

CSCSE 689: Special Topics in Modern Algorithms for Data Science

Lecture 26

Samson Zhou

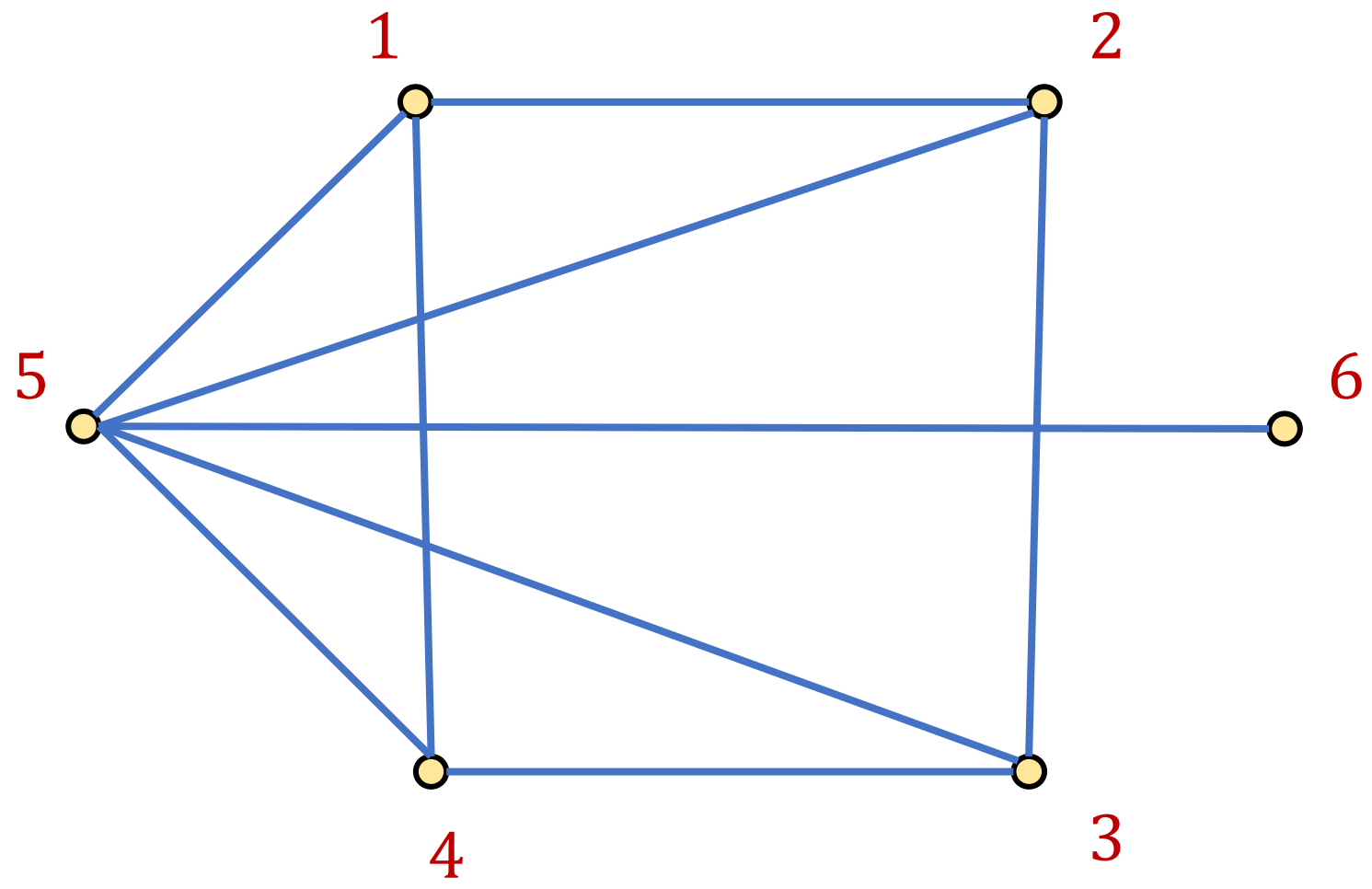
Presentation Schedule

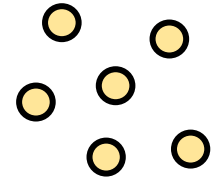
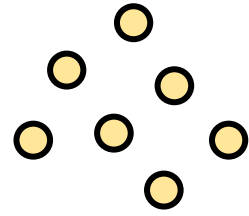
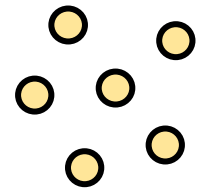
- **November 27:** Chunkai, Jung, Galaxy AI
- **November 29:** STMI, Anmol, Jason
- **December 1:** Bokun, Ayesha, Dawei, Lipai

Streaming Model

- **Input:** Elements of an underlying data set S , which arrives sequentially
- **Output:** Evaluation (or approximation) of a given function
- **Goal:** Use space *sublinear* in the size m of the input S

47 72 81 10 14 33 51 29 54 9 36 46 10





Linear Sketch

- Suppose stream S induces a frequency vector f
- Algorithm framework:
 - Generate a random matrix A and maintain $A \cdot f$
 - Apply a post-processing function $g(A \cdot f)$ as the output
- What algorithms have we discussed that fit this framework?

