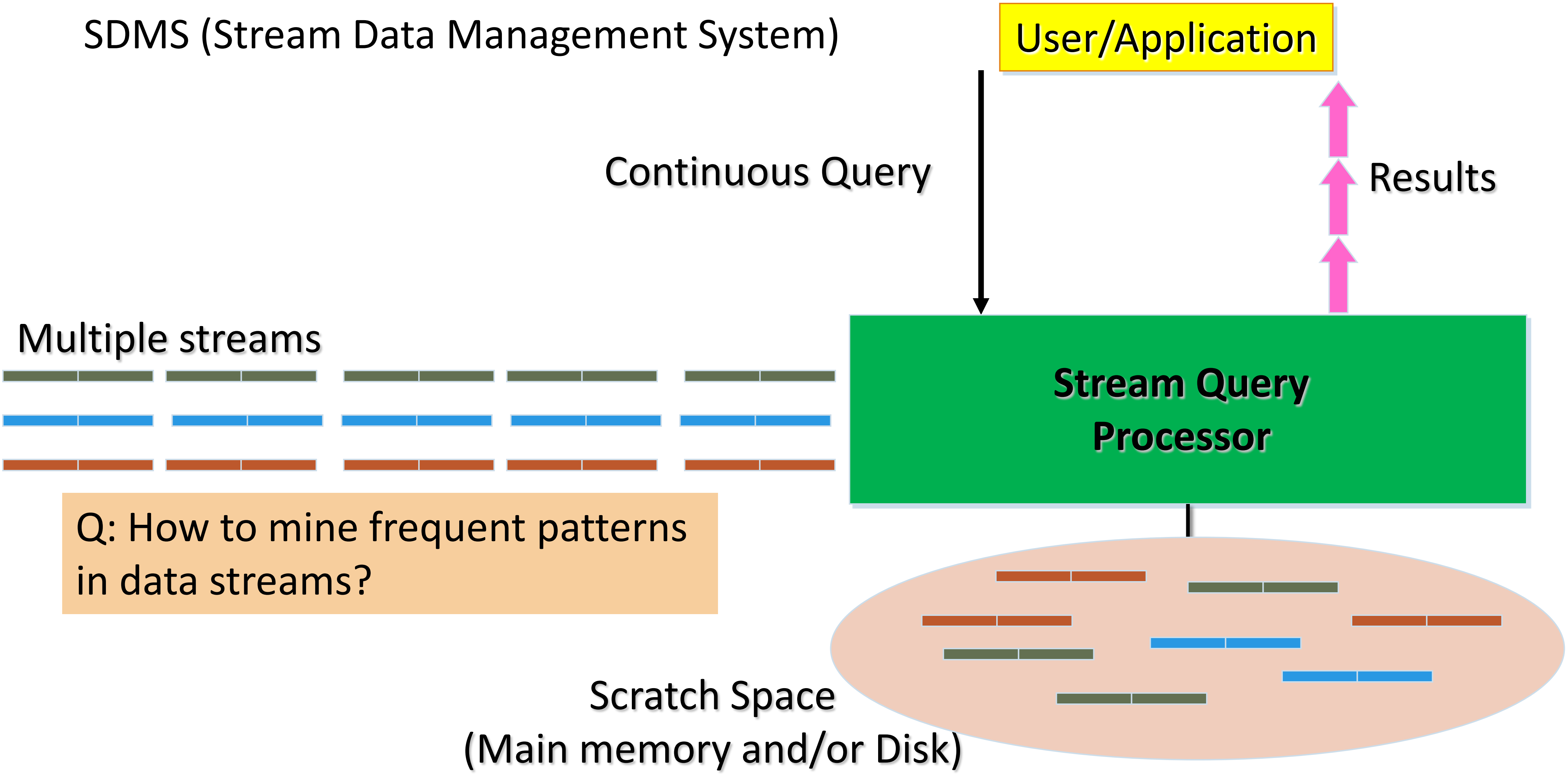
Frequent Pattern Mining in Data Streams

Challenges for Data Analysis in Data Streams

- Data Streams
  - Features: Continuous, ordered, changing, fast, huge volumn
  - Contrast with traditional DBMS (finite, persistent data sets)
- Characteristics
  - Huge volumes of continuous data, possibly infinite
  - Fast changing and requires fast, real-time response
  - Data stream captures nicely our data processing needs of today
  - Random access is expensive: single scan algorithm (*can only have one look*)
  - Store only the summary of the data seen thus far
  - Most stream data are at low-level and multi-dimensional in nature, needs multi-level and multi-dimensional processing