Linear Sketch

- What algorithms have we discussed that fit this framework?
 - **YES**: AMS, CountSketch, CountMin, sparse recovery, distinct elements, coresets construction for clustering
 - **NO**: Greedy algorithm for maximal matching, connectivity, bipartiteness, MisraGries

Linear Sketch

- What algorithms have we discussed that fit this framework?
 - **YES**: AMS, CountSketch, CountMin, sparse recovery, distinct elements, coresets construction for clustering
 - **NO**: Greedy algorithm for maximal matching, connectivity, bipartiteness, MisraGries
- Which of these algorithms work for insertion-deletion streams?

Linear Sketch

- What algorithms have we discussed that fit this framework?
 - **YES**: AMS, CountSketch, CountMin, sparse recovery, distinct elements, coresets construction for clustering
 - **NO**: Greedy algorithm for maximal matching, connectivity, bipartiteness, MisraGries
- Which of these algorithms work for insertion-deletion streams?
 - **YES**: AMS, CountSketch, CountMin, sparse recovery, distinct elements, coresets construction for clustering
 - **NO**: Greedy algorithm for maximal matching, connectivity, bipartiteness, MisraGries

Linear Sketch

- **Theorem [LiNguyenWoodruff14]**: For sufficiently long data streams with arbitrarily large coordinates at intermediate stages of the stream, any one-pass insertion-deletion streaming algorithm can be implemented with a linear sketch

Sliding Window Model

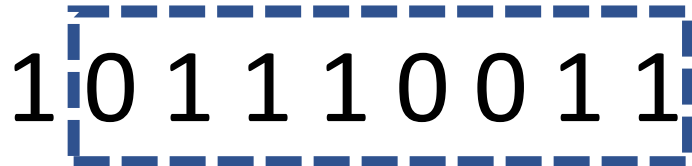
- **Input:** Elements of an underlying data set S , which arrives sequentially
- **Output:** Evaluation (or approximation) of a given function
- **Goal:** Use space *sublinear* in the size m of the input S
- **Sliding Window:** “Only the m most recent updates form the underlying data set S ”

1 0 1 1 1 0 0 1

Sliding Window Model

- **Input:** Elements of an underlying data set S , which arrives sequentially
- **Output:** Evaluation (or approximation) of a given function
- **Goal:** Use space *sublinear* in the size m of the input S
- **Sliding Window:** “Only the m most recent updates form the underlying data set S ”

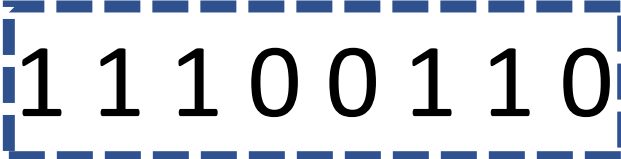
1 0 1 1 1 0 0 1 1



Sliding Window Model

- **Input:** Elements of an underlying data set S , which arrives sequentially
- **Output:** Evaluation (or approximation) of a given function
- **Goal:** Use space *sublinear* in the size m of the input S
- **Sliding Window:** “Only the m most recent updates form the underlying data set S ”

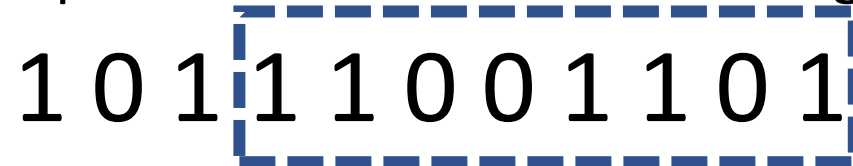
1 0 1 1 1 0 0 1 1 0



Sliding Window Model

- **Input:** Elements of an underlying data set S , which arrives sequentially
- **Output:** Evaluation (or approximation) of a given function
- **Goal:** Use space *sublinear* in the size m of the input S
- **Sliding Window:** “Only the m most recent updates form the underlying data set S ”
 - Emphasizes recent interactions, appropriate for time sensitive settings

1 0 1 1 1 0 0 1 1 0 1



Sliding Window Model

- **Consumer analytics:** Consumer patterns may be sensitive to temporal trends or seasonal shifts



Sliding Window Model

- **Data retention policy:** the Facebook data policy says user search histories are stored for **6** months, the Apple differential privacy overview says collected user information is retained for **3** months, and the Google data retention policy states that browser information may be stored for up **9** months

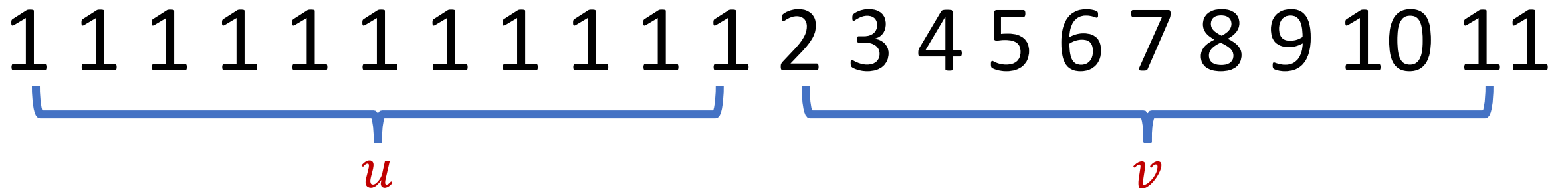


Sliding Window Model and Linear Sketches

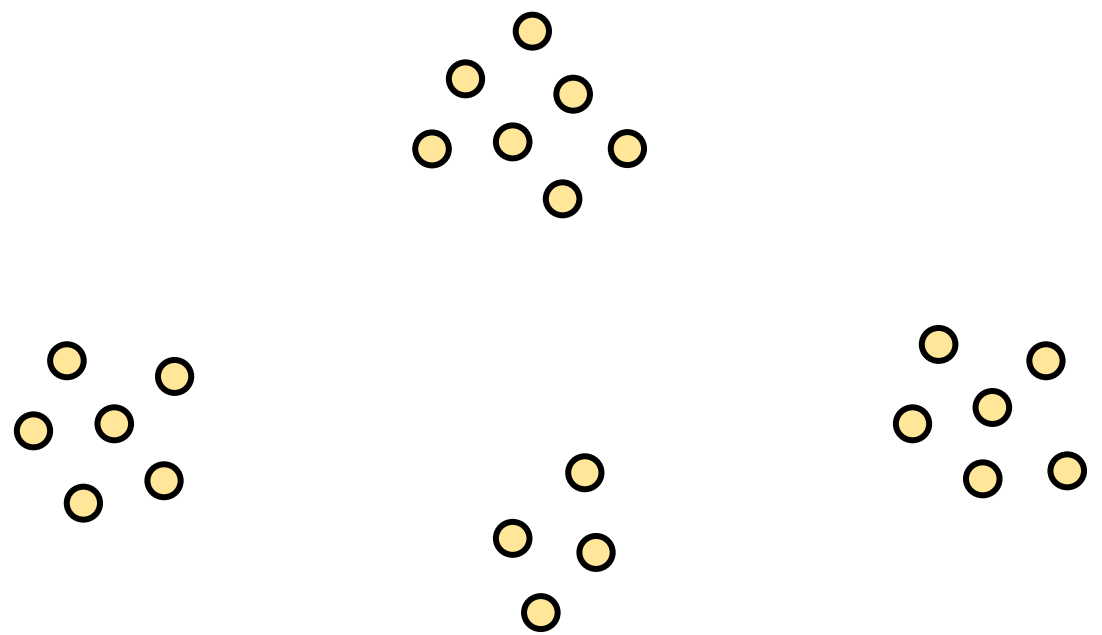
- Can we use linear sketches for the sliding window model?