Architecture: Stream Data Processing



Stream Data Mining Tasks

- Stream mining vs. stream querying
  - Stream mining shares many difficulties with stream querying
    - E.g., single-scan, fast response, dynamic, …
    - But often requires less "precision", e.g., no join, grouping, sorting
  - Patterns are hidden and more general than querying
- Stream data mining tasks
  - Pattern mining in data streams
  - Multi-dimensional on-line analysis of streams
  - Clustering data streams
  - Classification of stream data
  - Mining outliers and anomalies in stream data

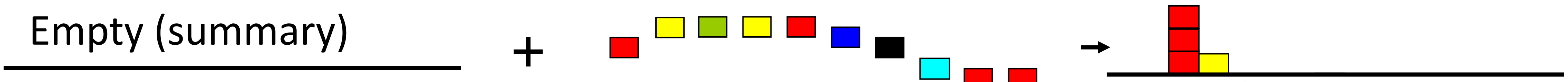
Mining Approximate Frequent Patterns

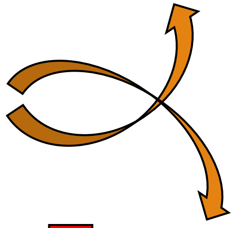
- Mining precise frequent patterns in stream data: Unrealistic
  - Cannot even store them in a compressed form (e.g., FPtree)
- Approximate answers are often sufficient for pattern analysis
  - Ex.: A router
    - is interested in all flows whose frequency is at least 1% (σ) of the entire traffic stream seen so far
    - and feels that 1/10 of σ (ε = 0.1%) error is comfortable
- How to mine frequent patterns with good approximation?
  - Lossy Counting Algorithm (Manku & Motwani, VLDB'02)
    - Major ideas: Not to keep the items with very low support count
    - Advantage: Guaranteed error bound
    - Disadvantage: Keeping a large set of traces

Lossy Counting for Frequent Single Items

Divide stream into ‘buckets’ (bucket size is 1/ ε = 1000)

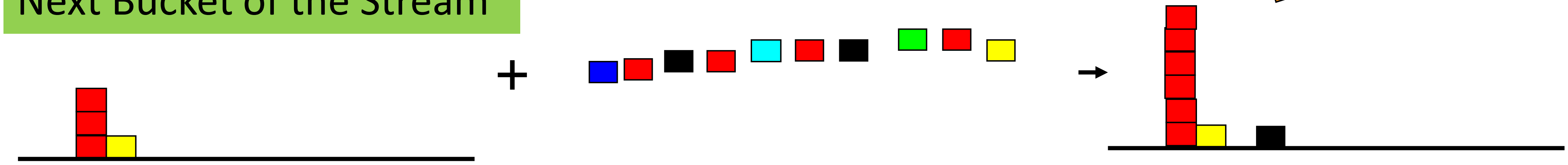
First Bucket of the Stream





At bucket boundary, decrease all counters by 1

Next Bucket of the Stream



Approximation Guarantee

- Given: (1) support threshold: σ, (2) error threshold: ε, and (3) stream length N
- Output: items with frequency counts exceeding $(\sigma - \varepsilon)$ N
- How much do we undercount?

If stream length seen so far = N and bucket-size = $1/\varepsilon$

then frequency count error $\leq$ # of buckets

$= N/\text{bucket-size} = N/(1/\varepsilon) = \varepsilon N$

- Approximation guarantee
  - No false negatives
  - False positives have true frequency count at least $(\sigma - \varepsilon)N$
  - Frequency count underestimated by at most εN

Other Issues and Recommended Readings

- Other issues on pattern discovery in data streams
  - Space-saving computation of frequent and top-k elements (Metwally, Agrawal, and El Abbadi, ICDT'05)
  - Mining approximate frequent k-itemsets in data streams
  - Mining sequential patterns in data streams
- Recommended Readings
  - G. Manku and R. Motwani, “Approximate Frequency Counts over Data Streams”, VLDB’02
  - A. Metwally, D. Agrawal, and A. El Abbadi, “Efficient Computation of Frequent and Top-k Elements in Data Streams”, ICDT'05