Sliding Window Model and Linear Sketches

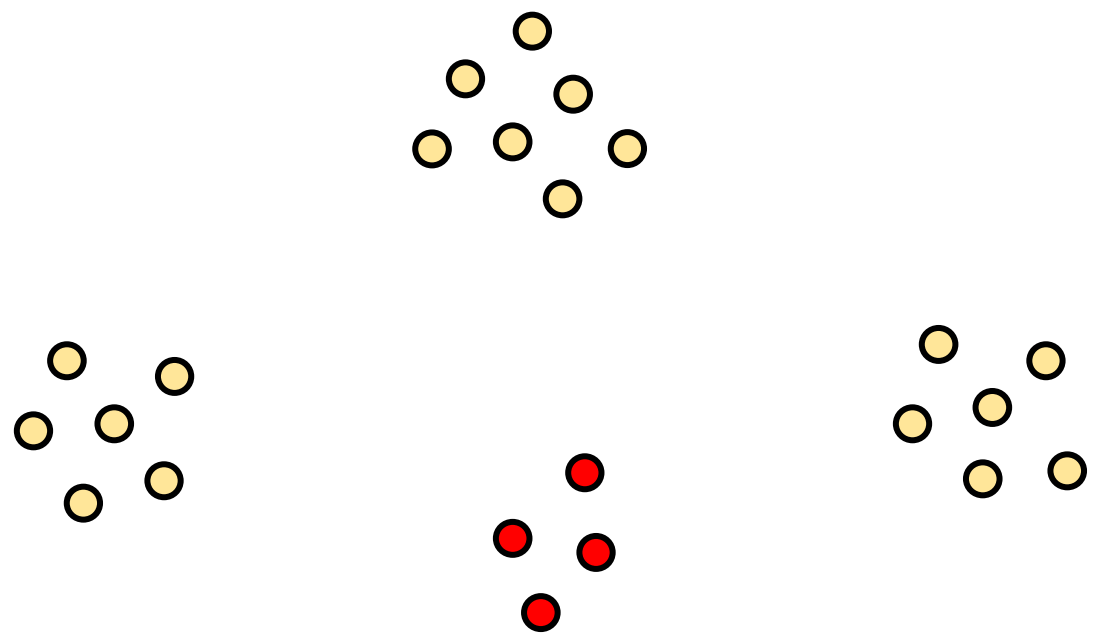
- Can we use linear sketches for the sliding window model? YES/NO
- Suppose $F_2(u) = n^2$ and $F_2(v) = n$.
- Then $A(u + v)$ might (and should) think $F_2(u + v) = n^2$
- If u expires, then what do we do with $A(u + v)$?



Sliding Window Model and Sampling

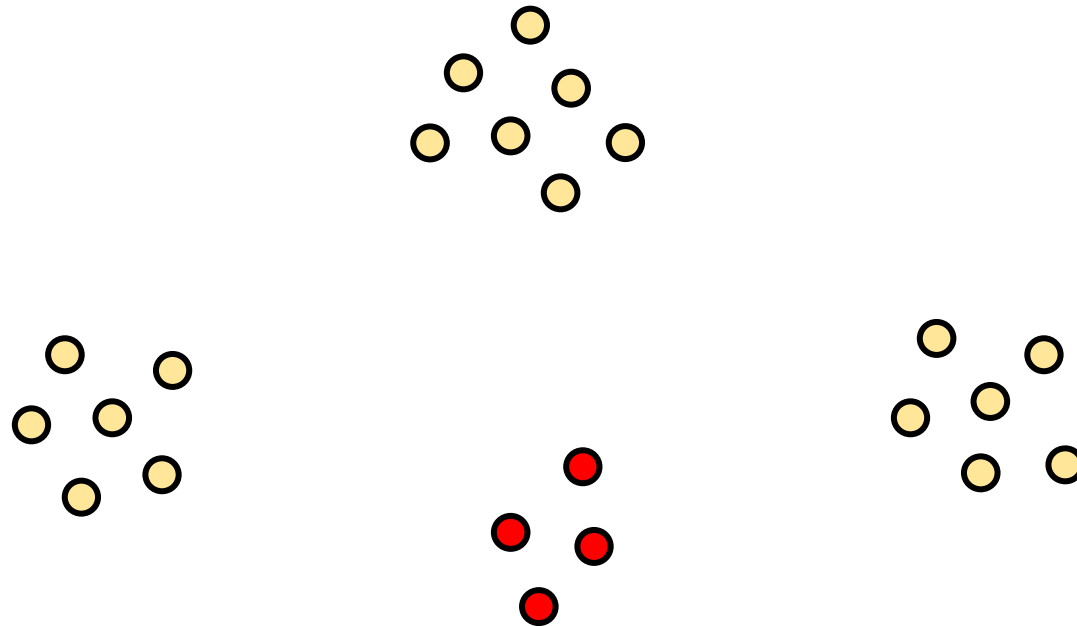


Sliding Window Model and Sampling



Sliding Window Model and Sampling

- **Recall:** we can sample each point with probability proportional to how “important” the point is

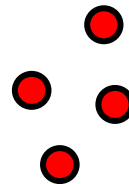
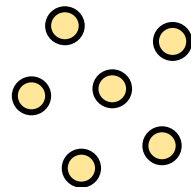
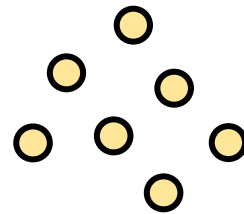


Last Time: Sensitivity Sampling

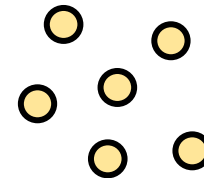
- Recall: $\frac{kd}{\varepsilon^2} \cdot \log \frac{n\Delta}{\varepsilon} \cdot \sum_{x \in X} s(x)$ points sampled
- $\sum_{x \in X} s(x) = O_Z(k)$
- In total, roughly $\frac{k^2 d}{\varepsilon^2} \cdot \log \frac{n\Delta}{\varepsilon}$ points sampled in expectation

Sliding Window Model and Sampling

- **Recall**: we can sample each point with probability proportional to how “important” the point is
- We can also consider the “importance” of each point with respect to the following points in the stream

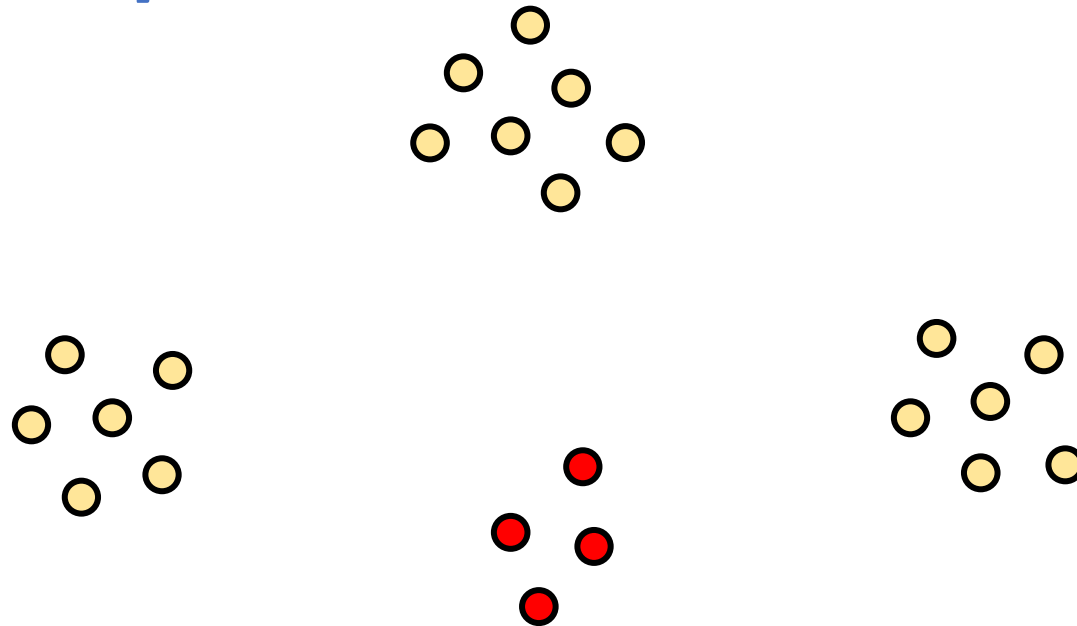


- **Intuition**: previous points do not matter in the importance of a point because the previous points can be expired



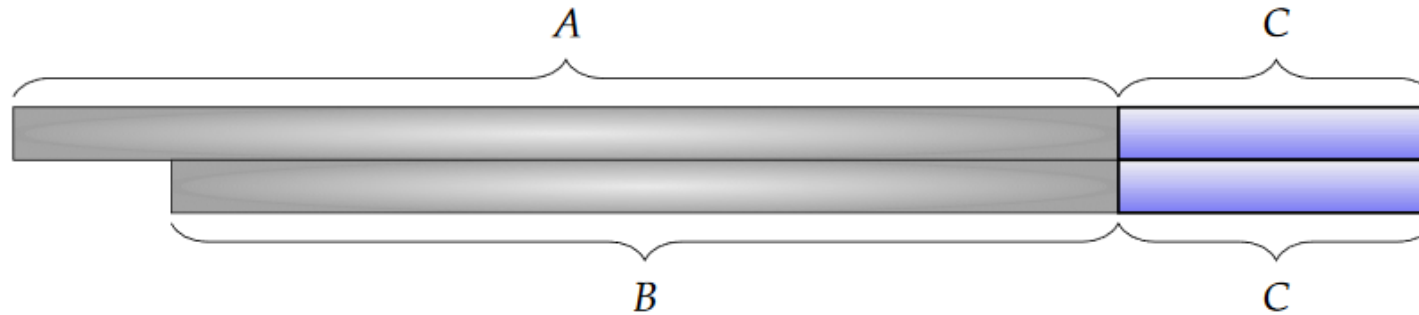
Sliding Window Model and Sampling

- **Theorem:** There exists a sliding window model algorithm that samples roughly $\frac{k^2 d}{\epsilon^2} \cdot \text{polylog} \frac{n\Delta}{\epsilon}$ points and outputs a coresets for (k, z) -clustering [WoodruffZhongZhou23]



Sliding Window Algorithms

- Suppose we are trying to approximate some given function
 - Suppose we have a streaming algorithm for this function
 - Suppose this function is “smooth”: If $f(B)$ is a “good” approximation to $f(A)$, then $f(B \cup C)$ will always be a “good” approximation to $f(A \cup C)$.



- Smooth histogram framework [\[BravermanOstrovsky07\]](#) gives a sliding window algorithm for this function

Smooth Histogram

- Suppose we are trying to approximate some given function
- Smooth histogram framework [BO07] gives a sliding window algorithm for this function
- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window

Smooth Histogram

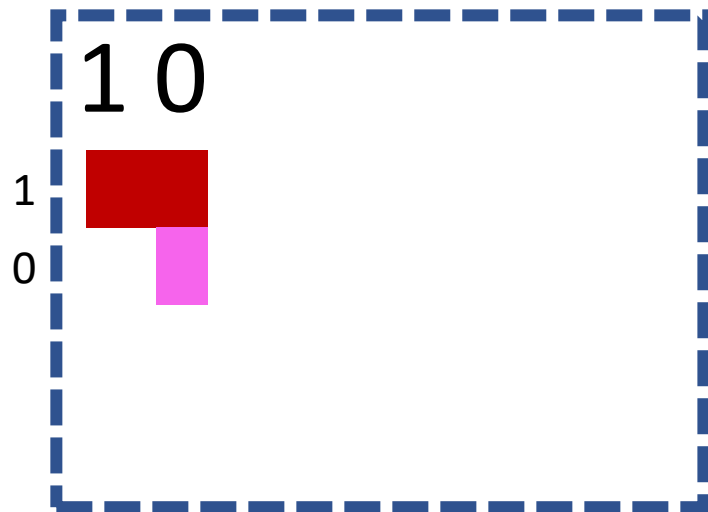
- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window



- **Example:** Number of ones in sliding window (2-approximation)

Smooth Histogram

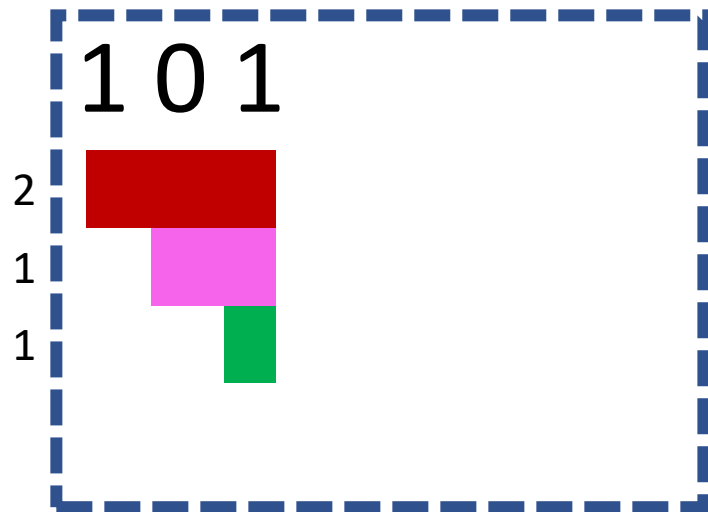
- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window



- **Example:** Number of ones in sliding window (2-approximation)

Smooth Histogram

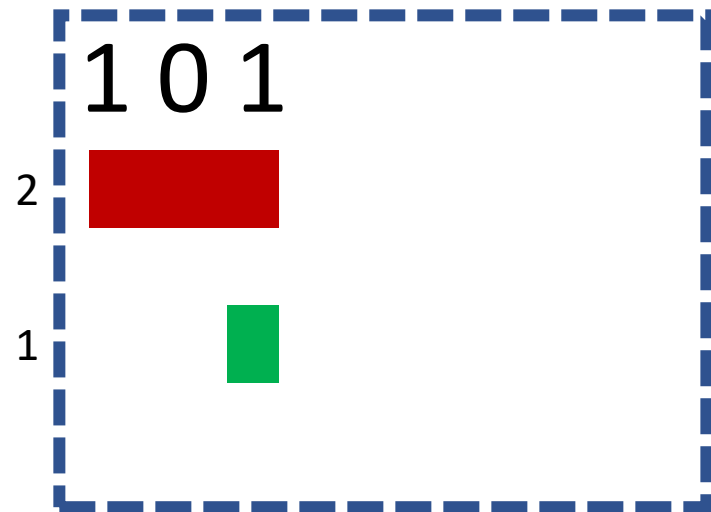
- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window



- **Example:** Number of ones in sliding window (2-approximation)

Smooth Histogram

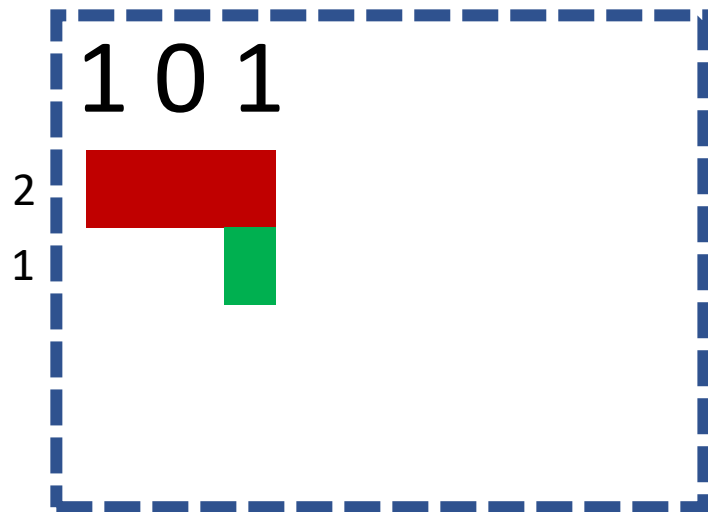
- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window



- **Example:** Number of ones in sliding window (2-approximation)

Smooth Histogram

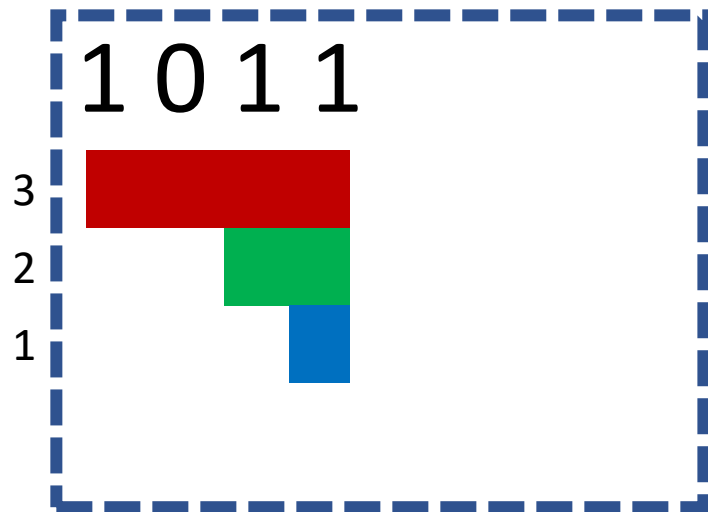
- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window



- **Example:** Number of ones in sliding window (2-approximation)

Smooth Histogram

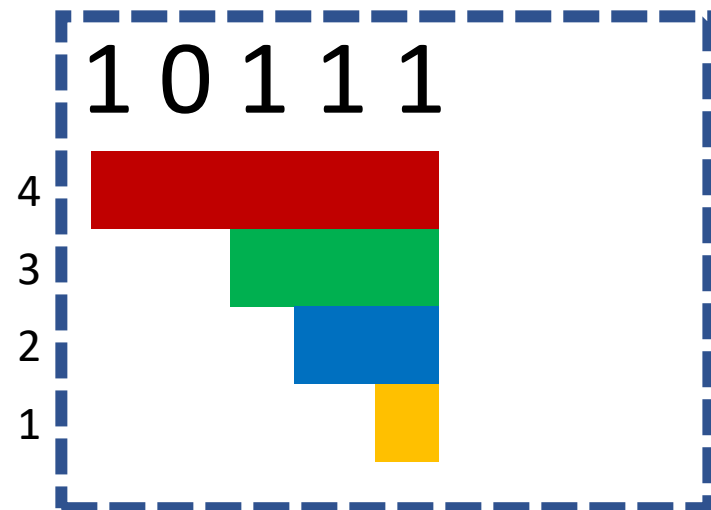
- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window



- **Example:** Number of ones in sliding window (2-approximation)

Smooth Histogram

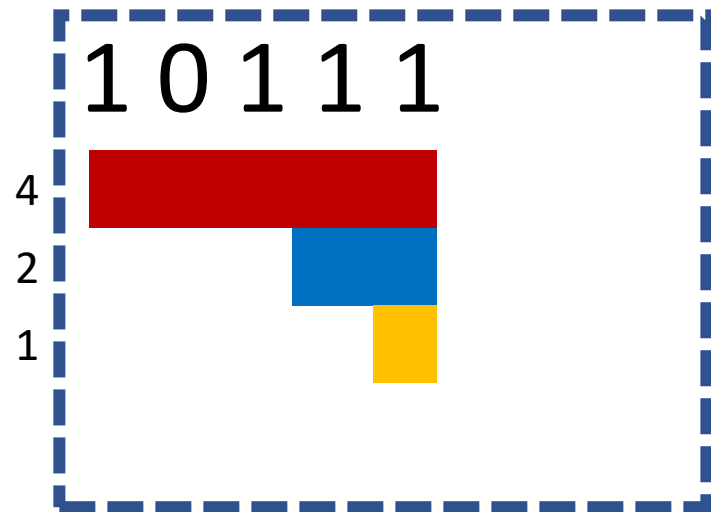
- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window



- **Example:** Number of ones in sliding window (2-approximation)

Smooth Histogram

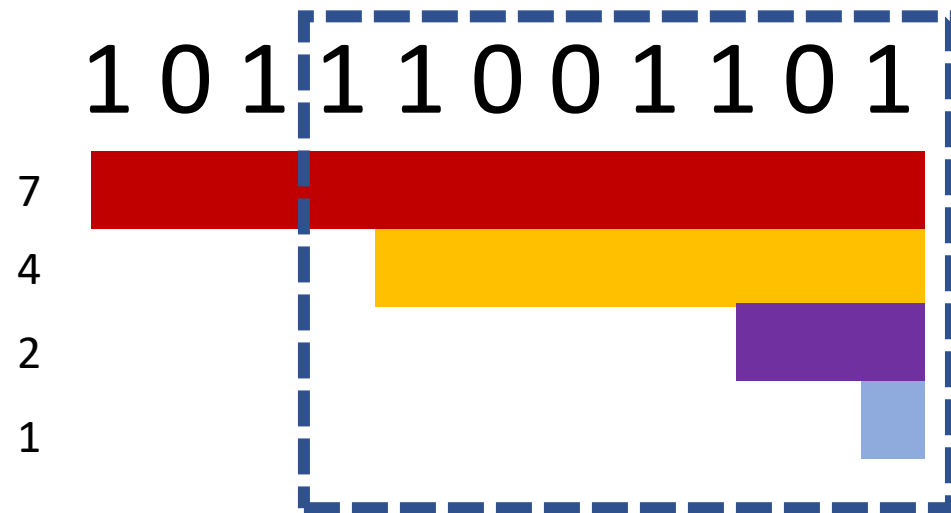
- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window



- **Example:** Number of ones in sliding window (2-approximation)

Smooth Histogram

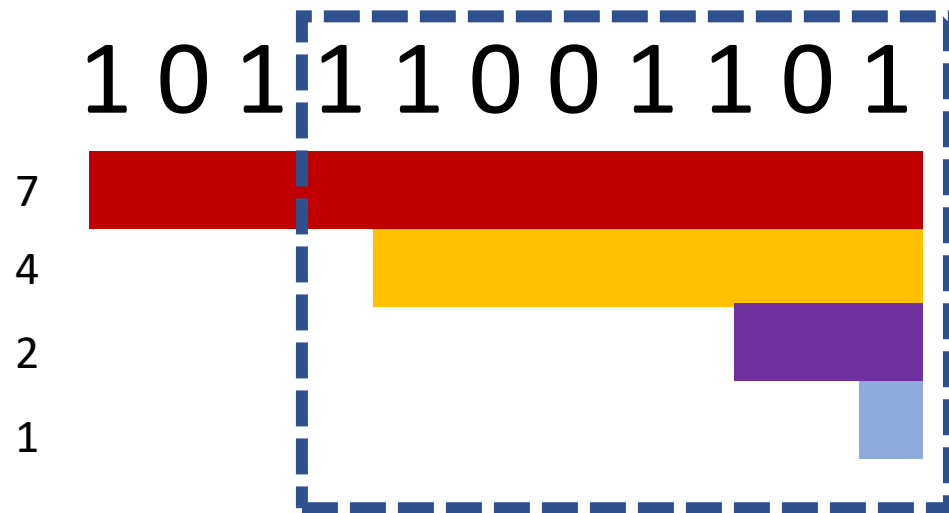
- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window



- **Example:** Number of ones in sliding window (2-approximation)

Smooth Histogram

- Start a new instance of the streaming algorithm (along with existing instances) each time a new element arrives
- Each time there are three instances that report “close” values, delete the middle one
- Use different checkpoints to “sandwich” the sliding window



- **Example:** Number of ones in sliding window (2-approximation)
- Number of ones in sliding window is at least 4 and at most 7
- 4 is a good approximation

Smooth Histogram

- Converts a streaming algorithm for a smooth function into a sliding window algorithm